MagicCryptoMVP
Software Version 1.0.0

Non-Proprietary FIPS 140-2 Security Policy
Document Version: 1.0.1

January 5, 2023

Table of Contents

List of Tables

List of Figures

1 Introduction

This document defines the Security Policy for the Dream Security MagicCryptoMVP module, hereafter denoted the Module. The MagicCryptoMVP cryptographic module supports various types of cryptographic algorithms to provide cryptographic services.

Functions were designed to make the interface easy to use by application program developers. This has the advantage of making it easy to use various cryptographic algorithms without having to study them in depth. In addition, it has been designed with high scalability, so that it is easy to add new algorithms, and able to minimize modification of application programs due to modification of cryptographic modules.

The MagicCryptoMVP cryptographic module is in the form of a software shared library that runs on Linux operating systems. The MagicCryptoMVP uses encryption API functions to provide cryptographic services such as encryption/decryption, hash, digital signatures generation and verification, message authentication (MAC), secret key generation, and key pair generation.

Table 1 – Cryptographic Module Configurations

Module	Component	Operation System
MagicCryptoMVP	libMagicCryptoMVP.so	Linux kernel 4.19 (64bits)

The Module is intended for use by US Federal agencies or other markets that require FIPS 140-2 validated cryptographic module.

The MagicCryptoMVP cryptographic complies with Security Level 1 of the FIPS 140-2 standard. The security levels for the Module are as follows:

Table 2 – Security Level of Security Requirements

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

1.1 Module Description and Cryptographic Boundary

The MagicCryptoMVP is a software, multi-chip standalone cryptographic module that runs on a general-purpose computer. The module is used as a shared library, linked to an application program, and exists in the auxiliary memory of the system.

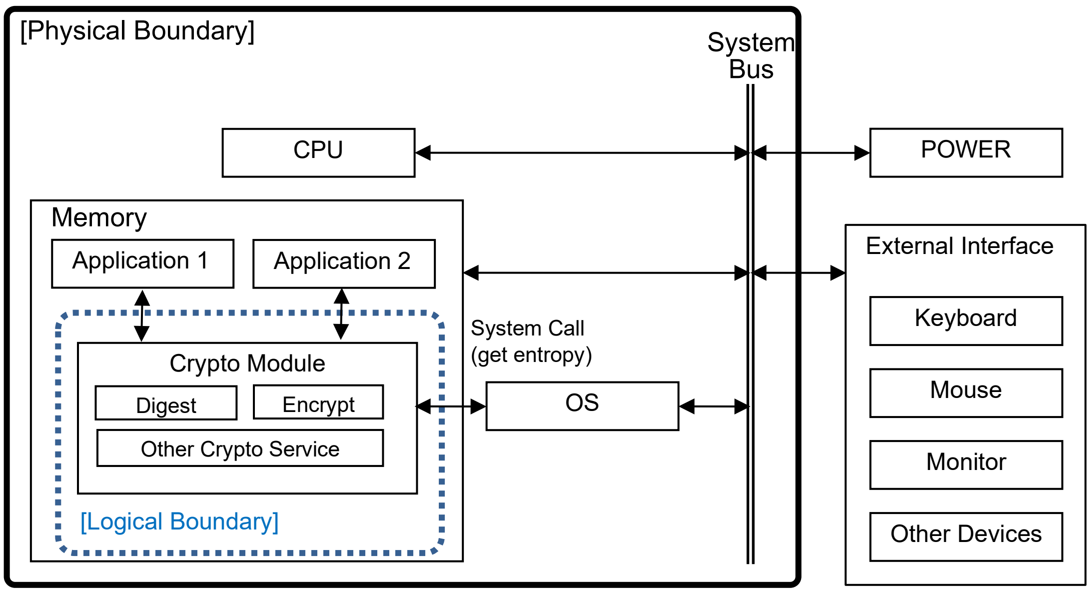

Figure 1 – Cryptographic Module Boundary

The MagicCryptoMVP cryptographic module is composed of a software library and provides a logical interface through API. Through this logical interface, four interfaces of data input and output, control input, and status output are supported. The API provides an interface in the form of a function for use in an application program.
Data input/output interface is input/output parameter of the cryptographic module API function. Control input is the cryptographic module API function call in an application program. Status output can be viewed through the return value of the cryptographic module API function or the output parameter of the status check function.

