WE
SECURE
THE
FUTURE

Non-proprietary Security Policy

Check Point CryptoCore version 4.0

FIPS 140-2

Level 1 Validation

Document Version 4.09

May 2019

Table of Contents

Introduction

Purpose

This non-proprietary Cryptographic Module Security Policy for the Check Point CryptoCore 4.0, describes how the Check Point CryptoCore meets the Level 1 security requirements of FIPS 140-2. Validation testing was performed on Windows 10 and Apple macOS 10.12. This policy document is part of FIPS 140-2 validation of the Check Point CryptoCore 4.0.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 Security Requirements for Cryptographic Modules) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the NIST website at http://csrc.nist.gov/groups/STM/cmvp/index.html.

FIPS 140-2 Validation Scope

Security requirements section	Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles and Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	3
Self-Tests	1
Design Assurance	1
Mitigation of other attacks	N/A

Table 1 FIPS 140-2 Security Requirements

References

This document deals only with operations and capabilities of the Check Point CryptoCore 4.0 in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the Check Point CryptoCore 4.0 application from the following source:

Refer to: http://www.checkpoint.com for information on Check Point products and services as well as answers to technical or sales related questions.

Additional external references:

[1] Intel® Advanced Encryption Standard (AES) New Instructions Set, Rev 3.01, Intel Architecture Group.
[2] Intel® Digital Random Number Generator Generator(DRNG) Software Software Implementation Guide.
[3] Recommendation for Password-Based Key Derivation, NIST Special Publication 800-132

Acronym list

Acronym	**Definition**
Triple-DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
MD5	Message Digest Algorithm 5
RSA	Rivest, Shamir, Adleman Private/Public key algorithm
SHA	Secure Hashing Algorithm
PRNG	Pseudo Random Number Generator
UEFI	Unified Extensible Firmware Interface
DRBG	Deterministic Random Bit Generator
DRNG	Digital Random Number Generator

Table 2 Acronyms

Check Point CryptoCore 4.0

Overview

The Check Point CryptoCore 4.0 (hereinafter referenced as the crypto module) provides cryptographic support for the Check Point line of products. The crypto module is used to perform cryptographic operations as well as create, manage and delete cryptographic keys.

The cryptographic services provided by the crypto module includes symmetric and asymmetric key based encryption algorithms, message digest, message authentication code, RSA encryption, signature generation and verification, and pseudo random number generation functions.

The crypto module can be used to provide multiple security functions in Check Point applications. A structured set of APIs can be called to perform these functions. The API set makes the module very flexible, and enables adding crypto functions to new applications without changing the module itself.

Utilizing the crypto module, Check Point applications can create encryption keys, which can then be used to encrypt data. The APIs provide the ability to encrypt both static data (such as hard disk blocks) as well as data streams (such as browser traffic). The crypto module also provides the ability to perform cryptographic MAC operations and Message Digest operations.

