Barracuda Cryptographic Software Module

Version 1.0.1.8

FIPS 140-2
Non-Proprietary Security Policy

Level 1 Validation

Document Version 2.0

Prepared By:

Revision History

Version	Modification Date	Modified By	Description of Changes
1.0	2014-09-12	ICSA Labs	Initial Document
1.1	2015-02-19	Barracuda Networks	Incorporating comments from BAH
1.2	2015-02-19	ICSA Labs	Updated block diagram
1.3	2015-02-19	ICSA Labs	Formatting correction after updating block diagram
1.4	2015-02-19	ICSA Labs	Updated block diagram Updated Table of Contents to include section 7.5 (added in v1.1)
1.5	2015-03-30	Barracuda Networks	Updated RSA Certificate number
1.6	2015-04-16	Barracuda Networks	Updated section “Cryptographic Key Management” (section 7.2)
1.7	2015-09-21	Barracuda Networks	Incorporating comments from CMVP
1.8	2016-08-11	ICSA Labs	Added in Dell Power Edge OEs
1.9	2016-11-16	ICSA Labs	Updated CAVS certification numbers
2.0	2016-11-22	ICSA Labs	Corrected listing of CAVS certification numbers

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the Barracuda Cryptographic Software Module from Barracuda Inc.. It provides detailed information relating to the Federal Information Processing Standard (FIPS) 140-2 security requirements for conformance to security Level 1, and instructions on how to run the module in a secure FIPS 140-2 approved mode.

2 Cryptographic Module Specification

The Barracuda Cryptographic Software Module is a cryptographic software library that provides fundamental cryptographic functions for applications in Barracuda security products that use Barracuda OS v2.3.4 and require FIPS 140-2 approved cryptographic functions. The FIPS 140-2 validation of the Barracuda Cryptographic Software Module is comprised of the *fips_crypto_module.o* file.

2.1 Module Overview

The Barracuda Cryptographic Software Module is a software-based cryptographic module.Table 2-1 provides a list of platforms, operational systems and processors on which the Barracuda Cryptographic Software Module was tested.

Hardware Test Platforms	Operating System	Processor	Processor Optimization
BNHW002	*Barracuda OS v2.3.4*	*Intel Xeon*	*None*
BNHW008	*Barracuda OS v2.3.4*	*Intel Xeon*	*AES-NI*
BNHW003	*Barracuda OS v2.3.4*	*AMD Opteron*	*None*
BNHW003	*Barracuda OS v2.3.4*	*AMD Opteron*	*AES-NI*
Dell PowerEdge R320	*Barracuda NextGen Firewall and Control Center OS 7 under Microsoft Windows 2012 (64-bit) Hyper-V*	*Intel Xeon*	*None*
Dell PowerEdge R320	*Barracuda NextGen Firewall and Control Center OS 7 under Microsoft Windows 2012 (64-bit) Hyper-V*	*Intel Xeon*	*AES-NI*

Table 2-1: Tested Configurations

The logical cryptographic boundary of the module is the Barracuda Cryptographic Software Module dynamic library (fips_crypto_module.o). It is contained in the physical boundary of the general purpose computer (GPC) on which the module resides.

