PQShield Ltd.

PQCryptoLib-Core

Software version 1.0.2

FIPS 140-3 Level 1 Non-Proprietary Security Policy

Document version 1.0, June 2026

UK Headquarters: Prama House, 267 Banbury Road, Oxford, England OX2 7HT, GB.

© PQShield Ltd 2026

Table of Contents

1 General **5**
- 1.1 Overview 5
- 1.2 Security Levels 5
- 1.3 Additional Information 5

2 Cryptographic Module Specification **6**
- 2.1 Description 6
  - 2.1.1 Purpose and Use: 6
  - 2.1.2 Cryptographic Boundary 6
- 2.2 Tested and Vendor Affirmed Module Version and Identification 7
  - 2.2.1 Tested Module Identification 7
  - 2.2.2 Tested Operational Environments 8
- 2.3 Excluded Components 8
- 2.4 Modes of Operation 8
- 2.5 Algorithms 8
  - 2.5.1 Approved Algorithms 8
  - 2.5.2 Vendor-Affirmed Algorithms 12
- 2.6 Security Function Implementations 13
- 2.7 Algorithm Specific Information 17
  - 2.7.1 DRBG 17
  - 2.7.2 ML-DSA 17
  - 2.7.3 Key Agreement 17
  - 2.7.4 Key Establishment 17
- 2.8 RBG and Entropy 17
  - 2.8.1 RDSEED 17
  - 2.8.2 Linux RNG 17
  - 2.8.3 DRBG seeding 18
- 2.9 Key Generation 18
  - 2.9.1 CKG 18
- 2.10 Key Establishment 18
  - 2.10.1 Key Agreement 18
  - 2.10.2 Key Establishment 18
- 2.11 Industry Protocols 18
- 2.12 Additional Information 18

3 Cryptographic Module Interfaces **19**
- 3.1 Ports and Interfaces 19

4 Roles, Services, and Authentication **20**
- 4.1 Authentication Methods 20
- 4.2 Roles 20
- 4.3 Approved Services 21
  - 4.3.1 Service Indicators 30
- 4.4 Non-approved services 30
- 4.5 External software loaded 30

5 Software Security **31**
5.1 Integrity Techniques 31
5.2 Initiate on Demand 31

6 Operational Environment **32**
6.1 Operational Environment Type and Requirements 32
6.1.1 How Requirements are Satisfied 32

7 Physical Security **33**

8 Non-Invasive Security **34**

9 Sensitive Security Parameters Management **35**
9.1 Storage Areas 35
9.2 SSP Input-Output Methods 35
9.3 SSP Zeroization Methods 35
9.4 SSPs 36
9.5 Transitions 42

10 Self-Tests **43**
10.1 Pre-Operational Self-Tests 43
10.2 Conditional Self-Tests 43
10.3 Periodic self-test information 44
10.4 Error States 45
10.5 Operator Initiation of Self-Tests 45

11 Life-Cycle Assurance **46**
11.1 Installation, Initialization, and Startup Procedures 46
11.2 Administrator Guidance 46
11.3 Non-Administrator Guidance 46
11.4 Design and Rules 46
11.5 Maintenance Requirements 47
11.6 End of Life 47

12 Mitigation of Other Attacks **48**

References and Definitions **49**

List of Tables

Table 1: Security Levels 5
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 7
Table 3: Tested Operational Environments - Software, Firmware, Hybrid 8
Table 4: Modes List and Description 8
Table 5: Approved Algorithms 11
Table 6: Vendor-Affirmed Algorithms 12
Table 7: Security Function Implementations 16
Table 8: Ports and Interfaces 19
Table 9: Roles 20
Table 10: Approved Services 29
Table 11: Storage Areas 35
Table 12: SSP Input-Output Methods 35
Table 13: SSP Zeroization Methods 35
Table 14: SSP Table 1 38
Table 15: SSP Table 2 41
Table 16: Pre-Operational Self-Tests 43
Table 17: Conditional Self-Tests 44
Table 18: Pre-Operational Periodic Information 44
Table 19: Conditional Periodic Information 45
Table 20: Error States 45
Table 21: References 49
Table 22: Acronyms and Definitions 51

List of Figures

Figure 1: Block Diagram 7

1 General

1.1 Overview

This document defines the non-proprietary Security Policy for the PQShield **PQCryptoLib-core** module, hereafter denoted **the Module**. The **Module** is a library of cryptographic primitives with a C interface offering security against quantum adversaries. PQShield is a spin-out of the University of Oxford which provides expertise in the design and implementation of quantum-resistant cryptography for software and hardware applications.

1.2 Security Levels

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

Table 1: Security Levels

1.3 Additional Information

Copyright PQShield LTD, 2026

PQShield Public Material – May be reproduced only in its original entirety (without revision).

2 Cryptographic Module Specification

2.1 Description

2.1.1 Purpose and Use:

The **Module** is classified as a software cryptographic module. It is a software library of cryptographic primitives with a unified and easy to use API. The **Module** is intended for use by US Federal agencies or other markets that require FIPS 140-3 validated general purpose cryptographic library running on GPC.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

2.1.2 Cryptographic Boundary

The **Module** is a software library providing C-language application program interface (API) for use by other processes that require cryptographic functionality. The physical form of the **Module** is depicted in Figure 1. The cryptographic boundary is a library called libpqcrypto.so.1.0.2 (software version 1.0.2) that is intended to link with the Application. The **Module** can run in a multi-threaded environment, it requires a POSIX thread library (pthread). The **Module** may call the CPU directly, that is done by performance optimized functions and to get entropy from the CPU. The **Module** performs no communications other than with the calling application, operating system and the CPU.

Tested Operational Environment's Physical Perimeter (TOEPP)

The TOEPP is defined by the general-purpose computer onto which the **Module** has been installed.

