WatchDox® CryptoModule

WatchDox® CryptoModule
Security Policy
Version 1.1

November 5, 2013

Copyright Notice

Acknowledgments

WatchDox, Inc. serves as the "vendor" for this validation. Project management coordination for this effort was provided by:

Adi Ruppin	+1 800 209 1688
WatchDox, Inc.	adi@watchdox.com
299 S California Ave.,	
Palo Alto, CA 94306	
USA	

Validation testing was performed by SAIC. For information on software validation or revalidation requirements and testing details, contact:

William Tung	+1-443-367-7389
SAIC	WILLIAM.TUNG@saic.com
6841 BENJAMIN FRANKLIN DR.	
COLUMBIA, MD 21046	
USA	

Modification History

2013-07-01 Initial revision

References

Reference	***Full Specification Name***
Services	*[ANS X9.31] Digital Signatures Using Reversible Public Key Cryptography for the Financial Industry (rDSA)*
[FIPS 140-2]	*Security Requirements for Cryptographic modules, May 25, 2001*
[FIPS 180-3]	*Secure Hash Standard*
[FIPS 186-3]	*Digital Signature Standard*
[FIPS 197]	*Advanced Encryption Standard*
[FIPS 198-1]	*The Keyed-Hash Message Authentication Code (HMAC)*
[SP 800-38B]	*Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
[SP 800-38C]	*Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*
[SP 800-38D]	*Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
[SP 800-56A]	*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*

Reference	***Full Specification Name***
[SP 800-67R1]	*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*
[SP 800-89]	*Recommendation for Obtaining Assurances for Digital Signature Applications*
[SP 800-90]	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
[SP 800-131A]	*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*

TABLE OF CONTENTS

1. Introduction

This document is the non-proprietary security policy for the WatchDox CryptoModule
FIPS Object Module, hereafter referred to as the Module.

The Module is a software library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality. The Module is classified by FIPS
140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the general purpose computer on which the module is installed. The logical cryptographic boundary of the Module is the fipscanister object module, a single object module file named *fipscanister.o* (Linux®[1]/Unix®[2]), fipcanister.framework (iOS), fipcanister.so (Android) or *fipscanister.lib* (Microsoft Windows®[4]). The Module performs no communications other than with the calling application (the process that invokes the Module services).

The FIPS 140-2 security levels for the Module are as follows:

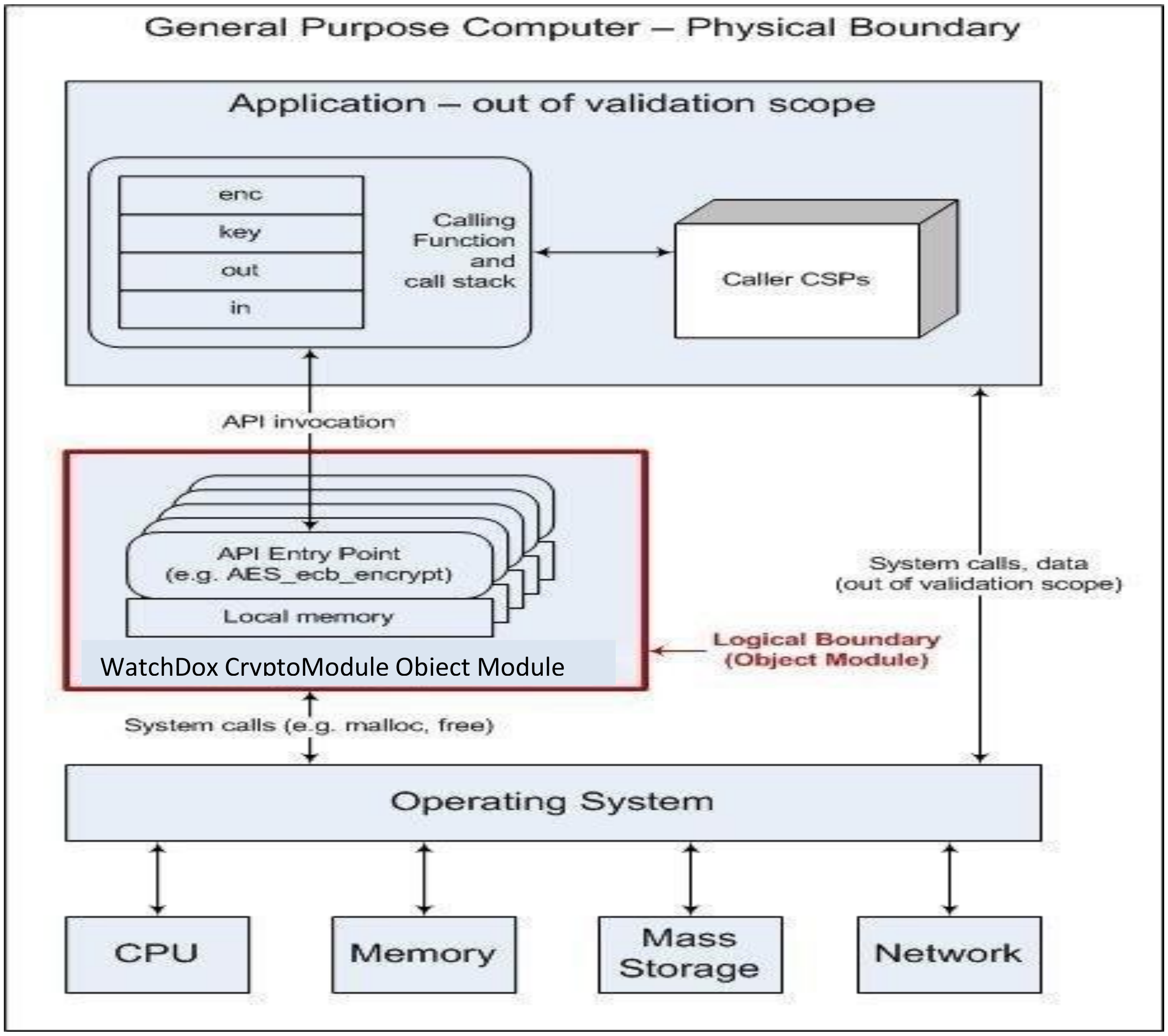
Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	2
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