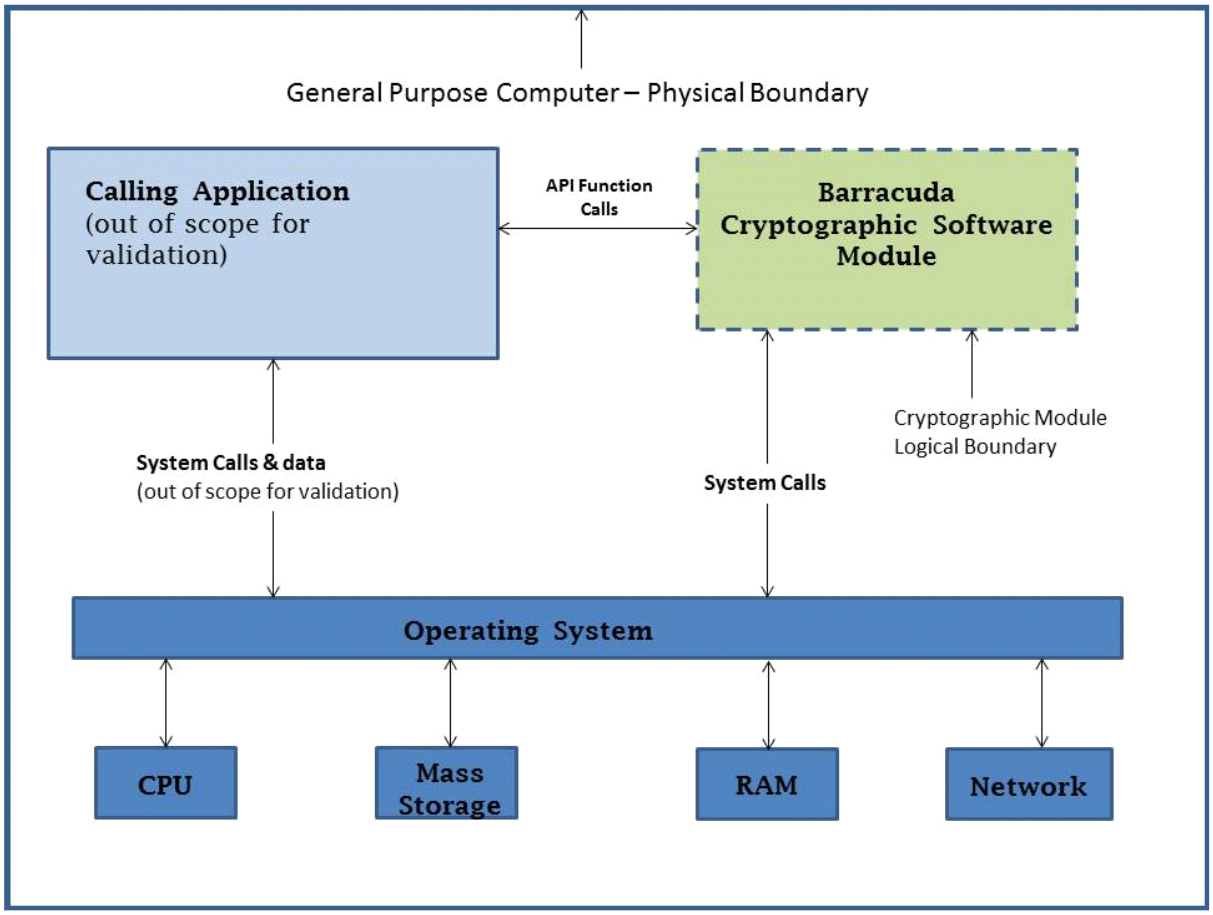

Figure 2-1 describes the GPC physical boundary, the Barracuda Cryptographic Software Module logical boundary, and their relationship.