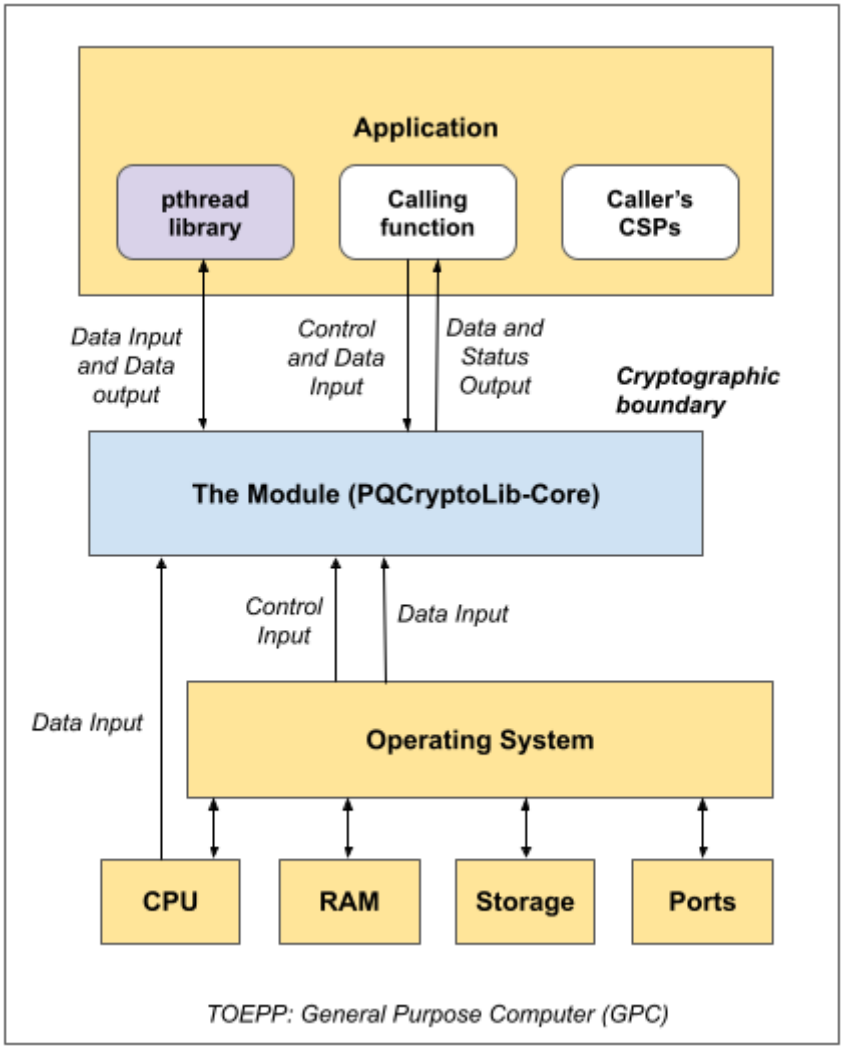


Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

The **PQCryptoLib-Core** cryptographic module is tested on the following operational environment.

2.2.1 Tested Module Identification

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libpqcrypto.so.1.0.2	1.0.2	-	MAC verification

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

2.2.2 Tested Operational Environments

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Ubuntu 24.04 LTS	Dell PowerEdge 740	Intel Xeon Platinum 8276	Yes		1.0.2
Ubuntu 24.04 LTS	Dell PowerEdge 740	Intel Xeon Platinum 8276	No		1.0.2
Ubuntu 22.04 LTS	Dell Latitude 7340	Intel Core i7-1365U	Yes		1.0.2
Ubuntu 22.04 LTS	Dell Latitude 7340	Intel Core i7-1365U	No		1.0.2

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

2.3 Excluded Components

There are no components within the cryptographic boundary that are excluded from the **Module**.

2.4 Modes of Operation

The **Module** supports Approved mode of operation only. The **Module** is operating the Approved mode after the CO loads the **Module** into memory and all self-tests are completed successfully. The **Module** does not support degraded mode and operates only in normal mode.

Modes List and Description

Mode Name	Description	Types of Services	Status Indicator
Approved	Approved mode	Approved	-

Table 4: Modes List and Description

2.5 Algorithms

The **Module** implements the FIPS Approved and vendor-affirmed cryptographic functions listed in the tables below.