Table 3 – Ports and Interfaces

Interface	Logical Interface
Data input	API Input parameters
Data output	API Output parameters
Control input	API function calls
Status output	Return values of all API functions. Status output through status check function

The cryptographic module defined IN and OUT through the preprocessor of C language to distinguish data for input and data for output. Control input is distinguished with input parameters of the function and status output is distinguished with return values of the function.

MagicCryptoMVP does not have a direct physical connection. The encryption module output data is obtained through the data output interface.

1.2 Modes of Operation

The MagicCryptoMVP cryptographic module supports FIPS Mode only.

2 Cryptographic Functionality

The following algorithms are supported in MagicCryptoMVP.

Table 4 – Approved Algorithms

Cert	Algorithm	Mode	Description	Functions/Caveats
#A948	AES [197]	ECB [38A]	Key Sizes: 128, 192, 256	Encrypt, Decrypt
		CBC [38A]	Key Sizes: 128, 192, 256	Encrypt, Decrypt
		CTR [38A]	Key Sizes: 128, 192, 256	Encrypt, Decrypt
		CCM [38C]	Key Sizes: 128, 192, 256 Tag Len: 32, 48, 64, 80, 96, 112, 128	Authenticated Encrypt, Authenticated Decrypt, Message Authentication
		GCM [38D]	Key Sizes: 128, 192, 256 Tag Len: 32, 48, 64, 80, 96, 112, 128	Authenticated Encrypt, Authenticated Decrypt
		GMAC [38D]	Key Sizes: 128, 192, 256 Tag Len: 32, 48, 64, 80, 96, 112, 128	Message Authentication
Vender Affirmed	CKG [133]	Section 4 & 5.1	Key Pairs for Digital Signature Schemes	Key Generation
		Section 4 & 5.2	Key Pairs for Key Establishment	
		Section 4 & 6.1	Derivation of Symmetric Keys	
		Section 6.2.2	Symmetric Keys Derived from Pre-Existing Key	
#A948	DRBG [90A]	Hash	SHA(256)	Deterministic Random Bit Generation Security Strength = 256 No assurance of the minimum strength of generated keys
#A948	ECDSA [186]		P-224, P-256, K-283, B-283, P-384, P-521	Key Generation
			P-224, P-256, K-283, B-283, P-384, P-521	PKV
			P-224, P-256, K-283, B-283, P-384, P-521	Signature Generation
			P-224, P-256, K-283, B-283, P-384, P-521	Signature Verification
#A948	HMAC [198]	SHA-256	Key Sizes: <range or enumeration> λ = *32, 48, 64, 80, 96,* 128, 192, 256	Message Authentication, KDF Primitive, Password Obfuscation
		SHA-384	Key Sizes: <range or enumeration> λ = *32, 48, 64, 80, 96,* 192, 256, 320, 384	

Cert	Algorithm	Mode	Description	Functions/Caveats
		SHA-512	Key Sizes: <range or enumeration> λ = *32, 48, 64, 80, 96,* 256, 320, 384, 448, 512	
#A948	KBKDF [108]	Counter	SHA(256)	Key Based Key Derivation
#A948	KTS-RSA [56Br2]	KTS-OAEP	n = 2048 n = 3072	Key establishment methodology provides 112 or 128 bits of encryption strength Provides Key Encapsulation and Key Un-Encapsulation
#A948	RSA [186]	X9.31	n = 2048 n = 3072	Key Generation
		PKCS1_v1.5	n = 2048 SHA(256, 384, 512) n = 3072 SHA(256, 384, 512)	Signature Generation
		PSS	n = 2048 SHA(256, 384, 512) n = 3072 SHA(256, 384, 512)	Signature Generation
		PKCS1_v1.5	n = 2048 SHA(256, 384, 512) n = 3072 SHA(256, 384, 512)	Signature Verification
		PSS	n = 2048 SHA(256, 384, 512) n = 3072 SHA(256, 384, 512)	Signature Verification
#A948	SHS [180]	SHA-224 SHA-256 SHA-384 SHA-512		Message Digest Generation, Password Obfuscation

2.1 Critical Security Parameters

Cryptographic keys, such as secret keys, private keys, and public keys used in cryptographic modules, exist in the context of the session in the form of an object. Applications that have obtained the correct session information can perform cryptographic services through the CSP in the context. Applications that have not obtained session information cannot access to the context by the session management mechanism of the cryptographic module and the memory management mechanism of the operating system.