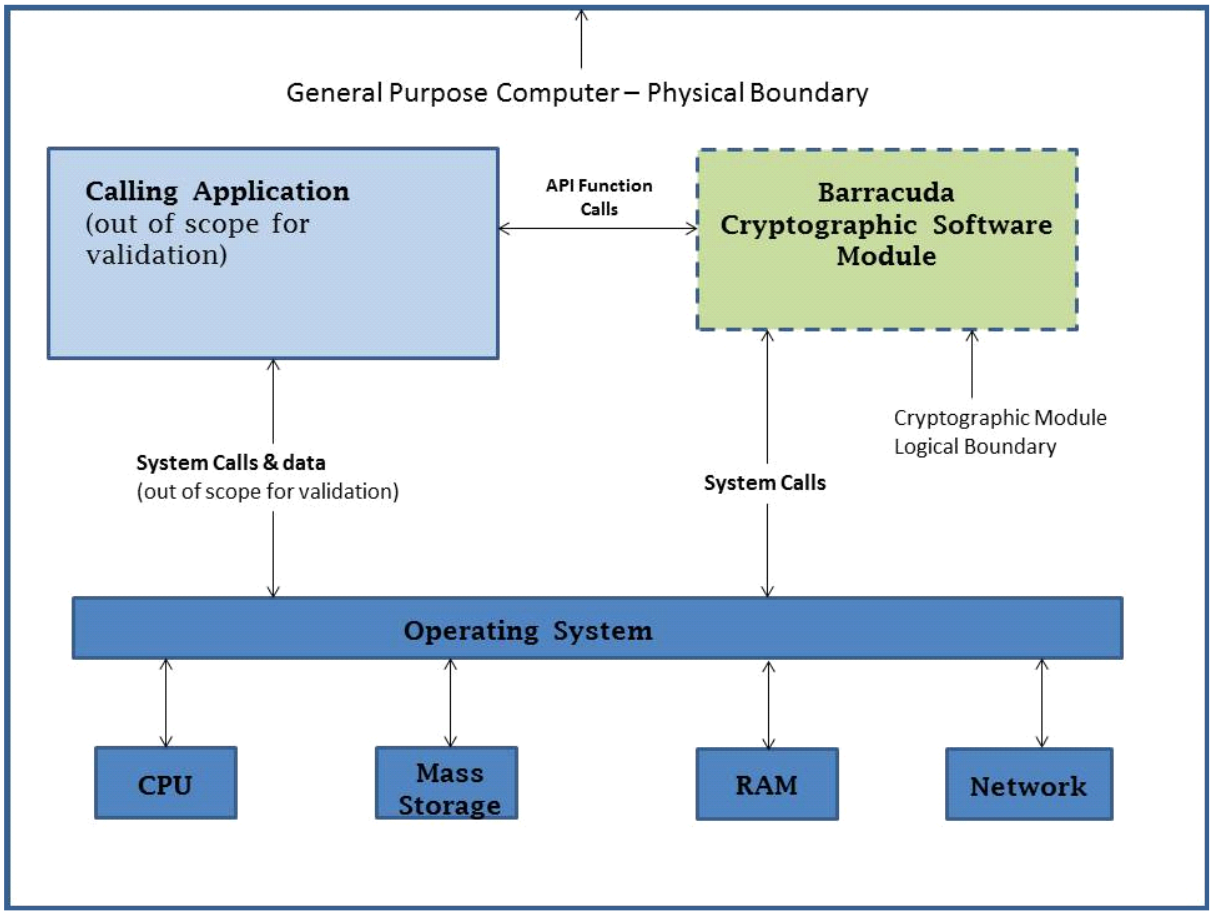

Figure 2-1: Logical Block Diagram

2.2 Security Levels

Per FIPS 140-2 terminology, the Barracuda Cryptographic Software Module is a multi-chip standalone module that meets overall level 1 FIPS 140-2 requirements. Table 2-2 lists the validation levels for each section of the Barracuda Cryptographic Software Module:

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1

Section	Section Title	Level
3	Roles, Services, and Authentication	2
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-tests	1
10	Design Assurance	3
11	Mitigation of Other Attacks	N/A

Table 2-2: Security Level per FIPS 140-2

2.3 Modes of Operation

The Barracuda Cryptographic Software Module has only a FIPS Approved mode of operation. The Barracuda Cryptographic Software Module must be initialized with the FIPS_module_mode_on function. The Barracuda Cryptographic Software Module will then operate in a FIPS approved mode of operation. Once initialized, the Barracuda Cryptographic Software Module supports the FIPS Approved Algorithms listed in Table 2-3:

Algorithm	Modes	CAVS Cert
AES-128/192/256	ECB, CBC, CFB1, CFB8, CFB128, OFB, CTR, CCM, CMAC, GCM, XTS	3165 4144
ECC CDH Component	• P-224/256/384/521 • K-233//283/409/571 • B-233/283/409/571	414 948
DRBG	• Hash • HMAC • CTR	651 1258

Algorithm	Modes	CAVS Cert
DSA (FIPS 186-4)	• PQG Generate ○ (2048, 224): SHA-224/256/384/512 ○ (2048, 256): SHA-256/384/512 ○ (3072, 256): SHA-256/384/512 • PQG Verify ○ (1024, 160) : SHA-1/224/256/384/512 ○ (2048, 224): SHA-224/256/384/512 ○ (2048, 256): SHA-256/384/512 ○ (3072, 256): SHA-256/384/512 • Key Pair ○ (2048, 224) ○ (2048, 256) ○ (3072, 256) • Signature Generate ○ (2048, 224): SHA-224/256/384/512 ○ (2048, 256): SHA-224/256/384/512 ○ (3072, 256): SHA-224/256/384/512 • Signature Verify ○ (1024, 160): SHA-1/224/256/384/512 ○ (2048, 224): SHA-1/224/256/384/512 ○ (2048, 256): SHA-1/224/256/384/512 ○ (3072, 256): SHA-1/224/256/384/512	911 1125
ECDSA (FIPS 186-4)	• PKG Curves: ○ P-224/256/384/521 ○ K-233//283/409/571 ○ B-233/283/409/571 • PKV Curves: ○ (All P, K and B curves) • SigGen Curves with SHA-224/256/384/512: ○ P-224/256/384/521 ○ K-233//283/409/571 ○ B-233/283/409/571 • SigVer Curves with SHA-1/224/256/384/512: ○ P-224/256/384/521 ○ K-233//283/409/571 ○ B-233/283/409/571	576 953
HMAC	SHA-1/224/256/384/512	1993 2716
RSA (FIPS 186-4)	• RSASSA-PKCS1_V1_5: ○ SigGen: ○ Mod 2048/3072 SHA-224/256/384/512 ○ SigVer: ○ Mod 1024/1536/2048/3072/4096 SHA-1/224/256/384/512 • RSASSA-PSS: ○ SigGen: ○ Mod 2048/3072 SHA-224/256/384/512 ○ SigVer: ○ Mod 1024/1536/2048/3072/4096 SHA-1/224/256/384/512	1603, 1690 2259