2.5.1 Approved Algorithms

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyGen (FIPS186-5)	A7693	Curve - P-256 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A7693	Curve - P-256 Hash Algorithm - SHA2-256	FIPS 186-5
ECDSA SigVer	A7693	Curve - P-256	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
(FIPS186-5)		Hash Algorithm - SHA2-256	
Hash DRBG	A7693	Prediction Resistance - No Supports Reseed - Yes Mode - SHA2-256 Entropy Input - Entropy Input: 256-2048 Increment 8 Nonce - Nonce: 128-2048 Increment 8 Personalization String Length - Personalization String Length: 0-2048 Increment 8 Additional Input - Additional Input: 0-2048 Increment 8 Returned Bits - 4096	SP 800-90A Rev. 1
HMAC-SHA2-224	A7693	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA2-256	A7693	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7693	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7693	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA3-224	A7693	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA3-256	A7693	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA3-384	A7693	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
HMAC-SHA3-512	A7693	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 112-524288 Increment 8 Message Length - Message Length: 0-4096 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A7693	Domain Parameter Generation Methods - P-256 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
KDA HKDF SP800-56Cr2	A7693	Fixed Info Pattern - uPartyInfo||vPartyInfo||l Fixed Info Encoding - concatenation Derived Key Length - 1024 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Perform Multiple Expansion Tests - Yes Uses Hybrid Shared Secret - Yes Auxiliary Shared Secret Length - Auxiliary Shared Secret Length: 112-65336 Increment 8	SP 800-56C Rev. 2
KDA TwoStep SP800-56Cr2	A7693	MAC Salting Methods - default, random Fixed Info Pattern - uPartyInfo||vPartyInfo||l Fixed Info Encoding - concatenation KDF Mode - feedback MAC Modes - HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512 Fixed Data Order - afterFixedData Counter Lengths - 8 The KDF supports an empty IV - No The KDF requires an empty IV - No Supported Lengths - Supported Lengths: 8-4096 Increment 8 Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 Perform Multiple Expansion Tests - Yes Uses Hybrid Shared Secret - Yes Auxiliary Shared Secret Length - Auxiliary Shared Secret Length: 112-65336 Increment 8	SP 800-56C Rev. 2
KDF SP800-108	A7693	KDF Mode - Feedback MAC Mode - HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512 Supported Lengths - Supported Lengths: 8-4096 Increment 8 Fixed Data Order - After Fixed Data, none Counter Length - 0, 8 Supports Empty IV - Yes Custom Key In Length - 224	SP 800-108 Rev. 1
ML-DSA KeyGen	A7693	Parameter Sets - ML-DSA-44, ML-DSA-65, ML-DSA-87	FIPS 204
ML-DSA SigGen	A7693	Signature Interfaces - internal Deterministic - Yes External Mu - No Parameter Sets - ML-DSA-44, ML-DSA-65, ML-DSA-87 Message Length - Message Length: 8-65536 Increment 8	FIPS 204
ML-DSA SigVer	A7693	Signature Interfaces - internal External Mu - No	FIPS 204

Algorithm	CAVP Cert	Properties	Reference
		Parameter Sets - ML-DSA-44, ML-DSA-65, ML-DSA-87 Message Length - Message Length: 8-65536 Increment 8	
ML-KEM EncapDecap	A7693	Parameter Sets - ML-KEM-1024, ML-KEM-512, ML-KEM-768 Functions - Decapsulation, Decapsulation Key Check, Encapsulation, Encapsulation Key Check	FIPS 203
ML-KEM KeyGen	A7693	Parameter Sets - ML-KEM-1024, ML-KEM-512, ML-KEM-768	FIPS 203
SHA2-224	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-224	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A7693	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHAKE-128	A7693	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A7693	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
TLS v1.3 KDF (CVL)	A7693	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK-DHE	SP 800-135 Rev. 1

Table 5: Approved Algorithms

2.5.2 Vendor-Affirmed Algorithms

Name	Properties	Implementation	Reference
CKG	Key type:Symmetric and asymmetric	N/A	SP 800-133r2 section 4

Table 6: Vendor-Affirmed Algorithms

NOTE: The **Module** does not implement any Non-Approved Algorithms.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
CKG	CKG	Symmetric key generation and seeds for asymmetric keys		CKG: ()
DRBG	DRBG	Random bit generation		Hash DRBG: (A7693) SHA2-256: (A7693)
ECC KeyGen	AsymKeyPair-KeyGen	ECDSA Key pair generation		ECDSA KeyGen (FIPS186-5): (A7693)
ECC SigGen	DigSig-SigGen	ECDSA Signature generation		ECDSA SigGen (FIPS186-5): (A7693) SHA2-256: (A7693)
ECC SigVer	DigSig-SigVer	ECDSA Signature verification		ECDSA SigVer (FIPS186-5): (A7693) SHA2-256: (A7693)
Hash	SHA	Compute a hash		SHA2-224: (A7693) SHA2-256: (A7693) SHA2-384: (A7693) SHA2-512: (A7693) SHA3-224: (A7693) SHA3-256: (A7693) SHA3-384: (A7693) SHA3-512: (A7693)
HKDF	KAS-56CKDF	Key derivation		KDA HKDF SP800-56Cr2: (A7693) SHA2-224: (A7693) SHA2-256: (A7693) SHA2-384: (A7693) SHA2-512: (A7693)

Name	Type	Description	Properties	Algorithms
				SHA3-224: (A7693) SHA3-256: (A7693) SHA3-384: (A7693) SHA3-512: (A7693)
KBKDF	KBKDF	Key-based key derivation		KDF SP800-108: (A7693) HMAC-SHA2-224: (A7693) HMAC-SHA2-256: (A7693) HMAC-SHA2-384: (A7693) HMAC-SHA2-512: (A7693) HMAC-SHA3-224: (A7693) HMAC-SHA3-256: (A7693) HMAC-SHA3-384: (A7693) HMAC-SHA3-512: (A7693)
KDF TwoStep	KAS-56CKDF	Two-step key derivation		KDA TwoStep SP800-56Cr2: (A7693) HMAC-SHA2-224: (A7693) HMAC-SHA2-256: (A7693) HMAC-SHA2-384: (A7693) HMAC-SHA2-512: (A7693) HMAC-SHA3-224: (A7693) HMAC-SHA3-256: (A7693) HMAC-SHA3-384: (A7693) HMAC-SHA3-512: (A7693)
KDF TLS	KAS-135KDF	Key derivation for TLS		TLS v1.3 KDF: (A7693) HMAC-SHA2-256: (A7693) HMAC-SHA2-384: (A7693)
KEM KeyGen	AsymKeyPair-KeyGen	ML-KEM Key pair generation		ML-KEM KeyGen: (A7693) SHA3-256: (A7693) SHA3-512: (A7693) SHAKE-128: (A7693)