Table 1 – Security Level of Security Requirements

1 Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.
2 UNIX is a registered trademark of The Open Group
4 Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

The Module's software version for this validation is 1.0.

Figure 1 - Module Block Diagram


General Purpose Computer – Physical Boundary
Application – out of validation scope
enc
key
out
in
Calling Function and call stack
Caller CSPs
API invocation
API Entry Point (e.g. AES_ecb_encrypt)
Local memory
WatchDox CryptoModule Object Module
Logical Boundary (Object Module)
System calls, data (out of validation scope)
System calls (e.g. malloc, free)
Operating System
CPU
Memory
Mass Storage
Network

2. Tested Configurations

#	Operational Environment	Processor	Optimizations (Target)	EC	B
1	Red Hat Enterprise Linux 6	Intel Xeon family	Without AES-NI	P	U2
2	Windows 7 32-bit	Intel Core family (x86)	With AES-NI	P	W2
3	Apple iOS 6.1	ARMv7	With NEON	P	U2
4	Android 4.1	ARM Cortex A9	With NEON	P	U2

Table 2 - Tested Configurations (B = Build Method; EC = Elliptic Curve Support). The EC column indicates support for prime curve only (P), or all NIST defined B, K, and P curves (BKP).

3. Ports and Interfaces

The physical ports of the Module are the same as the computer system on which it is executing. The logical interface is a C-language application program interface (API).

Logical interface type	Description
Control input	*API entry point and corresponding stack parameters*
Data input	*API entry point data input stack parameters*
Status output	*API entry point return values and status stack parameters*
Data output	*API entry point data output stack parameters*

Table 3 - Logical interfaces

As a software module, control of the physical ports is outside module scope. However, when the module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The module is single-threaded and in error scenarios returns only an error value (no data output is returned).

4. Modes of Operation and Cryptographic Functionality

The Module supports only a FIPS 140-2 Approved mode. Tables 4a and 4b list the Approved and Non-approved but Allowed algorithms, respectively.

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[ANS X9.31] RNG	AES 128/192/256	#1239
Encryption, Decryption and CMAC	[FIPS 197] AES	128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify	#2623
	[SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS		
Message Digests	[FIPS 180-3]	SHA-1, SHA-2 (224, 256, 384, 512)	#2199
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	#1621
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5, SigVerPSS (1024/1536/2048/3072/4096 with all SHA sizes)	#1340
	[FIPS 186-2] ECDSA	Key Pair, PKV, SigGen, SigVer (all NIST defined P curves with SHA-1 only)	#451
	[FIPS 186-3] ECDSA	Key Pair, PKV, SigGen, SigVer (all NIST defined P curves with all SHA sizes)	#451

Table 4a – FIPS Approved Cryptographic Functions

The Module supports only NIST defined curves for use with ECDSA and ECC CDH. The Module supports two operational environment configurations for elliptic curve; NIST prime curve only (listed in Table 2 with the EC column marked "P") and all NIST defined curves (listed in Table 2 with the EC column marked "BKP").