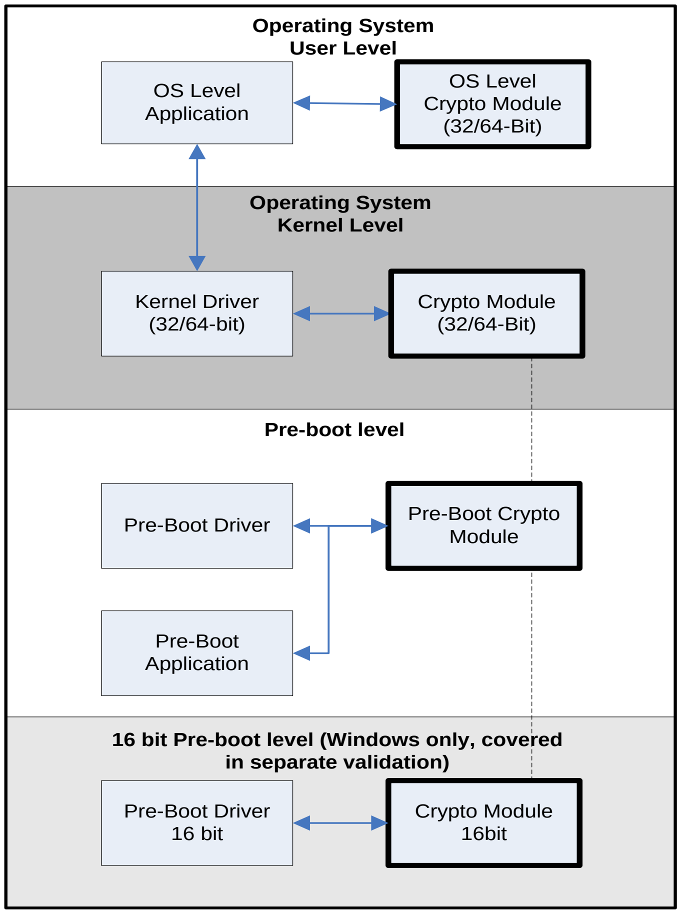

Figure 1 Interaction of Crypto module in different system modes.

Cryptographic Module

The Check Point CryptoCore 4.0 is classified as a multi-chip standalone module for FIPS 140-2 purposes. The module was tested for FIPS validation on a GPC running Microsoft Windows 10, Apple macOS Sierra 10.12, both with and without the Intel AES-NI Processor Algorithm Accelerator (PAA). The Windows 10 module is tested compiled with both Visual Studio 2008 and Visual Studio 2017. For exact details on tested platforms refer to the algorithm certificates listed in Table 6. Compliance is maintained for all of the above-mentioned operating system platforms on which the binary executable is unchanged. In addition to the validated platforms, the module has been affirmed by the vendor to be operational on the platforms listed in Table 3, according to FIPS 140-2 Implementation Guidance G.5. The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys if the specific operational environment is not listed on the validation certificate (see Table 6).

Operating System	Processor/Platform

Microsoft Windows 7	x86 32/64 –bit both with and without AES-NI
Microsoft Windows 8	x86 32/64-bit both with and without AES-NI
Microsoft Windows 8.1	x86 32/64-bit both with and without AES-NI
Apple Mac OSX 10.10	x86 64-bit kernel 32/64 –bit user mode both with and without AES-NI
Apple Mac OSX 10.11	x86 64-bit kernel 32/64 –bit user mode both with and without AES-NI

Table 3 Vendor affirmed platforms

The Cryptographic Module is packaged in the form of 32-bit dll/shared lib, used by all 32-bit user mode components in the system, 32-bit kernel export driver/kext used by kernel mode components, a 64-bit dll/shared library used by 64-bit OS modes, 64-bit kernel mode export driver/kext used by kernel mode in 64-bit OS and a 64-bit UEFI binary used by the EFI/UEFI Pre-boot environment. See also: Module Ports and Interfaces.

The relationship between the different modes is shown in Figure 1 (above). The 16-bit mode provides symmetric key cryptographic functions during 16 bit pre-boot operation while the other modes provide crypto functions thereafter. Note that the 16-bit module is validated as a separate module and only mentioned herein for the sake of completeness.

AES-NI Support

The 4.0 version of the module supports AES acceleration through the AES-NI instruction set [1]. All versions of the module running on Windows, and Apple macOS support the AES-NI acceleration. During initialization of the module it will detect if the CPU supports AES-NI and, if it is present, the module engages the AES-NI acceleration.

Module Ports and Interfaces

The Check Point CryptoCore 4.0 is classified as a multi-chip standalone module for FIPS 140-2 purposes. As such, the module’s cryptographic boundary includes the following:

- Microsoft Windows binaries: cryptocore.dll, ccore32.sys, ccore64.sys
- Apple macOS, CkpCryptocore.kext, cryptocore.dylib

A PC or mobile device running an operating system and interfacing with the computer, keyboard, mouse screen, floppy drive, CD-ROM drive, speaker, serial ports, parallel ports, and power plug.