Name	Type	Description	Properties	Algorithms
KEM Encap	KEM-Encap	ML-KEM Encapsulation	IG D.S:Scenario 1	ML-KEM EncapDecap: (A7693) Function: Encapsulation SHA3-256: (A7693) SHA3-512: (A7693) SHAKE-128: (A7693) SHAKE-256: (A7693)
KEM Decap	KEM-Decap	ML-KEM Decapsulation	IG D.S:Scenario 1	ML-KEM EncapDecap: (A7693) Function: Decapsulation SHA3-256: (A7693) SHA3-512: (A7693) SHAKE-128: (A7693) SHAKE-256: (A7693)
MAC	MAC	Generate or verify a message authentication code		HMAC-SHA2-224: (A7693) HMAC-SHA2-256: (A7693) HMAC-SHA2-384: (A7693) HMAC-SHA2-512: (A7693) HMAC-SHA3-224: (A7693) HMAC-SHA3-256: (A7693) HMAC-SHA3-384: (A7693) HMAC-SHA3-512: (A7693)
ML-DSA KeyGen	AsymKeyPair-KeyGen	ML-DSA key pair generation		ML-DSA KeyGen: (A7693) SHAKE-128: (A7693) SHAKE-256: (A7693)
ML-DSA Sign	DigSig-SigGen	ML-DSA signature generation		ML-DSA SigGen: (A7693) SHAKE-128: (A7693) SHAKE-256: (A7693)
ML-DSA Verify	DigSig-SigVer	ML-DSA signature verification		ML-DSA SigVer: (A7693) SHAKE-128: (A7693) SHAKE-256: (A7693)
KAS-SSC	KAS-SSC	Shared secret computation	IG D.F:Scenario 2, path	KAS-ECC-SSC Sp800-56Ar3:

Name	Type	Description	Properties	Algorithms
			(2)	(A7693)
XOF	XOF	Compute a message digest using an XOF		SHAKE-128: (A7693) SHAKE-256: (A7693)

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 DRBG

Hash_DRBG is instantiated through the API. The security strength for random bit generation assumes that the random bit generator has been provided with at least 256-bits of entropy.

2.7.2 ML-DSA

As disclosed in the Approve Algorithm table above, the **Module's** implementations of ML-DSA security functions have been CAVP-tested by exercising the "internal" functions defined in FIPS 204. However, these functions are not part of the **Module's** API, and are not exposed to the calling application.

To use ML-DSA security functions, the operator must use the general-purpose "Signing keypair generation", "Signature Generation" and "Signature Verification" services and their corresponding API commands.API commands are described in the Users Manual (see section 11.2).

2.7.3 Key Agreement

The **Module** does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.7.4 Key Establishment

The module does not establish SSPs using an approved key encapsulation mechanism (KEM). However, it does offer some or all of the underlying KEM cryptographic functionality to be used by an external operator/application as part of an approved KEM.

2.8 RBG and Entropy

The **Module** leverages two different entropy sources, one provided by the CPU and the other provided by the operating system. Neither source has been ESV tested.

2.8.1 RDSEED

The RDSEED is a CPU instruction, used to get entropy directly from the Intel CPU. The module requests 256-bits of entropy from this source per call.

2.8.2 Linux RNG

Additionally, the **Module** uses system call `getrandom` (in Linux), which pulls entropy from environmental noise. The module requests 256-bits of entropy from this source per call.

2.8.3 DRBG seeding

Returned bytes from both sources are XORd together and provided to the **Module's** RNG initialization function as entropy. In case an entropy source fails to produce entropy, the RNG initialization procedure sets FSM to an ERROR state, resulting in the **Module** being not available for use in the approved mode of operation. The module conforms to FIPS 140-3 IG 9.3.A scenario 1b, thus the following caveat is applicable:

The module generates SSPs (e.g., keys) whose strengths are modified by available entropy

2.9 Key Generation

2.9.1 CKG

The cryptographic module performs Cryptographic Key Generation (CKG) in compliance with section 4 of SP 800-133r2. Symmetric cryptographic keys and seeds for asymmetric keys are generated by the direct unmodified output of the module's approved DRBG.

2.10 Key Establishment

The **Module** does not implement any key establishment schemes. However, it offers the following functionality related to key establishment.

2.10.1 Key Agreement

The **Module's** implementation of KAS-ECC-SSC can be used by a calling application to compute a shared secret for ECDH keys generated on the P-256 curve. Such shared secrets would provide 128-bits of security strength.

2.10.2 Key Establishment

The **Module's** implementations of ML-KEM security functions can be used by a calling application to establish a shared secret key (SSK). The SSK provides 128, 192 or 256-bits of security strength, depending on the parameter set employed.

2.11 Industry Protocols

The **Module** does not support any industry protocols.

2.12 Additional Information

If run as a recipient, the **Module** performs partial public key validation as specified in SP 800-56Ar3, section 5.6.2.3.4.

3 Cryptographic Module Interfaces

The **Module** is a software-only implementation. All keys, encrypted data, and control information are exchanged through calls to library functions (logical interfaces). As a software module, it has no access to the physical ports (physical covers, manual controls, physical status indicators) of the host GPC.

The **Module's** ports and associated FIPS-defined logical interface categories are listed in the table below.

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters that specify plaintext data; ciphertext or signed data; cryptographic keys, initialization vectors; kernel I/O, entropy.
N/A	Data Output	API output parameters that receive plaintext data, ciphertext data, digital signatures cryptographic keys and initialization vectors. API Return values.
N/A	Control Input	Function calls and control data (e.g., algorithms, algorithm modes, or module settings). Values stored in the CPU and read by the Module during the initialization phase.
N/A	Status Output	API return values.

Table 8: Ports and Interfaces

The **Module** does not implement a control output interface.

When the **Module** is performing self-tests, during zeroization or is in the error state, all output via data output interface is inhibited.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The Module does not support operator authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
CO	Role	Crypto-Officer	None

Table 9: Roles

The **Module** implements a single instance of one authorized role: Crypto Officer (CO). The role is implicitly assumed by the entity accessing services implemented by the **Module** and is authorized to access all services provided by the **Module**. Only one concurrent user is allowed, namely, a CO is considered the owner of executing thread. The **Module** does not support a maintenance role or bypass capability.