Table 5 – Critical Security Parameters (CSPs)

Service	CSP	Management Location	Size (bits)
Symmetric Encryption	AES Secret Key	hObject	128/192/256
	AES Round Key	Within context mcCtx->mcKeyCtx	Size of ARIA_KEY
Message Authentication key	HMAC Secret Key	hObject mcCtx->mcHashCtx	256/384/512 or more SHA2_CTX structure
	GMAC Secret Key	hObject mcCtx->mcKeyCtx	Same as block encryption key

Service	CSP	Management Location	Size (bits)
Public Key Encryption	Private Key for Decryption	hObject mcCtx->mcKeyCtx	Size of ASN.1 encoded RSA structure
	Internal Random Number for Encryption	Local variables within the cryptographic functions	256
Digital Signature	Private Key for Generation	hObject mcCtx->mcKeyCtx	Size of ASN.1 encoded RSA/ECDSA structure
	Internal Random Number for Digital Signature	Local variable within a digital signature function	Depending on an algorithm
Random number generator	Noise Sources Collected Externally	Noise source collection provided as variable from API	Depending on a noise source
	Internal State Variable V,C	Global variables within the cryptographic module	V:440 C:440
KDF	KBKDF CTR Derived Master Key	hObject mcCtx->mcKeyCtx	Depending on User Input

2.2 Public Keys

Table 6 – Public Keys

Service	PSP	Management Location	Size (bits)
Operation Mode	IV / CTR / Nonce	Within context mcCtx->mcParam	128
Public Key Encryption	Public Key for Encryption	hObject mcCtx->mcKeyCtx	Size of ASN.1 encoded RSA structure
Digital Signature	Public Key for Verification	hObject mcCtx->mcKeyCtx	Size of ASN.1 encoded RSA/ECDSA structure

2.3 Keys Information

All keys are generated using the FIPS mode random number generator. Internally used random numbers (ex: RSA-PSS algorithm) are not stored in MC_CONTEXT within the cryptographic module. The module implements a Hash_DRBG, which accepts input from entropy sources that are external to the cryptographic boundary, for seed material. External entropy can be added by the calling application via the API. The Cryptographic Module's calling application shall use entropy sources that meet the security strength required for the random bit generator Hash_DRBG (SHA-256). A minimum of 256 bits of entropy must be provided by the calling application. No assurance of the minimum strength of generated keys. The DRBG is seeded with the following:

- Initialize: SHA256DRBG_Init with Entropy input (32 bytes), nonce (16 bytes), and personalize String (0 bytes)
- Reseed: SHA256DRBG_Reseed with Entropy input (32 bytes) and additional data (0 bytes)

The CSPs that are in the session context of this cryptographic module are created directly by a request of an application program (ex: MC_GenerateKeyPair) or replaced by data entered by a user in plain text (ex: MC_CreateObject).

When an application successfully establishes a session, then calls a service that returns the context's CSP, the CSP is returned plain text form (MC_GetObjectValue).

There are no CSPs permanently stored in the module. The masking key used to verify the integrity of the module is stored, but this is not applicable to the CSP.

The CSPs that are in the session or OBJECT are managed by an application program and provide a zeroizing service function (MC_DestroyObject, MC_CloseSession) so that they can all be zeroized when the cryptographic service ends.

3 Roles, Authentication and Services

3.1 Roles and Services

MagicCryptoMVP encryption module is a software library. It supports only a single crypto officer or user (single-user mode of operation).

Roles can be separated into the crypto officer and user roles, and each have access to the supported services.

The role of the crypto officer includes self-test, cryptographic module initialization, cryptographic module shutdown, encryption/decryption, digital signature, hash, MAC, and cryptographic services such as random number generation.

The MagicCryptoMVP is a Security Level 1 cryptographic module, and it does not have a maintenance interface.

The detailed service items provided by the cryptographic module are shown in the following table.