The Check Point CryptoCore 4.0 provides a logical interface via an Application Programming Interface (API). The API provided by the module is mapped to the FIPS 140-2 logical interfaces: data input, data output, control input, and status output. All of these physical interfaces are separated into the logical interfaces from FIPS as described in the following table:

FIPS 140-2 Logical Interface	Module Mapping
Data Input Interface	**Parameters passed to the module via the API call**
Data Output Interface	**Data returned by the module via the API call**
Control Input Interface	**Control input through the API function calls**
Status Output Interface	**Information returned via exceptions and calls**
Power Interface	**Does not provide a separate power or maintenance access interface beyond the power interface provided by the computer itself**

Table 4 FIPS 140-2 Logical Interfaces

Roles, Services and Authentication

The cryptographic module provides Crypto Officer and User roles. All the services exported by the module are common to both the roles except key zeroization. Only the Crypto-officer is allowed to perform key zeroization. Since the module is validated at security level 1, it does not provide an authentication mechanism.

Exported Services	Supported	Exported to
cryptInitSystem	X	User/CO
cryptCipherInit	X	User/CO
cryptCipherDestroy	X	CO
cryptCipherSetParams	X	User/CO
cryptCipherSetKey	X	User/CO
cryptCipherSetIV	X	User/CO
cryptCipherGetIV	X	User/CO
cryptEncrypt	X	User/CO
cryptDecrypt	X	User/CO
cryptDigestInit	X	User/CO
cryptDigestDestroy	X	CO

cryptDigestCopy	X	User/CO
cryptDigestUpdate	X	User/CO
cryptDigestFinal	X	User/CO
cryptHmacInit	X	User/CO
cryptHmacDestroy	X	CO
cryptHmacCopy	X	User/CO
cryptHmacUpdate	X	User/CO
cryptHmacFinal	X	User/CO
cryptPrngInitEx	X	User/CO
cryptPrngDestroy	X	CO
cryptPrngAddEntropy	X	User/CO
cryptPrngReadBytesEx	X	User/CO
cryptPkInit	X	User/CO
cryptPkDestroy	X	CO
cryptPkSetKey	X	User/CO
cryptPkGetKey	X	User/CO
cryptPkGenKey	X	User/CO
cryptPkSign	X	User/CO
cryptPkVerify	X	User/CO
cryptPkEncrypt	X	User/CO
cryptPkDecrypt	X	User/CO
cryptGetFunctionList	X	User/CO
cryptXTSEncrypt	X	User/CO
cryptXTSDecrypt	X	User/CO
cryptDeriveKey	X	User/CO
cryptAesKeyWrap	X	User/CO
cryptAesKeyUnwrap	X	User/CO
cryptGetStatusInfo	X	User/CO
cryptEnableAesCpuAcceleration	X	User/CO
cryptRdRandReadBytes	X	User/CO
cryptRdSeedReadBytes	X	User/CO

Table 5 Exported Functions

Physical Security

Since the Check Point Crypto Module is implemented solely in software, the physical security section of FIPS 140-2 is not applicable.

Operational Environment

The Cryptographic module’s software components are designed to be installed on the targets listed below as indicated in section 2.2 above.

Microsoft Windows

The Cryptographic module’s software components are designed to be installed on an IBM-compatible PC running 64-bit versions of Microsoft Windows 10.

Apple macOS

The Cryptographic module’s software components are designed to be installed on an Apple Mac computer running 64-bit versions of macOS Sierra 10.12.

UEFI/EFI

The Cryptographic module’s software components are designed to be used on an IBM-compatible PC or Apple computer running 64-bit EFI or UEFI pre-boot environment. An operating system is not required for the UEFI/EFI module.

Each software components of the module will implement an approved message authentication code, used to verify the integrity of software component during the power-up self-test (see section on self-test below). While loaded in the memory, the respective target OS will protect all unauthorized access to the Cryptographic module’s address memory and process space.

Cryptographic Key Management

The Check Point CryptoCore 4.0 module implements algorithms according to the following table.

The FIPS approved column specifies whether the algorithm is available in the FIPS-mode (non-approved algorithms are not to be used, see Operation of the Check Point CryptoCore 4.0 for more information). The XTS-AES mode is only approved for storage applications. XTS-AES key management meets the requirements of IG A.9.