4.3 Approved Services

In the table below, SSP access is defined as follows:

- **G** = Generate: The Module generates or derives the SSP.
- **R** = Read: The SSP is read from the Module (e.g. the SSP is output).
- **W** = Write: The SSP is imported, or written to the Module.
- **E** = Execute: The Module uses the SSP in performing a cryptographic operation.
- **Z** = Zeroize: The Module zeroizes the SSP

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Close	Deinitializes the module	-	-	-	None	CO - DRBG-C: Z - DRBG-V: Z
Disable PAA	Switch off PAA	-	Command	Status	None	CO
Data cleanup	Zeroises sensitive data in the buffer	-	Command	Status	None	CO - SYM-K: Z
Hash_DRBG operation context cleanup	Zeroize Hash_DRBG SSPs	-	Context	Status	None	CO - DRBG-C: Z - DRBG-V: Z
Hash_DRBG pseudorandom byte stream generation	Generate pseudo random byte strings and keys	Implicit	Size of requested random bytes and optional additional input string	Random bytes	DRBG	CO - DRBG-C: E - DRBG-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						V: E - DRBG-OUT: G
Hash_DRBG reseeding	Seed Hash_DRBG instance with externally supplied entropy	Implicit	Entropy, additional data	Status	DRBG	CO - DRBG-EI: W - DRBG-SEED: G - DRBG-C: G - DRBG-V: G
Hash_DRBG instantiation	Seed Hash_DRBG instance with externally supplied entropy	Implicit	Entropy, nonce, personalization string	Status	DRBG	CO - DRBG-EI: W - DRBG-SEED: G - DRBG-C: G - DRBG-V: G - DRBG-N: W
Hashing	Calculate a message digest	Implicit	Data to be hashed	Message digest	Hash	CO
Key agreement	Calculate shared secret key from private and public keys	Implicit	Peer public key	Shared secret	KAS-SSC	CO - ECDH-PRV: E - ECDH-PUB: E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- ECDH-K-Z: G,R
Key agreement key-pair cleanup	Zeroize SSPs	-	Context	Status	None	CO - ECDH-PRV: Z - ECDH-PUB: Z
Key agreement key-pair export	Export a public and/or private ECDH key(s)	-	Context	Private and/or public key	None	CO - ECDH-PRV: R - ECDH-PUB: R
Key agreement key-pair generation	Generate a key-pair for key agreement	Implicit	Command	Status	CKG ECC KeyGen	CO - ECDH-PRV: G - ECDH-PUB: G - DRBG-OUT: E
Key agreement key-pair import	Import a public and/or private ECDH key(s)	-	Private and/or public key	Status	None	CO - ECDH-PRV: W - ECDH-PUB: W
Key derivation	Symmetric key derivation from shared secret key	Implicit	Shared secret	Derived key	HKDF KBKDF	CO - ECDH-K-Z: E
KDF context cleanup	Removed sensitive key material from the HKDF context	-	Context	Status	None	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key expansion in two-step key derivation	PRF-based key derivation function	Implicit	Key derivation key	Derived key	KDF TwoStep	CO - HKDF-K: G,R - HKDF-KDK: W,E
Keyed hash context cleanup	Zeroize HMAC SSPs	-	Context	Status	None	CO - HMAC-K: Z
Keyed hash key import	Import a HMAC symmetric key into operation context	-	Key	Status	None	CO - HMAC-K: W
Keyed hash signing	Calculate an authentication tag on data	Implicit	Message	MAC	MAC	CO - HMAC-K: E
Keyed hash verification	Validate an authentication tag on data	Implicit	Message, MAC	Ruling	MAC	CO - HMAC-K: E
KEM context cleanup	Deletes operation context	-	Context	Status	None	CO - KEM-DK: Z - KEM-SSK: Z
KEM encapsulation	Generates a shared secret and ciphertext	Implicit	Public key	Shared secret key, ciphertext	KEM Encap	CO - KEM-EK: E - KEM-SSK: G,R - DRBG-OUT: E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
KEM decapsulation	Decapsulates shared secret from provided ciphertext	Implicit	Private key, Ciphertext	Shared secret	KEM Decap	CO - KEM-DK: E - KEM-SSK: G,R
KEM key-pair export	Exports and public and/or private key from the operation context	-	Context	Private and/or public key	None	CO - KEM-EK: R - KEM-DK: R
KEM key-pair generation	Generates a key pair	Implicit	Command	Status	CKG KEM KeyGen	CO - KEM-EK: G - KEM-DK: G - DRBG-OUT: E
KEM key-pair import	Imports a key or keypair to the KEM operation context	-	Private and/or public key	Status	None	CO - KEM-EK: W - KEM-DK: W
Module initialization	Initializes internal structure of the Module	-	Command	Status	None	CO
Randomness extraction	The randomness extraction step in two- step key derivation procedure as specified by [56Cr2]	Implicit	Shared secret	Key derivation key	KDF TwoStep	CO - HKDF-KDK: G
Randomness extraction with	The randomness extraction step in two- step key derivation procedure	Implicit	Shared secret, auxiliary shared secret	Key derivation key	KDF TwoStep	CO - HKDF-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
auxiliary shared secret	with auxiliary shared secret T as specified by [56Cr2]					KDK: G
Self-test	Self-tests invoked by reboot	-	-	-	None	CO
Show name	Returns name of the Module	-	Command	Text	None	CO
Show status	Returns state of the Module	-	Command	Status	None	CO
Show version	Returns version of the Module	-	Command	Version	None	CO
Signature generation	Calculate digital signature using a private key	Implicit	Message or digest	Signature	ECC SigGen ML-DSA Sign	CO - ECDSA-PRV: E - ML-DSA-PRV: E - DRBG-OUT: E
Signature verification	Verify digital signature using a public key	Implicit	Message or digest, signature	Ruling	ECC SigVer ML-DSA Verify	CO - ECDSA-PUB: E - ML-DSA-PUB: E
Signing keypair cleanup	Zeroize SSPs and deletes operation context	-	Context	Status	None	CO - ECDSA-PRV: Z - ECDSA-PUB: Z - ML-DSA-PRV: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- ML-DSA-PUB: Z
Signing keypair export	Export a public and/or private ECDSA key(s)	-	Context	Private and/or public key	None	CO - ECDSA-PRV: R - ECDSA-PUB: R - ML-DSA-PRV: R - ML-DSA-PUB: R
Signing keypair generation	Generate a key-pair for signing and verification	Implicit	Command	Status	CKG ECC KeyGen ML-DSA KeyGen	CO - ECDSA-PRV: G - ECDSA-PUB: G - ML-DSA-PRV: G - ML-DSA-PUB: G - DRBG-OUT: E
Signing keypair import	Import a public and/or private asymmetric key into operation context	-	Private and/or public key	Status	None	CO - ECDSA-PRV: W - ECDSA-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						PUB: W - ML-DSA-PRV: W - ML-DSA-PUB: W
Symmetric key generation	Generates a symmetric key of size between 112 and 256 bits, multiple of 8.	-	Command	Symmetric key	CKG	CO - SYM-K: G,R - DRBG-OUT: E
TLS v1.3 KDF	Expands TLS secrets into key as specified by the RFC8446	Implicit	Key derivation key, shared secret, transcript hash, pre-shared key	Handshake keys	KDF TLS	CO - TLS-PSK: W,E - TLS-ES: G,E - TLS-EBK : G,E - TLS-RBK: G,E - TLS-CETS: G,E - TLS-EEMS: G,E - TLS-HS : G,E - TLS-CHTS: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- TLS-SHTS: G,E - TLS-MS: G,E - TLS-CATS: G,E - TLS-SATS: G,E - TLS-EMS: G,E - TLS-RMS: G,E - TLS Key: G,R
XOF	Calculate output from XOF	Implicit	Message	Message digest	XOF	CO