Table 7 – Services and CSP

Service	Description	Keys & CSPs	Role	Access Keys
Basic Service	Install/Remove cryptographic module	-	CO, U	-
	Self-test	-	CO, U	
	Initialize/Finalize cryptogaphic module	-	CO, U	
	Show status	-	CO, U	
	Create session	-	CO, U	
	Close session	-	CO, U	

Service	Description	Keys & CSPs	Role	Access Keys
Hash	SHA	-	CO, U	-
MAC	HMAC, GMAC	MAC Key	CO, U	Generate/Execute
Symmetric Encryption	AES	AES symmetric Keys	CO, U	Generate/Execute
Asymmetric Encryption	RSA-ES	RSA asymmetric PublicKey RSA asymmetric PrivateKey	CO, U	Generate/Execute
Digital Signature	RSA ECDSA	PrivateKey for Sign generate PublicKey for Sign Verification	CO, U	Generate/Execute
Random	Hash-DRBG		CO, U	Generate/Execute
Generate KeyPair	RSA EC	RSA PublicKey and PrivateKey EC PublicKey and PrivateKey	CO, U	Generate/Execute
KDF	KBKDF_CTR	Master Key	CO, U	Generate/Execute

3.2 Authentication

The MagicCryptoMVP encryption module is Security Level 1 and does not provide an authentication mechanism for users and crypto officer.

4 Self-Tests

The cryptographic module performs both power-up tests and conditional tests, and the details are as follows.

Table 8 – Self-Test

Self-Test	Test	Description
Power-up Test	Cryptographic Algorithm test	Hash Algorithm KAT
		HMAC Algorithm KAT
		Random number generation Algorithm KAT
		Symmetric Key Algorithm KAT
		Asymmetric Key Algorithm test
		Digital signature Algorithm test
		Derived key KAT
	Software integrity test	
Conditional test	Pair-wise consistency test	
	SP800-90A Health Tests (Instantiate, Generate, and Reseed)	

If an error occurs in the self-test, the cryptographic module can no longer be used in the system. MC_Selftest outputs status MC_OK on success and MC_FAIL on error.

Table 9 – Self-Test Error Status

Self-Test Result	Return Code
Success	MC_OK
Fail	MC_FAIL

4.1 Power-up Test

The cryptographic module performs the test before it goes into an operating state when the power is applied to the module without any user intervention. When the power-up test is completed, the result of the test is output as the return value of the API function, which is the status output interface of MC_Initialize.

Table 10 – Auto Call Function

OS	Auto Call Function	Description
Linux	__attribute__((constructor)) _MC_Initialize()	Constructor autoload function

4.1.1 Cryptographic Algorithm Test

To test each cryptographic algorithm used in the cryptographic module, KAT is performed to see if a correct answer is output for a given input.

Cryptographic algorithm tests are performed on encryption, decryption, message authentication, message integrity, and random number generators.

Pair-wise consistency tests are performed when the output has more than one value for a given input, such as the RSA digital signature algorithm. The test verifies the generated digital signature and the KAT confirms the result of signature verification.

Table 11 – Algorithm Test

Cryptographic Algorithm	Test Method
AES-128-ECB	Encryption/Decryption KAT Test
AES-192-ECB	Encryption/Decryption KAT Test
AES-256-ECB	Encryption/Decryption KAT Test
AES-128-CBC	Encryption/Decryption KAT Test
AES-192-CBC	Encryption/Decryption KAT Test
AES-256-CBC	Encryption/Decryption KAT Test
AES-128-CTR	Encryption/Decryption KAT Test
AES-192-CTR	Encryption/Decryption KAT Test
AES-256-CTR	Encryption/Decryption KAT Test
AES-128-GCM	Encryption/Decryption/Verification KAT Test
AES-192-GCM	Encryption/Decryption/Verification KAT Test
AES-256-GCM	Encryption/Decryption/Verification KAT Test
AES-128-CCM	Encryption/Decryption/Verification KAT Test
AES-192-CCM	Encryption/Decryption/Verification KAT Test
AES-256-CCM	Encryption/Decryption/Verification KAT Test
HMAC-SHA-256	CreateMAC/VerifyMAC KAT Test
HMAC-SHA-384	CreateMAC/VerifyMAC KAT Test
HMAC-SHA-512	CreateMAC/VerifyMAC KAT Test
GMAC-AES-128	CreateMAC/VerifyMAC KAT Test
GMAC-AES-192	CreateMAC/VerifyMAC KAT Test
GMAC-AES-256	CreateMAC/VerifyMAC KAT Test
Hash-DRBG SHA-256	SHA256 Hash DRBG Algorithm (No PR) KAT Test
SHA-224	Hash KAT Test
SHA-256	Hash KAT Test
SHA-384	Hash KAT Test