Algorithm Type	Algorithm, Modes and Key length	Supported	FIPS Approved	Algorithm certificate #
Symmetric Key	AES - ECB, CBC, XTS – 128, 256	X	Yes	4112
	DES - ECB, CBC – 64	X	No	
	Triple-DES – ECB, CBC – 192 (168)	X	Yes	2247
	Blowfish ECB, CBC - 56 – 448	X	No	
	CAST-128, 256	X	No	
Message Digest	MD5 (128)	X	No	
	SHA-1 (160)	X	Yes	3385
	SHA-2 (224, 256, 384, 512)	X	Yes	3385
	SHA-3 (224, 256, 384, 512)	X	Yes	7
HMAC	HMAC-SHA-1 (160)	X	Yes	2687
	HMAC-SHA-2 (224, 256, 384, 512)	X	Yes	2687
	HMAC-SHA-3 (224, 256, 384, 512)	X	Yes	2687
Asymmetric Key	RSA (less than 2048 bits) key wrapping	X	No	
	RSA (less than 2048 bits) PKCS#1 sign	X	No	
	RSA (less than 1024 bits) PKCS#1 verify	X	No	
	RSA (2048, 3072, 4096) key wrapping	X	No, but allowed in FIPS mode	
	RSA (2048, 3072, 4096) PKCS#1 sign	X	Yes	2225
	RSA (1024, 1536, 2048, 3072, 4096) PKCS#1 verify	X	Yes	2225
	RSA Key generation (less than 2048 bits)	X	No	
	RSA Key generation (2048, 3072, 4096)	X	Yes	2225
Random number generation	SP800-90A DRBG	X	Yes	1238
Key derivation	PKCS#5	X	No, but allowed in	

			FIPS mode	
Key Wrap	NIST AES Key Wrap	X	Yes	4112
Entropy generation	Intel RdRand/RdSeed	X	No, but allowed in FIPS mode	
MAC	Triple-DES MAC	X	Yes	

Table 6 Algorithms list

The module implements vendor affirmed algorithms listed in **Table 7**.

Algorithm	Description	IG Ref.
KDF Password Based	SP 800-132 [3] Options: 1a Functions: HMAC-based KDF using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Vendor Affirmed IG D.6

Table 7 Approved Cryptographic Functionality with Vendor Affirmation.

The approved PBKDF must only be used for storage applications to conform to SP 800-132 [3]. Any other use of the PBKDF is non-compliant. The module is a general purpose software module and the PBKDF function, cryptDeriveKey(), uses parameterized input. The size of the generated key is also parameterized and a choice of the user of the module, a minimum of 128-bits is recommended. The salt should be generated with the approved DRBG, appropriately seeded, and the size of the salt should be chosen so that it matches the output size of the generated key. A minimum salt size of 128-bits is then implied and recommended. The iteration count should be as large as possible without hampering the usability of the application in which the module is used. Password input data should also reflect the size of the key being generated. The module does not implement password policy enforcement. Using US-ASCII printable characters a minimum password length 20 characters is required for 128-bit entropy. For further information regarding parameter choice to the PBKDF users are referred to Appendix A of SP 800-132 [3].

The following table provides a list of keys and key sizes that can be generated and/or used with the module. Keys are generated or inserted, where inserted means data is provided as plaintext or cipher text input to the data input interface of the service (API) whereas keys are inserted in plaintext. Insertion is done as specified in the API listing. See Table 8 for details of how the critical security components (CSP) are inserted into, or generated by, the module.

Key Name	Created	Size(s) in bits	Purpose
AES_key	Inserted	128, 192, 256,	Encryption, Decryption For XTS mode only 128 and 256 bits size.
Triple-DES_key	Inserted	192 (168)	Encryption, Decryption
RSA_Private_key	Inserted	1024, 1536, 2048, 3072, 4096 mod size	Key transport Decryption and Signing. 1024 and 1536 not approved for use in