Category	Algorithm	Description
Key Agreement	EC DH	Non-compliant (untested) DH scheme using elliptic curve, supporting all NIST defined B, K and P curves. Key agreement is a service provided for calling process use, but is not used to establish keys into the

		Module.
Key Encryption, Decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services.

Table 4b – Non-FIPS Approved But Allowed Cryptographic Functions

Category	Algorithm	Description
Random Number Generation; Symmetric key generation Key Encryption, Decryption	[SP 800-90] DRBG[5] Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function Dual EC DRBG: P-256, P-384, P-521
Encryption, Decryption and CMAC	[SP 800-67] Triple-DES	3-Key Triple-DES TECB, TCBC, TCFB, TOFB; CMAC generate and verify
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] DSA	PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024 with SHA-1 only)
	[FIPS 186-3] DSA	PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024/2048/3072 with all SHA sizes)
	[FIPS 186-2] ECDSA	Key Pair, PKV, SigGen, SigVer (all NIST defined B and K curves with SHA-1 only)
	[FIPS 186-3] ECDSA	Key Pair, PKV, SigGen, SigVer (all NIST defined B & K curves with all SHA sizes)
ECC CDH (KAS)	[SP 800-56A] (5.7.1.2)	All NIST defined B and K curves
		All NIST defined P curves

Table 4c – Non-FIPS Approved Non-Compliant Cryptographic Functions

5 For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90] and [SP800-57].

EC DH Key Agreement provides a maximum of 256 bits of security strength. RSA Key Wrapping provides a maximum of 256 bits of security strength.

The Module supports only a FIPS 140-2 Approved mode. The Module requires an initialization sequence (see IG 9.5): the calling application invokes FIPS_mode_set()6, which returns a “1” for success and “0” for failure. If FIPS_mode_set() fails then all cryptographic services fail from then on. The application can test to see if FIPS mode has been successfully performed.

The Module is a cryptographic engine library, which can be used only in conjunction with additional software. Aside from the use of the NIST defined elliptic curves as trusted third party domain parameters, all other FIPS 186-3 assurances are outside the scope of the Module, and are the responsibility of the calling process.

4.1 Critical Security Parameters and Public Keys

All CSPs used by the Module are described in this section. All access to these CSPs by Module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

CSP Name	Description
RSA SGK	RSA (1024 to 16384 bits) signature generation key
RSA KDK	RSA (1024 to 16384 bits) key decryption (private key transport) key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves) signature generation key
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
HMAC Key	Keyed hash key (160/224/256/384/512)
RNG CSPs	Seed (128 bit), AES 128/192/256 seed key and associated state variables for ANS X9.31 AES based RNG[7]
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

Table 4.1a – Critical Security Parameters

6 The function call in the Module is FIPS_module_mode_set() which is typically used by an application via the FIPS_mode_set() wrapper function.

7 There is an explicit test for equality of the seed and seed key inputs

Critical Security Parameters and Public Keys

All CSPs used by the Module are described in this section. All access to these CSPs by Module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

CSP Name	Description
RSA SGK	RSA (1024 to 16384 bits) signature generation key
RSA KDK	RSA (1024 to 16384 bits) key decryption (private key transport) key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves) signature generation key
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
HMAC Key	Keyed hash key (160/224/256/384/512)
RNG CSPs	Seed (128 bit), AES 128/192/256 seed key and associated state variables for ANS X9.31 AES based RNG[7]
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

Table 4.1a – Critical Security Parameters

6 The function call in the Module is FIPS_module_mode_set() which is typically used by an application via the FIPS_mode_set() wrapper function.
7 There is an explicit test for equality of the seed and seed key inputs

Authentication data is loaded into the module during the module build process, performed by an authorized operator (Crypto Officer), and otherwise cannot be accessed.

The module does not output intermediate key generation values.

CSP Name	Description
RSA SVK	RSA (1024 to 16384 bits) signature verification public key
RSA KEK	RSA (1024 to 16384 bits) key encryption (public key transport) key
ECDSA SVK	ECDSA (All NIST defined B, K and P curves) signature verification key

Table 4.1b – Public Keys

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The Module stores RNG and DRBG state values for the lifetime of the RNG or DRBG instance. The module uses CSPs passed in by the calling application on the stack. The Module does not store any CSP persistently (beyond the lifetime of an API call), with the exception of RNG state values used for the Modules' default key generation service.