Table 10: Approved Services

The **Module** does not support an invocable self-test service. Self-test are invoked by rebooting the host device.

4.3.1 Service Indicators

The **Module** only supports approved algorithms. This means that the module is compliant to IG 2.4.C Scenario 2 by default and no explicit indicator is required. The successful completion of the service satisfies the implicit indication.

4.4 Non-approved services

The **Module** does not support a non-approved mode and does not offer any non-approved services.

4.5 External software loaded

The **Module** does not support software loading.

5 Software Security

5.1 Integrity Techniques

The **Module** is composed of the software component delivered as a library in a binary form. The integrity of the module is tested during initialization by computing a MAC of the entire software image using HMAC-SHA2-512. In case the integrity test fails, the library is moved into an ERROR state and the **Module** needs to be unloaded from the memory of the Application that uses the **Module**.

5.2 Initiate on Demand

The integrity test can be invoked on demand by rebooting the **Module** and by calling pqcl_init().

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

The tested operational environments are listed in Section 2.2 above.

6.1.1 How Requirements are Satisfied

Each Operational Environment manages processes in a logically separated manner, each process is assigned a private memory space, access to that space is restricted to the process running the **Module** and trusted parts of the operational environment. Process private memory space is used to store CSPs and SSPs. The CO role is considered the owner of the calling application that instantiates the module.

7 Physical Security

Physical security is not applicable to software-only modules.

8 Non-Invasive Security

The **Module** does not implement any mitigation method against non-invasive attack.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	SDRAM of the host platform	Dynamic

Table 11: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Import	Calling application	RAM	Plaintext	Manual	Electronic	
Export	RAM	Calling application	Plaintext	Manual	Electronic	

Table 12: SSP Input-Output Methods

9.3 SSP Zeroization Methods

SSPs are zeroized when the appropriate operation context is destroyed, when the **Module** is unloaded, or on power cycling/reboot of host platform. Input and output interfaces are inhibited while zeroization is performed.

Zeroization needs to be performed under the control of the Crypto Officer. SSPs are overwritten with fixed pattern (octet-string filled with 0) and hence are not retrievable upon zeroization.

Zeroization Method	Description	Rationale	Operator Initiation
Service	Invoking a 'zeroisation' service	Context SSPs are overwritten with 0's	Issue a context cleanup API command
Reboot	Power-cycle the host device	All RAM is zeroised by a power-cycle.	Unplugging the host device
Auto	Automatic zeroisation of SSPs that are no longer needed.	Memory is overwritten with 0's	N/A
Unload	Unloading the module	Memory is overwritten with 0's	Calling the 'pqcl_close' API command

Table 13: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG-EI	Hash_DRBG entropy input	256 - 256	Entropy - CSP			DRBG
DRBG-N	Hash_DRBG nonce	128 to 2048 - 128 to 256	Nonce - CSP			DRBG
DRBG-SEED	Hash_DRBG initialization and reseeding	440 - 256	DRBG state - CSP	DRBG		DRBG
DRBG-C	Hash_DRBG C value	440 - -	DRBG state - CSP	DRBG		DRBG
DRBG-V	Hash_DRBG v value	440 - -	DRBG state - CSP	DRBG		DRBG
DRBG-OUT	DRBG output	4096 - 256	DRBG - CSP	DRBG		CKG ECC KeyGen ECC SigGen KEM KeyGen KEM Encap ML-DSA KeyGen
HMAC-K	HMAC authentication key and OPAD value	112 to 256 - 112 to 256	Symmetric - CSP			MAC
HKDF-K	KDA-HKDF secret key	228 to 65536 - ???	Symmetric - CSP			HKDF KBKDF KDF TwoStep
HKDF-KDK	Key-derivation key	224, 256, 384, 512 - 224, 256, 384, 512	Symmetric - CSP			HKDF KBKDF KDF TwoStep
ECDSA-PRV	ECDSA signature generation key	P-256 - 128	Asymmetric - CSP			ECC SigGen