			FIPS mode.
RSA_Public_key	Inserted	1024, 1536, 2048, 3072, 4096 mod size	Key transport Encryption and Verification,
HMAC_SHA1_key	Inserted	160	HMAC creation
HMAC_SHA224_key	Inserted	224	HMAC creation
HMAC_SHA256_key	Inserted	256	HMAC creation
HMAC_SHA384_key	Inserted	384	HMAC creation
HMAC_SHA512_key	Inserted	512	HMAC creation
HMAC_SHA3_224_key	Inserted	224	HMAC creation
HMAC_SHA3_256_key	Inserted	256	HMAC creation
HMAC_SHA3_384_key	Inserted	384	HMAC creation
HMAC_SHA3_512_key	Inserted	512	HMAC creation
Triple-DES_MAC_MIT_key	Hard-coded	192 (168)	Module Integrity Testing
DRBG key (AES Key)	Inserted	256	CTR DRBG AES Key
DRBG seed	Inserted	Dynamic	CTR DRBG Seed data
DRBG internal state (V value)	Generated	128	CTR DRBG Internal state

Table 8 List of Keys/CSPs

Type	Algorithms	Service	CSP	Inserted/ Generated	Access Type
Initialization	Triple-DES MAC	cryptInitSystem	Triple-DES_MAC_MIT_Key	N/A	N/A
Symmetric Cipher					
	AES, Triple-DES (CAST, Blowfish DES only available in non-approved mode)	cryptCipherInit	Secret Key	Inserted	Read
		cryptCipherDestroy	Secret Key	Inserted	Write
		cryptCipherSetParams	Secret Key	Inserted	Read
		cryptCipherSetKey	Secret Key	Inserted	Read
		cryptCipherSetIV	N/A	N/A	N/A
		cryptCipherGetIV	N/A	N/A	N/A
		cryptEncrypt	Secret Key	Inserted	Read
		cryptDecrypt	Secret Key	Inserted	Read
		cryptXTSEncrypt	Secret Key	Inserted	Read
		cryptXTSDecrypt	Secret Key	Inserted	Read
Message Digest					
	SHA-1, SHA-2, SHA-3 (MD5 only available in non-approved mode)	cryptDigestInit	N/A	N/A	N/A
		cryptDigestDestroy	N/A	N/A	N/A
		cryptDigestCopy	N/A	N/A	N/A
		cryptDigestUpdate	N/A	N/A	N/A
		cryptDigestFinal	N/A	N/A	N/A
Message Authentication					
	HMAC with SHA-1, SHA-2, SHA-3 (MD5 only available in non-approved mode)	cryptHmacInit	Secret Key	Inserted	Read
		cryptHmacDestroy	Secret Key	Inserted	Write
		cryptHmacCopy	Secret Key	Inserted	Read/Write
		cryptHmacUpdate	Secret Key	Inserted	Read

		cryptHmacFinal	Secret Key	Inserted	Read/Write
Deterministic Random number generator					
	SP800-90A CTR	cryptPrngInitEx	Internal State/Entropy Input/Nonce	Generated	Read/Write
		cryptPrngDestroy	Internal State	Inserted	Write
		cryptPrngAddEntropy	Internal State/Entropy Input	Generated	Read/Write
		cryptPrngReadBytesEx	Internal State/Entropy Input	Generated	Read/Write
Asymmetric Encryption					
	RSA	cryptPkInit	N/A	N/A	N/A
		cryptPkDestroy	Private Key	Inserted	Write
		cryptPkSetKey	Private Key	Inserted	Read
		cryptPkGetKey	Private Key	Inserted	Read
		cryptPkGenKey	Private Key	Generated	Write
		cryptPkSign	Private Key	Inserted	Read
		cryptPkVerify	N/A	N/A	N/A
		cryptPkEncrypt	N/A	N/A	N/A
		cryptPkDecrypt	Private Key	Inserted	Read
NIST AES key wrap		cryptAesKeyWrap	Secret Key	Inserted	Read
NIST AES key unwrap		cryptAesKeyUnwrap	Secret Key	Inserted	Read
PKCS#5 key derivation (SP 800-132)	HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512	cryptDeriveKey	Secret Key	Generated	Write
Non-approved, but allowed services					
Retrieve data from Intel Secure Key Technology	Hardware	cryptRdRandReadBytes	N/A	N/A	N/A
	Hardware	cryptRdSeedReadBytes	N/A	N/A	N/A
Non-security-relevant services					
Retrieve function pointers		cryptGetFunctionList	N/A	N/A	N/A
Get module info		cryptGetStatusInfo	N/A	N/A	N/A
Disable/Enable AES-NI		cryptEnableAesCpuAcceleration	N/A	N/A	N/A

Table 9 Key/CSP Access

When keys are set for deletion, the key is zeroized by overwriting the keys to ensure it cannot be retrieved. Zeroization is done by calling the crypt*Destroy() set of services. Sensitive intermediate data is zeroized by the module itself.