Algorithm	Modes	CAVS Cert
SHA	• SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512	2618 3412
Triple-DES	• Encrypt: 3-Key: ECB/CBC/CFB1/CFB8/CFB64/OFB • Decrypt: 2-Key & 3-Key: ECB/CBC/CFB1/CFB8/CFB64/OFB • CMAC (Generation/Verification)	1803 2264

Table 2-3: FIPS Approved Algorithms

In addition to the FIPS Approved algorithms, the module also supports the non-approved but allowed EC Diffie-Hellman (Shared Secret Computation) primitive, and RSA Encrypt/Decrypt for key transport only (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength). The FIPS 186-4 complaint RSA key generation function is FIPS_rsa_generate_key_ex ().

The AES XTS mode is only to be used for storage applications. The Barracuda Cryptographic Software Module does not support concurrent operators.

3 Module Ports and Interfaces

The physical ports of the module include those of the GPC on which the module is executed, but are outside the scope of the FIPS 140-2 validation. The logical interface consists of a C language application program interface (API) through which consumers of the module's services may exact control, request status, or pass data in/out. The FIPS 140-2 interfaces are described in Table 3-1: FIPS 140-2 Logical Interfaces. The Barracuda Cryptographic Software Module API documentation includes all the inputs, outputs, control, and status parameters.

FIPS 140-2 Logical Interface	Implementation
Data Input	C-language API with stack and register input parameters
Data Output	C-language API with stack and register output parameters
Control Input	C-language API with stack and register control parameters
Status Output	C-language API with stack and register status parameters
Power Interface	N/A

Table 3-1: FIPS 140-2 Logical Interfaces

4 Roles, Services, and Authentication

The Barracuda Cryptographic Software Module operates only in FIPS Approved mode and supports operators in either a Crypto-Officer (CO) role or User role. To initialize the cryptographic functions and select an operational role, the consumer of the module supplies a pre-defined password identifying the desired role to the FIPS_module_mode_on() API. As the operator that uses the FIPS module is a software program/application, the pre-defined password of the required role may be set during the application compile time.

The crypto-officer password is 36 characters in length and the user password is 33 characters in length. The probability of a random successful authentication attempt is $2^{-(8*36)}$ for the crypto-officer and is $2^{-(8*33)}$ for the user. As the operator is a software application and it is expected to have the password at the application compiletime, f failure to provide a valid password is treated as a module level error and will result in the module entering an error state, which can be cleared only by terminating and restarting the offending application. The password is not entered manually, but passed as a parameter in an API call by the calling application. Hence, there will be only one attempt and it is required to treat invalid password as module level error. The module does not allow for multiple authentication attempts. Since the error state can be cleared by power cycling the module, it would be possible to make one authentication attempt per second and restart the module per attempt. Thus 60 attempts per minute could be made. However since the probability of guessing the password per attempt has probability 1 in 2^ (8*36), it is clear that 60* (1 in 2^ (8*36)) is much less than 1 in 100,000.

The module provides the services listed in Table 4-1. Both the CO and the User roles have full read/write/execute/zeroize access to all services.

Service	Standard	Roles	Description	CSPs & Public Keys	API
AES-128/192/256 Encrypt/Decrypt (Modes: CBC, CFB1, CFB128, CFB8, CTR, ECB, GCM, OFB) AES-128/256 Encrypt/Decrypt (Mode XTS)	FIPS 197 SP 800-38A SP 800-38D (GCM) SP 800-38E (XTS)	User/CO	Symmetric Encryption/ Decryption using the AES encryption Standard	AES Encrypt/Decrypt Key (all modes), Generate/Verify key (GCM)	FIPS_evp_aes_128_cbc() FIPS_evp_aes_128_cfb1() FIPS_evp_aes_128_cfb128() FIPS_evp_aes_128_cfb8() FIPS_evp_aes_128_ctr() FIPS_evp_aes_128_ecb() FIPS_evp_aes_128_gcm() FIPS_evp_aes_128_ofb() FIPS_evp_aes_128_xts() FIPS_evp_aes_192_cbc() FIPS_evp_aes_192_cfb1() FIPS_evp_aes_192_cfb128() FIPS_evp_aes_192_cfb8() FIPS_evp_aes_192_ctr() FIPS_evp_aes_192_ecb() FIPS_evp_aes_192_gcm() FIPS_evp_aes_192_ofb() FIPS_evp_aes_256_cbc() FIPS_evp_aes_256_cfb1() FIPS_evp_aes_256_cfb128() FIPS_evp_aes_256_cfb8() FIPS_evp_aes_256_ctr() FIPS_evp_aes_256_ecb() FIPS_evp_aes_256_gcm() FIPS_evp_aes_256_ofb() FIPS_evp_aes_256_xts()
Triple-DES Encrypt (Modes CBC, CFB1, CFB64, CFB8, ECB, OFB)	SP 800-67	User/CO	Symmetric Encryption using the Triple-DES encryption Standard	Triple-DES Keys Three-key: K1 != K2 != K3 != K1	FIPS_evp_des_ede3() FIPS_evp_des_ede3_cbc() FIPS_evp_des_ede3_cfb1() FIPS_evp_des_ede3_cfb64() FIPS_evp_des_ede3_cfb8() FIPS_evp_des_ede3_ecb() FIPS_evp_des_ede3_ofb()