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
ECDSA-PUB	ECDSA signature verification key	P-256 - 128	Asymmetric - PSP			ECC SigVer
ECDH-PRV	ECDH key agreement private key	P-256 - 128	Asymmetric - CSP			KAS-SSC
ECDH-PUB	ECDH key agreement public key	P-256 - 128	Asymmetric - PSP			KAS-SSC
ECDH-K-Z	ECDH shared secret	256 - 128	Shared secret - CSP		KAS-SSC	KDF TLS
ML-DSA-PRV	ML-DSA private key	2560, 4032, 4896 bytes - 128, 192, 256	Asymmetric - CSP	ML-DSA KeyGen		ML-DSA Sign
ML-DSA-PUB	ML-DSA public key	1312, 1952, 2592 bytes - 128, 192, 256	Asymmetric - PSP	ML-DSA KeyGen		ML-DSA Verify
KEM-EK	ML-KEM encapsulation key	800, 1184, 1568 bytes - 128, 192, 256	Asymmetric - PSP	KEM KeyGen		KEM Encap
KEM-DK	ML-KEM decapsulation key	1632, 2400, 3168 bytes - 128, 192, 256	Asymmetric - CSP	KEM KeyGen		KEM Decap
KEM-SSK	ML-KEM shared secret key	256 - 128, 192, 256	Symmetric - CSP	KEM Encap	KEM Decap	
TLS-PSK	TLS v1.3 pre-shared key,	112 to 256 - 112 to 256	Symmetric - CSP			KDF TLS
TLS-ES	TLS v1.3 early secret	256, 384 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-EBK	TLS v1.3 binder secret for external PSKs	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-RBK	TLS v1.3 binder key for resumption PSKs	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-CETS	TLS v1.3 early traffic secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
TLS-EEMS	TLS v1.3 early exporter master secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-HS	TLS v1.3 handshake secret	256, 384 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-CHTS	TLS v1.3 client handshake traffic secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-SHTS	TLS v1.3 server handshake traffic secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-MS	TLS v1.3 master secret	256, 384 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-CATS	TLS v1.3 client application traffic secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-SATS	TLS v1.3 server application traffic secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-EMS	TLS v1.3 exporter master secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS-RMS	TLS v1.3 resumption master secret	128 - 128	Secret - CSP	KDF TLS		KDF TLS
TLS Key	TLS v1.3 key	128, 256 - 128, 256	Symmetric - CSP	KDF TLS		
SYM-K	General purpose symmetric key	112 to 256 - 112 to 256	Symmetric - CSP	CKG		

Table 14: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG-EI	Import	RAM:Plaintext		Auto	
DRBG-N	Import	RAM:Plaintext		Auto	
DRBG-SEED		RAM:Plaintext		Auto	DRBG-EI:Derived From DRBG-N:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG-C		RAM:Plaintext		Reboot Unload	DRBG-SEED:Derived From
DRBG-V		RAM:Plaintext		Reboot Unload	DRBG-SEED:Derived From
DRBG-OUT		RAM:Plaintext		Auto	DRBG-C:Derived From DRBG-V:Derived From
HMAC-K	Import	RAM:Plaintext		Service	
HKDF-K	Export	RAM:Plaintext		Service	
HKDF-KDK	Import	RAM:Plaintext		Service	
ECDSA-PRV	Import Export	RAM:Plaintext		Service	ECDSA-PUB:Paired With
ECDSA-PUB	Import Export	RAM:Plaintext		Service	ECDSA-PRV:Paired With
ECDH-PRV	Import Export	RAM:Plaintext		Service	ECDH-PUB:Paired With
ECDH-PUB	Import Export	RAM:Plaintext		Service	ECDH-PRV:Paired With
ECDH-K-Z	Export	RAM:Plaintext		Service	ECDH-PRV:Derived From ECDH-PUB:Derived From
ML-DSA-PRV	Import Export	RAM:Plaintext		Service	ML-DSA-PUB:Paired With
ML-DSA-PUB	Import Export	RAM:Plaintext		Service	ML-DSA-PRV:Paired With
KEM-EK	Import Export	RAM:Plaintext		Service	KEM-DK:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
KEM-DK	Import Export	RAM:Plaintext		Service	KEM-EK:Paired With
KEM-SSK	Export	RAM:Plaintext		Service	DRBG-EI:Derived From
TLS-PSK	Import	RAM:Plaintext		Service	
TLS-ES		RAM:Plaintext		Service	TLS-PSK:Derived From
TLS-EBK		RAM:Plaintext		Service	TLS-ES:Derived From
TLS-RBK		RAM:Plaintext		Service	TLS-EBK :Derived From
TLS-CETS		RAM:Plaintext		Service	TLS-ES:Derived From
TLS-EEMS		RAM:Plaintext		Service	TLS-ES:Derived From
TLS-HS		RAM:Plaintext		Service	ECDH-K-Z:Derived From TLS-ES:Derived From
TLS-CHTS		RAM:Plaintext		Service	TLS-HS :Derived From
TLS-SHTS		RAM:Plaintext		Service	TLS-HS :Derived From
TLS-MS		RAM:Plaintext		Service	TLS-HS :Derived From
TLS-CATS		RAM:Plaintext		Service	
TLS-SATS		RAM:Plaintext		Service	
TLS-EMS		RAM:Plaintext		Service	
TLS-RMS		RAM:Plaintext		Service	
TLS Key	Export	RAM:Plaintext		Service	TLS-CATS:Derived From TLS-SATS:Derived From TLS-EMS:Derived From TLS-RMS:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
SYM-K	Export	RAM:Plaintext		Service	DRBG-OUT:Derived From

Table 15: SSP Table 2

9.5 Transitions

No algorithm or security strength transitions are forecasted to occur over the lifetime of the validation.