Generation: The Module implements ANSI X9.31 compliant RNG services for creation of symmetric keys, and for generation of elliptic curve and RSA keys as shown in Table 4a. The calling application is responsible for storage of
generated keys returned by the module.

Entry: All CSPs enter the Module’s logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.
Output: The Module does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.

Destruction: Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

Private and secret keys as well as seeds and entropy input are provided to the Module by the calling application, and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently. An authorized application as user (Crypto-Officer and User) has access to all key data generated during the operation of the Module.

In the event Module power is lost and restored the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism: 128 bits for the [ANS X9.31] RNG mechanism, and as shown in [SP 800-90] Table 2 (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG) and Table 4 (Dual_EC_DRBG). This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

5. Roles, Authentication and Services

The Module implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators. The User or Crypto Officer role is assumed by passing the appropriate password to the FIPS_module_mode_set() function. The password values may be specified at build time and must have a minimum length of 16 characters. Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the Module unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

Authentication data is loaded into the Module during the Module build process, performed by the Crypto Officer, and otherwise cannot be accessed.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

Both roles have access to all of the services provided by the Module.

User Role (User): Loading the Module and calling any of the API functions.

Crypto Officer Role (CO): Installation of the Module on the host computer system and calling of any API functions.

All services implemented by the Module are listed below, along with a description of service CSP access.

Service	Role	Description
Initialize	User, CO	Module initialization. Does not access CSPs.
Self-test	User, CO	Perform self tests (FIPS_selftest). Does not access CSPs.
Show status	User, CO	Functions that provide module status information: Version (as unsigned long or const char *) FIPS Mode (Boolean) Does not access CSPs.
Zeroize	User, CO	Functions that destroy CSPs: fips_rand_prng_reset: destroys RNG CSPs. fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs.) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.

Service	Role	Description
Random number generation	User, CO	Used for random number and symmetric key generation. Seed or reseed an RNG instance Determine security strength of an RNG instance Obtain random data Uses and updates RNG CSPs.
Asymmetric key generation	User, CO	Used to generate DSA, ECDSA and RSA keys: RSA SGK, RSA SVK; ECDSA SGK, ECDSA SVK There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, CMAC (passed in by the calling process).
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. Executes using HMAC Key (passed in by the calling process).
Key transport[8]	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). Executes using RSA KDK, RSA KEK (passed in by he calling process).
Digital Signature	User, CO	Used to generate or verify RSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs.

Table 5 - Services and CSP Access

8 "Key transport" can refer to a) moving keys in and out of the module or b) the use of keys by an external application. The latter definition is the one that applies to the WatchDox CryptoModule FIPS Object Module.

6. Self-test

The Module performs the self-tests listed below on invocation of Initialize or Self-test.

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 Per IG 9.3, this testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128 bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256 bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256)
AES CMAC	KAT	Sign and verify CBC mode, 128, 192, 256 key lengths
RSA	KAT	Sign and verify using 2048 bit key, SHA-256, PKCS#1
ECDSA	PCT	Keygen, sign, verify using P-224, K-233 and SHA512. The K-233 self-test is not performed for operational environments that support prime curve only (see Table 2).
X9.31 RNG	KAT	128, 192, 256 bit AES keys

Table 6a - Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test)

The Module is installed using one of the set of instructions in Appendix A, as appropriate for the target system. The HMAC-SHA-1 of the Module distribution file as tested by the CMT Laboratory and listed in Appendix A is verified during installation of the Module file as described in Appendix A.

The FIPS_mode_set()[9] function performs all power-up self-tests listed above with no operator intervention required, returning a “1” if all power-up self-tests succeed, and a “0” otherwise. If any component of the power-up self-test fails an internal flag is set to prevent subsequent invocation of any cryptographic function calls. The module will only enter the FIPS Approved mode if the module is reloaded and the call to FIPS_mode_set()[9] succeeds.

9 FIPS_mode_set() calls Module function FIPS_module_mode_set()

The power-up self-tests may also be performed on-demand by calling FIPS_selftest(), which returns a “1” for success and “0” for failure. Interpretation of this return code is the responsibility of the calling application.

The Module also implements the following conditional tests:

Algorithm	**Test**
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair
ANSI X9.31 RNG	Continuous test for stuck fault

Table 6b - Conditional Tests

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

The Module supports two operational environment configurations for elliptic curve: NIST prime curves only (listed in Table 2 with the EC column marked "P") and all NIST defined curves (listed in Table 2 with the EC column marked "BKP").

7. Operational Environment

The tested operating systems segregate user processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software
and hardware. The Module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

8. Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.