Service	Standard	Roles	Description	CSPs & Public Keys	API
Triple-DES Decrypt (Modes CBC, CFB1, CFB64, CFB8, ECB, OFB)	SP 800-67	User/CO	Symmetric Decryption using the Triple-DES encryption Standard	Triple-DES Keys Three-key: K1 != K2 != K3 != K1 Two-Key: K1 != K2 != K3 = K1 (Legacy use only)	FIPS_evp_des_ede3() FIPS_evp_des_ede3_cbc() FIPS_evp_des_ede3_cfb1() FIPS_evp_des_ede3_cfb64() FIPS_evp_des_ede3_cfb8() FIPS_evp_des_ede3_ecb() FIPS_evp_des_ede3_ofb()
DSA Signature Verification	FIPS 186-4	User/CO	Verify a signed message using DSA	DSA Public signature verification key	FIPS_dsa_verify() FIPS_dsa_verify_ctx() FIPS_dsa_verify_digest()
DSA Generate Domain Parameters	FIPS 186-4	User/CO	L>=2048, N=256 with SHA256	public domain parameters	FIPS_dsa_generate_parameters_ex()
DSA-2048/3072 Generate Key Pair	FIPS 186-4	User/CO	Generate 2048 or 3072 bit DSA key pair	DSA Private/Public Keys	FIPS_dsa_generate_key()
DSA Sign	FIPS 186-4	User/CO	Sign a message using DSA	Private Key provided by calling application	FIPS_dsa_sign() FIPS_dsa_sign_ctx() FIPS_dsa_sign_digest()
RSA Signature Verification	FIPS 186-4	User/CO	Verify an RSA 1024, 2048 or 3072 bit RSA key signature. Based on PKCS#1 v1.5 or PSS	RSA Signature Verification Public Key	FIPS_rsa_verify() FIPS_rsa_verify_ctx() FIPS_rsa_verify_digest()
RSA Generate Key Pair	FIPS 186-4	User/CO	Generate 2048 or 3072 bit RSA key pair. Based on ANSI X9.31	RSA Private/Public Keys	FIPS_rsa_x931_generate_key_ex
RSA Private Key Encrypt	FIPS 186-4	User/CO	Used for digital signature	RSA Private Key	FIPS_rsa_private_encrypt()
RSA Public Key Decrypt	FIPS 186-4	User/CO	Used for digital signature verification	RSA Public Key	FIPS_rsa_public_decrypt()

Service	Standard	Roles	Description	CSPs & Public Keys	API
RSA Sign	FIPS 186-4	User/CO	Generate 2048, 3072 bit RSA signature. Based on PKCS#1 v1.5 or PSS	RSA Private Signature Generation Key	FIPS_rsa_sign() FIPS_rsa_sign_ctx() FIPS_rsa_sign_digest()
ECDSA Signature Verification	FIPS 186-4	User/CO	Verify message signature (uses all SHA sizes including SHA-1 for legacy use)	ECDSA Public Signature Verification Key	FIPS_ecdsa_verify() FIPS_ecdsa_verify_ctx()
Generate Shared Secret (ECC CDH Primitive)	SP 800-56A Section 5.7.1.2	User/CO	Generate Shared Secret (KAS component). Allows only NIST recommended B, K and P curves.	Shared Secret	ECDH_compute_key()
EC Generate Key Pair	FIPS 186-4	User/CO	Allows only NIST recommended B, K and P curves.	EC Private Key	EC_KEY_generate_key()
ECDSA Sign	FIPS 186-4	User/CO	Sign message	ECDSA Private Signature Generation Key	FIPS_ecdsa_sign() FIPS_ecdsa_sign_ctx()
SHA-1/224/256/384/512	FIPS 180-4	User/CO	Generate a hash value based on the Secure Hash Standard (SHS)	None	FIPS_digestinit() FIPS_digestupdate() FIPS_digestfinal() FIPS_evp_sha1 () FIPS_evp_sha224 () FIPS_evp_sha256 () FIPS_evp_sha384 () FIPS_evp_sha512 ()