The cryptRdRandReadBytes() and cryptRdSeedReadBytes functions are implemented as calls to the Intel RDRAND/RDSEED hardware CPU instructions. RDRAND/RDSEED entropy is generated from thermal noise and uses an internal entropy size of 256 bits [2].

Self-Tests

The Check Point CryptoCore 4.0 performs several power-up self-tests including known answer tests for the FIPS Approved algorithms listed in the table below.

The crypto module also performs a self-test integrity check using TRIPLE-DES-MAC with a fixed key to verify the integrity of the module.

Algorithm	Power-up self-test type	Conditional self test
AES	Encrypt/Decrypt KAT	N/A
Triple-DES	Encrypt/Decrypt KAT	N/A
SHA-1	KAT	N/A
SHA-224	KAT	N/A
SHA-256	KAT	N/A
SHA-384	KAT	N/A
SHA-512	KAT	N/A
HMAC-SHA-1	KAT	N/A
HMAC-SHA-256	KAT	N/A
HMAC-SHA-384	KAT	N/A
HMAC-SHA-512	KAT	N/A
RSA	2048 Sign/Verify KAT using SHA-2 256	Yes
DRBG	KAT	Yes
SHA3-224	KAT	N/A
SHA3-256	KAT	N/A
SHA3-384	KAT	N/A
SHA3-512	KAT	N/A
AES-KEY-WRAP	KAT (Wrap/Unwrap)	N/A
AES-KEY-UNWRAP FAIL	KAT (Unwrap with failure expected)	N/A
HMAC-SHA3-224	KAT	N/A
HMAC-SHA3-256	KAT	N/A
HMAC-SHA3-384	KAT	N/A
HMAC-SHA3-512	KAT	N/A
RDRAND	N/A	Yes

RDSEED	N/A	Yes

Table 10 List of Self tests

The module performs the following conditional tests: Continuous tests on the DRBG each time it is used to generate random data. DRBG health tests according to SP800-90A. A pair-wise consistency test each time the module generates RSA key pairs. Continuous tests on the output from the cryptRdRandReadBytes()/cryptRdSeedReadBytes() functions.

Design Assurance

Check Point maintains versioning for all source code and associated documentation through GIT versioning handling system.

Mitigation of Other Attacks

The Check Point CryptoCore 4.0 does not employ security mechanisms to mitigate specific attacks.

Operation of the Check Point CryptoCore 4.0

The Check Point CryptoCore 4.0 contains both FIPS-approved and non-FIPS-approved algorithms. In FIPS mode only Approved algorithms must be used.

To exemplify what we mean by FIPS mode vs. non-FIPS mode we provide the following example:

If Triple-DES is being used to encrypt plaintext data, then the module is operating in FIPS-mode, but if the Blowfish algorithm was being used, it would not be in FIPS-mode.

While RSA encryption and decryption is not an approved FIPS algorithm it may be used in a FIPS approved mode as part of a key transport mechanism; however, when transporting keys, the operator must use an RSA key pair with a minimum modulus size of 2048-bits to comply with CMVP requirements.

RSA signing and key wrapping using keys of sizes 1024 and 1536 is not approved in FIPS mode.

RSA (key wrapping; key establishment methodology provides between 112 and 150 bits of encryption strength; non-compliant less than 112-bits of encryption strength).

The Check Point CryptoCore 4.0 is designed for installation and use on a computer configured in single user mode, and is not designed for use on systems where multiple, concurrent users are active.

AES (Cert. #4112, key wrapping; key establishment methodology provides between 128 and 256 bits of encryption strength).