SHA-512	Hash KAT Test
RSAES-2048	PublicKey Encryption / PrivateKey Decryption Test
RSAES-3072	PublicKey Encryption / PrivateKey Decryption Test
RSA-SHA-256-2048	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
RSA-SHA-256-3072	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
RSA-SHA-384-2048	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
RSA-SHA-384-3072	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
RSA-SHA-512-2048	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
RSA-SHA-512-3072	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-256 P-224	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-256 P-256	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-256 P-384	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-256 P-521	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-256 B-283	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-256 K-283	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-384 P-224	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-384 P-256	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-384 P-384	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-384 P-521	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-384 B-283	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA-384 K-283	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA512-P-224	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA512-P-256	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA512-P-384	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA512-P-521	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA512-B-283	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
ECDSA-SHA512-K-283	Generation Signature by PrivateKey / Verification Signature by PublicKey Test
KDF-CTR	Derived Key KAT Test

4.1.2 Software/Firmware Integrity Test

Software integrity test for the MagicCryptoMVP cryptographic module is performed during the power-up self-test included in the initialization process. When the module calls the initialization function, MC_Initialize, the integrity test for the installed module is performed by calculating a HMAC (SHA-256) integrity verification value for the shared library itself.

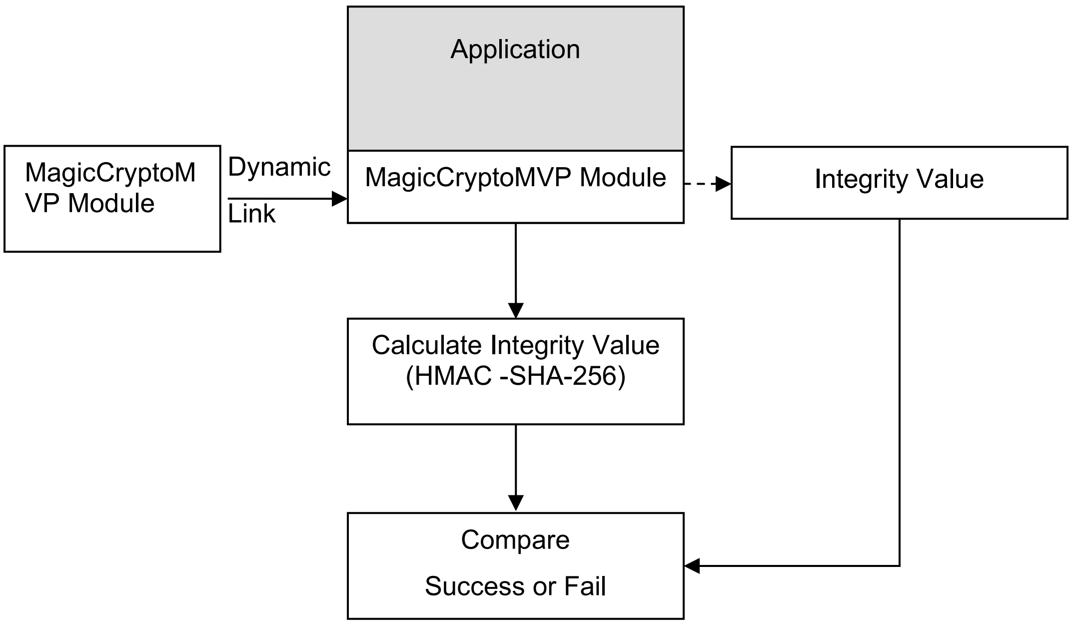

Figure 2 – Integrity Test

4.1.3 Critical Function Test

The cryptographic service APIs provided by the module can also be critical functions. The critical function tests are performed in power-up test and conditional test. The following functions are tested.