Service	Standard	Roles	Description	CSPs & Public Keys	API
HMAC-SHA-1/224/256/384/512	FIPS 198-1	User/CO	Generate HMAC-SHA	HMAC Key	FIPS_hmac_init FIPS_hmac_init_ex FIPS_evp_sha1 () FIPS_evp_sha224 () FIPS_evp_sha256 () FIPS_evp_sha384 () FIPS_evp_sha512 ()
CMAC AES-128/192/256	SP 800-38B	User/CO	Generate CMAC with AES	AES Generate/Verify Key	FIPS_cmac_init () FIPS_cmac_update() FIPS_cmac_final () FIPS_evp_aes_128_cbc() FIPS_evp_aes_192_cbc() FIPS_evp_aes_256_cbc()
CMAC Triple-DES	SP 800-38B	User/CO	Generate CMAC with Triple-DES	Triple-DES Keys Three-key: K1 != K2 != K3 != K1	FIPS_cmac_init () FIPS_cmac_update() FIPS_cmac_final () FIPS_evp_des_ede3_cbc()
CCM AES-128/192/256	SP 800-38C	User/CO	Generate CCM with AES	AES Encrypt/Decrypt Key	FIPS_cipherinit() FIPS_cipher() EVP_aes_128_ccm EVP_aes_192_ccm EVP_aes_256_ccm
Reseed DRBG	SP 800-90A	User/CO	Reseed the DRBG from a NDRBG	V, Key, and entropy input for HMAC and CTR DRBG; V, C and entropy input for Hash DRBG	drbg_ctr_reseed() drbg_hash_reseed() drbg_hmac_reseed() FIPS_drbg_reseed() FIPS_drbg_set_reseed_interval()
Get security strength	SP800-57, Table 2	User/CO	Provides the security strength of the DRBG based on the strength of the underlying DRBG mechanism	None	FIPS_drbg_get_strength()

Service	Standard	Roles	Description	CSPs & Public Keys	API
Generate Random Bits; Generate Symmetric Key	SP 800-90A	User/CO	Generate Random Bits as defined in SP800-90A. Supported options: Hash DRBG, HMAC DRBG, no reseed, CTR DRBG (AES), no derivation function. Prediction Resistance supported for all options.	Returned Symmetric Key (depends on usage); V, Key, and entropy input for HMAC and CTR DRBG; V, C and entropy input for Hash DRBG.	FIPS_rand_bytes() FIPS_drbg_generate
Initialization & Operator Authorization		User/CO	Prepare the module for use in FIPS approved mode for the role associated with "password"	Pre-calculated HMAC-SHA-1's for CO and User role authentications	FIPS_module_mode_on(password)
Status / Version		User/CO	Retrieve the current status of the module or version information	None	FIPS_module_mode() FIPS_incore_fingerprint() FIPS_module_version() FIPS_module_version_text()
Zeroize		User/CO	Zeroize the CSP's of an algorithm. All symmetric and public key Encrypt/Decrypt algorithms are automatically zeroized when the associated context is released. The DRBG CSP's may be zeroized by uninstantiating the DRBG or via the fips_drbg_free function.	V, Key, and entropy input for HMAC and CTR DRBG; V, C and entropy input for Hash DRBG; Symmetric keys; Pubic /Private Keys	fips_drbg_uninstantiate() fips_drbg_free

Service	Standard	Roles	Description	CSPs & Public Keys	API
Self-Test		User/CO	Performs integrity test (using HMAC-SHA256) and algorithm self-tests. These are always peformed at power-on and may optionally be run on –demand.	None	FIPS_selftest() FIPS_selftest_sha1(); FIPS_selftest_aes_ccm(); FIPS_selftest_aes_gcm(); FIPS_selftest_aes_xts(); FIPS_selftest_aes(); FIPS_selftest_des(); FIPS_selftest_rsa(); FIPS_selftest_dsa(); FIPS_selftest_ecdsa(); FIPS_selftest_ecdh(); FIPS_drbg_stick(); FIPS_selftest_hmac(); FIPS_selftest_drbg(); FIPS_selftest_drbg_all(); FIPS_selftest_cmac(); FIPS_check_incore_fingerprint()

Table 4-1: FIPS Approved Services with Roles/CSPs

Service	Reference	Roles	Description	CSPs	API
RSA Public Key Encrypt / Private Key Decrypt	IG D.9	User/CO	Used to encrypt/decrypt key material for key transport	RSA Private Key, Wrapped Key	FIPS_rsa_private_decrypt() FIPS_rsa_public_encrypt()
EC Diffie-Hellman (Shared Secret Computation) Primitive	IG D.8, Scenario 6	User/CO	Calculate the shared secret. The ECDH_compute_key() function is same as listed in Table 4-1. But this entry is for non-Approved (non complaint with SP 800-56A) primitive only.	Calculated Shared Secret	ECDH_compute_key()

Table 4-2: Non-FIPS Approved but Allowed Cryptographic Functions

5 Physical Security

The physical security requirements do not apply to the Barracuda Cryptographic Software Module because the module is a FIPS 140-2 Level 1 software module and the physical security is provided by the host platform.

6 Operational Environment

The module operates on a General Purpose Computer (GPC) which is a modifiable operating system. The module was tested on the platforms defined in Table 2-1.

The operating systems on the platforms tested segregate each process into a separate process space that is logically separated from all other processes. The module only allows for single user operation in that each module function is processed in the process space of the calling application (operator).

7 Cryptographic Key Management

7.1 Critical Security Parameters (CSPs)

Table 7-1 contains a list of keys/CSPs used in the module. Sections 7.2-7.4 describe the generation, entry, storage, output and zeroization of the keys/CSPs used in the module.

CSP	Description
AES EDK, CMAC, GCM, XTS	AES Encrypt/Decrypt Key (all modes), Generate/Verify key (CMAC, GCM)
Triple-DES Symmetric Keys	Triple-DES Keys Three-key: K1 != K2 != K3 != K1 Two-Key: K1 != K2 != K3 = K1 (Legacy use only) CMAC Generate/Verify Key
DSA Sign/Verify Keys	Public domain parameters DSA Private/Public Keys
RSA Sign/Verify, Encrypt/Decrypt Keys	RSA Private/Public Key
ECDSA Sign/Verify Keys	ECDSA Signature Keys
ECC CDH Shared Secret	Shared Secret used to derive keying material
EC Public/Private Keys	Elliptic Curve Private/Public keys
HMAC Key	Message Authentication Code Key
DRBG State	V, Key and entropy input for HMAC and CTR DRBG, V, C and entropy input for Hash DRBG
CO Auth Digest	Digest for Crypto Officer authentication
User Auth Digest	Digest for User authentication

Table 7-1: Module CSPs

7.2 Key Generation

The module supports generation of Elliptic Curve, RSA, DSA key pairs and symmetric keys using an approved SP800-90A DRBG. Table 4-1 identifies keys generated by the module.

Keys are generated from the output of an SP800-90 compliant random bit generator (DRBG). The entropy input provided to the DRBG originates in the NDRBG of the platform.

No assurance of the minimum strength of generated keys.

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.

IG D.8 Scenario 5 requires compliance with one or more of the key agreement primitives specified in SP 800-56A. Domain parameters and key sizes shall conform to SP 800-56A. A CVL algorithm validation certificate for a DLC primitive is required (See CVL cert. #414)

7.3 Key Entry, Storage, Output

No keys are persisted by the module beyond the lifetime of the API call, except the DRBG CSPs. All keys/keying material is entered into the module from the consuming application (i.e. "operator") as plaintext parameters in RAM to API functions. Keys/keying material originates within the physical boundary of the module and is not output outside the physical boundary.

7.4 Zeroization

Temporarily stored keys and keying material are zeroized automatically by the API functions when complete. CSPs related to random number functions (identified in Table 7-1) may be zeroized via explicit function calls. The operating system protects system memory and process space from access by unauthorized users.

CSPs, secret and private keys that are used by the API function are stored temporarily in RAM during the function process. The zeroization is performed by each API function, which calls the function OPENSSL_cleanse at the end of the process. The OPENSSL_cleanse function overwrites the memory space with pseudorandom values that are produced based on the address of the buffer that is being zeroized and an internal counter.

7.5 Entropy

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism. This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met

8 EMI/EMC

The module is a software module and was tested on standard GPC platforms that meet the applicable Federal Communication Commission (FCC) Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined in Subpart B of FCC Part 15.