Table 12 – Critical Function Tests

Critical Function	Description
MC_SignInit	Initialize digital signature generation
MC_Sign	Generate digital signature
MC_VerifyInit	Initialize digital signature verification
MC_Verify	Verify digital signature
MC_EncryptInit	Initialize message encryption
MC_Encrypt	Encrypt message
MC_DecryptInit	Initialize message decryption
MC_Decrypt	Decrypt message
MC_DigestInit	Initialize message digest
MC_Digest	Generate message digest
MC_CreateMacInit	Initialize MAC
MC_CreateMac	Generate MAC value
MC_VerifyMacInit	Initialize verify MAC

MC_VerifyMac	Verify MAC value
MC_DeriveKey	Derive key (KBKDF CTR)

4.2 Conditional Test

Conditional tests are automatically performed by the module when keypairs and random numbers are generated.

Table 13 – Conditional Tests

Algorithm	Test Method
RSA	Pair wise consistency Test
ECDSA	Pair wise consistency Test
Hash-DRBG	Continuous Test
Hash-DRBG	SP800-90A Health Tests (Instantiate, Generate, and Reseed)

4.2.1 Pair-Wise Consistency Test

The module generates public and private key pairs using MC_GenerateKeyPair function to generate and verify digital signatures. When the private key and public key pair are generated, a pair-wise consistency test is performed to check that the key pair is generated successfully.

The pair-wise consistency test verifies that the public key and the private key pair match by generate signature and verify signature.

5 Operational Environment

The MagicCryptoMVP cryptographic module is a dynamic library that runs on Linux operating systems.

The MagicCryptoMVP cryptographic module is a software shared library that works in conjunction with an application program when the application program is running.

Linux operating systems run running processes in a virtual memory area to logically separate them from other processes, and the cryptographic module operates only in the area where the process is loaded. Therefore, intermediate values for critical security parameters and key generation cannot access other processes in the operating system.

The module operates in a modifiable operational environment per FIPS 140-2 level 1 specifications.

5.1 Tested Environment

The Module was tested on the following environment:

Table 14 – Tested Operating Environment

Operating System/Platform	Hardware Platform	Processor	Module bits
Linux 4.19 x86_64 64bits	MacBookPro	Intel Core i5	64bits

5.2 Vendor Affirmed Environment

The following platforms have not been tested as part of the FIPS 140-2 level 1 validation. However, DreamSecurity “vendor affirms” that these platforms are equivalent to the tested and validated platforms. Additionally, DreamSecurity affirms that the module will function the same way and provide the same security services on any of the systems listed below.

Table 15 – Vendor Affirmed Operating Environment

Platform	Version	Kernel bits	Processor	Module bits
Linux	2.6	64bits	X86_64 processor	64bits
Linux	3.2	64bits	X86_64 processor	64bits
Linux	3.4	64bits	X86_64 processor	64bits
Linux	3.10	64bits	X86_64 processor	64bits
Linux	3.11	64bits	X86_64 processor	64bits
Linux	3.13	64bits	X86_64 processor	64bits
Linux	3.16	64bits	X86_64 processor	64bits
Linux	3.18	64bits	X86_64 processor	64bits
Linux	3.19	64bits	X86_64 processor	64bits
Linux	4.1	64bits	X86_64 processor	64bits
Linux	4.4	64bits	X86_64 processor	64bits
Linux	4.9	64bits	X86_64 processor	64bits
Linux	4.10	64bits	X86_64 processor	64bits
Linux	4.13	64bits	X86_64 processor	64bits
Linux	4.14	64bits	X86_64 processor	64bits
Linux	4.19	64bits	X86_64 processor	64bits
Linux	5.4	64bits	X86_64 processor	64bits

Note: The Cryptographic Module will operate correctly on other GPC platforms running Linux.

Note: CMVP makes no statement as to the correct operation of the module when so ported if the specific operational environment is not listed on the validation certificate.

6 Mitigation of Other Attacks Policy

6.1 Prevention of Timing Attacks on RSA

The RSA algorithm is vulnerable to timing attacks. As a counter measure the RSA Fixed Window technique was applied.

6.2 Prevention of ECC Side Channel Attack

Side channel attacks are known vulnerability for the ECC Point Multiplication algorithm. To prevent the side channel attack, the Fixed Based Comb algorithm and Montgomery's ladder algorithm were applied.

The ECC P256R1/B283R1/B283K1 Curve was complemented using the Fixed Based Comb algorithm, and the ECC P224R1 Curve was complemented using the Montgomery's ladder algorithm.