9 Self-Tests

The Barracuda Cryptographic Software Module performs the required suite of self-tests upon initialization of the module. The self-tests are performed automatically without operator intervention. The following self-tests are performed:

Self Tests:

- Software integrity KAT: HMAC-SHA256
- SHA-1
- HMAC- SHA1 KAT
- HMAC- SHA224 KAT
- HMAC- SHA256 KAT
- HMAC- SHA384 KAT
- HMAC- SHA512 KAT
- AES KAT: ECB mode, Encrypt, 128-bit

- AES KAT: ECB mode, Decrypt, 128-bit
- AES CCM KAT: Encrypt, 192-bit
- AES CCM KAT: Decrypt, 192-bit
- AES GCM KAT: Encrypt, 256-bit
- AES GCM KAT: Decrypt, 256-bit
- XTS-AES KAT: Encrypt, 128,256
- XTS-AES KAT: Decrypt, 128,256
- AES CMAC KAT: CBC mode, sign, 128,192,256
- AES CMAC KAT: CBC mode, verify, 128,192,256
- Triple-DES KAT: ECB mode, Encrypt, 3-key
- Triple-DES KAT: ECB mode, Decrypt, 3-key
- Triple-DES CMAC KAT: CBC mode, generate, 3-key
- Triple-DES CMAC KAT: CBC mode, verify, 3-key
- RSA KAT: sign, 2048 bit, SHA256
- RSA KAT: verify, 2048 bit, SHA256
- DSA Pairwise Consistency: sign, 2048 bit, SHA384
- DSA Pairwise Consistency: verify, 2048 bit, SHA384
- DRBG SP800-90:
  - CTR_DRBG: AES 256-bit, with and without derivation function
  - HASH_DRBG: SHA256
  - HMAC_DRBG: SHA256
- ECDSA Pairwise Consistency: KeyGen, sign, P-224, K-233 and SHA512
- ECDSA Pairwise Consistency: KeyGen, verify, P-224, K-233 and SHA512
- ECC CDH KAT: Shared secret calculation per section 5.7.1.2 of SP800-56A, IG 9.6

The module also inplements the following conditional tests:

Conditional Self-test

- DRBG SP800-90 continuous test
- DSA: Pairwise Consistency test on each generation of a key pair
- RSA: Pairwise Consistency test on each generation of a key pair
- ECDSA: Pairwise Consistency test on each generation of a key pair
- NDRBG: continuous test

The module will enter an error state if any of the self-tests fail and an internal flag is set to prevent any subsequent requests for cryptographic functions. The module must be power cycled to remove it from the error state. Once power cycled the self-test will be run upon initialization. If all tests pass the module will move into an operational state. If any of the self-test fails the module will move back to the error state.

The self-tests can be performed on demand by the operator by invoking the FIPS_selftest() function.

10 Design Assurance

Barracuda uses Git for configuration management of source code and documentation. All module source code and documentation is maintained on a server that is internal to Barracuda. Git maintains a history of all changes made to documents and source code.

The Barracuda Cryptographic Software Module is for use inside of Barracuda products. The module is a binary object module and is only distributed to the Barracuda development team as the FIPS 140-2 validated *fips_crypto_module.o* binary object. The module code has a computed HMAC SHA-256 embedded in it for the software integrity test. If there are any changes to the module or the HMAC SHA-256 the software integrity test will

fail. The Barracuda development teams work in secure environments with controlled access. The module and the host application are installed on one of the operational environments listed in Table 2-1.

11 Mitigation of Other Attacks

This module was not designed to mitigate any specific attacks outside the scope of the FIPS 140-2 requirements.

12 Crypto-Officer and User Guidance

The calling application is the operator (crypto-officer or user depending on the password supplied) of the module. The Barracuda Cryptographic Software Module is for use on a GPC. It is the responsibility of the calling application to secure any keys or CSPs passed outside of the logical boundary of the module, to the calling application. The module does not provide any persistent storage of keys or CSPs.

13 Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Program Interface
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CO	Cryptographic Officer
CMAC	Cryptographic Message Authentication Code
CSP	Critical Security Parameter
CTR	Counter
DES	Data Encryption Scheme
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECB	Electronic Codebook
EMC	Electromagnetic Compatibility
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EDK	Encrypt Decrypt Key
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GCM	Galois Counter Mode
GPC	General Purpose Computer
HMAC	Keyed-Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
NDRBG	Non-Deterministic Random Bit Generator
OFB	Output Feedback
OS	Operating System
PKCS	Public Key Cryptography Standard
PKG	Public Key (Q) Generation
PKV	Public Key (Q) Validation
PQG	DSA parameters P, Q and G
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory

Acronym	Definition
RNG	Random Number Generator
RSA	Rivest, Shamir and Adleman Algorithm
RSASSA	RSA Signature Scheme with Appendix
SHA	Secure Hash Algorithm
Triple-DES	Triple-DES
XEX	XOR Encrypt XOR
XOR	Exclusive OR
XTS	XEX Tweakable Block Cipher with Ciphertext Stealing