The Montgomery's ladder algorithm (Montgomery ladder with (X, Y) -only co-Z addition) used in the cryptographic module is as follows.

```
Input: P ∈ E(F_q), k = (k_n−1, . . . , k_1, k_0)_2 ∈ N with k_n−1 = 1
Output: Q = [k]P
1. (R_1, R_0) ← XYCZ-IDBL(P)
2. for i = n − 2 downto 1 do
3. b ← k_i
4. (R_1−b, R_b) ← XYCZ-ADDC(R_b, R_1−b)
5. (R_b, R_1−b) ← XYCZ-ADD(R_1−b, R_b)
6. end for
7. b ← k_0
8. (R_1−b, R_b) ← XYCZ-ADDC(R_b, R_1−b)
9. λ ← FinalInvZ(R_0, R_1, P , b)
10. (R_b, R_1−b) ← XYCZ-ADD(R_1−b, R_b)
11. return (X_0*λ^2, Y_0*λ^3)
```

Figure 3 – Montgomery's Ladder Algorithm

7 Security Rules and Guidance

This section documents the security rules for the secure operation of the cryptographic module to implement the security requirements of FIPS 140-2.

1. The Cryptographic Module provides two distinct operator roles. Roles can be separated into crypto officer and users, and each have access to the same supported services.
2. The Cryptographic Module does not provide any operator authentication. The crypto officer and user roles are implicitly assumed.
3. The Cryptographic Module allows the operator to initiate power-up self-tests by power cycling power or MC_SelfTest function.
4. Power up self-tests do not require any operator action.
5. The Cryptographic Module is available to perform services only after successfully completing the Power-Up Self-Tests.
6. Data output are inhibited during key generation, self-tests, zeroization, and error states.
7. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
8. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
9. The Cryptographic Module does not support concurrent operators.
10. The Cryptographic Module does not support a maintenance interface or role.
11. The Cryptographic Module does not support manual key entry.
12. The Cryptographic Module does not have any proprietary external input/output devices used for entry/output of data.
13. The Cryptographic Module does not enter or output plaintext CSPs.
14. The Cryptographic Module does not store any plaintext CSPs
15. The Cryptographic Module does not output intermediate key values.
16. Per IG A.5, the GCM IV is generated according to Scenario 2, where it is generated internally using the DRBG. The IV length must be 96 bits or greater.

The module is installed by cryptographic officer by copying the library file (.so) to any directory within the operating environment. The operating system running the MagicCryptoMVP module must be configured in a single-user mode of operation. The directory path is then recorded and used by the calling application to access the module.

8 References and Definitions

The following standards are referred to in this Security Policy.

Table 16 – References

Abbreviation	Full Specification Name
[FIPS140-2]	*Security Requirements for Cryptographic Modules*, May 25, 2001
[IG]	*Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*
[108]	*NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions (Revised), October 2009*
[131A]	*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, March 2019*
[132]	*NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, Part 1: Storage Applications, December 2010*
[133]	*NIST Special Publication 800-133, Recommendation for Cryptographic Key Generation, June 2020*
[135]	*National Institute of Standards and Technology, Recommendation for Existing Application-Specific Key Derivation Functions, Special Publication 800-135rev1, December 2011.*
[186]	*National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-4, July, 2013.*
[197]	*National Institute of Standards and Technology, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, November 26, 2001*
[198]	*National Institute of Standards and Technology, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198-1, July, 2008*
[180]	*National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standards Publication 180-4, August, 2015*
[202]	*FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, August 2015*
[38A]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation, Methods and Techniques, Special Publication 800-38A, December 2001*
[38B]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication 800-38B, May 2005*

Abbreviation	Full Specification Name
[38C]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, Special Publication 800-38C, May 2004*
[38D]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, Special Publication 800-38D, November 2007*
[38E]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, Special Publication 800-38E, January 2010*
[38F]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, Special Publication 800-38F, December 2012*
[56Br2]	*NIST Special Publication 800-56B Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, March 2019*
[90A]	*National Institute of Standards and Technology, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90A, June 2015.*
[90B]	*National Institute of Standards and Technology, Recommendation for the Entropy Sources Used for Random Bit Generation, Special Publication 800-90B, January 2018.*