10 Self-Tests

The **Module** performs self-tests to ensure the proper operation. Per FIPS 140-3, these are categorized as either pre-operational self-tests or conditional self-tests. Pre-operational self-tests are available on demand by power cycling the **Module** and calling the Module Initialization service.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
Integrity test	256-bit key	MAC	SW/FW Integrity	Successful boot	MAC verification

Table 16: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
DRBG	SHA2-256	KAT	CAST	Successful boot	Instantiate, generate, reseed	on boot
ECDSA	P-256	KAT	CAST	Successful boot	Sign & verify	on boot
HMAC	SHA2-512	KAT	CAST	Successful boot	Generate	on boot
KAS-SSC	P-256	KAT	CAST	Successful boot	Compute	on boot
KBKDF	HMAC-SHA2-256	KAT	CAST	Successful boot	Derive	on boot
KDA HKDF	HMAC-SHA2-256	KAT	CAST	Successful boot	Derive	on boot
KDA Two Step	HMAC-SHA2-384	KAT	CAST	Successful boot	Extract & expand	on boot
SHAKE	SHAKE128	KAT	CAST	Successful boot	Generate	on boot
SHA	SHA2-256	KAT	CAST	Successful boot	Generate	on boot
TLS KDF	SHA2-256	KAT	CAST	Successful boot	Derive	on boot
ML-KEM Generate	ML-KEM-512	KAT	CAST	Successful boot		on boot
ML-KEM Encaps	ML-KEM-512	KAT	CAST	Successful boot	-	on boot

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ML-KEM Decaps	ML-KEM-512	KAT	CAST	Successful boot	-	on boot
ML-DSA Generate	ML-DSA-44	KAT	CAST	Successful boot	-	on boot
ML-DSA Sign	ML-DSA-44, all rejection sampling loop paths	KAT	CAST	Successful boot	-	on boot
ML-DSA Verify	ML-DSA-44	KAT	CAST	Successful boot	-	on boot
ECC Keys	-	PCT	PCT	No error	Sign and verify	on key pair generation
ML-KEM Keys	-	PCT	PCT	No error	Encaps, decaps	on key pair generation
ML-DSA Keys	-	PCT	PCT	No error	Sign and verify	on key pair generation

Table 17: Conditional Self-Tests

10.3 Periodic self-test information

The following self-tests can be invoked by the administrator as described in section 10.5 below. The module cannot invoke these tests automatically.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Integrity test	MAC	SW/FW Integrity	On demand	Reloading

Table 18: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
DRBG	KAT	CAST	On demand	Reloading
ECDSA	KAT	CAST	On demand	Reloading
HMAC	KAT	CAST	On demand	Reloading
KAS-SSC	KAT	CAST	On demand	Reloading
KBKDF	KAT	CAST	On demand	Reloading
KDA HKDF	KAT	CAST	On demand	Reloading
KDA Two Step	KAT	CAST	On demand	Reloading
SHAKE	KAT	CAST	On demand	Reloading
SHA	KAT	CAST	On demand	Reloading
TLS KDF	KAT	CAST	On demand	Reloading
ML-KEM Generate	KAT	CAST	On demand	Reloading

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ML-KEM Encaps	KAT	CAST	On demand	Reloading
ML-KEM Decaps	KAT	CAST	On demand	Reloading
ML-DSA Generate	KAT	CAST	On demand	Reloading
ML-DSA Sign	KAT	CAST	On demand	Reloading
ML-DSA Verify	KAT	CAST	On demand	Reloading
ECC Keys	PCT	PCT	-	-
ML-KEM Keys	PCT	PCT	-	-
ML-DSA Keys	PCT	PCT	-	-

Table 19: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error	-	Failed integrity test Failed CAST Failed PCT	Power cycle	Output status: PQCL_STATE_ERROR

Table 20: Error States

10.5 Operator Initiation of Self-Tests

The **Module** does not support an invocable self-test service. The operator may initiate self-tests on-demand by reloading the **Module**.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The **Module** is delivered in a form of dynamically linkable software library and the API declared in the header files. The **Module** is intended to be linked with the Application. An application that wishes to use the **Module**, includes `pqcl.h` and `fips.h` header files in its source code and links the application against the binary file of the **Module**.

The **Module** is initialized by the Application, by calling the Module Initialization service. That function must be called and finished before any other API function of the **Module** is used. Error code returned by the function must be checked. In case of successful initialization, the function returns `PQCL_SUCCESS` code and any other code returned by the function indicates initialization failure. The deinitialization of the library is done by the Application by calling the Close service.

11.2 Administrator Guidance

Administrator guidance is provided in the Users Manual, which is delivered with the **Module**.

11.3 Non-Administrator Guidance

Non-Administrator guidance is provided in the Users Manual, which is delivered with the **Module**.

When using the **Module**, the library is only to be used in a single-threaded application. All API functions should be considered not thread-safe.

The **Module** provides the ability to generate a digital signature on a message that has been pre-hashed. Such signatures are only considered compliant if the hash was generated from the **Module** itself, or another module using an approved and CAVP-tested hash algorithm.

11.4 Design and Rules

The Module Initialization service automatically detects and enables PAA if available. After the **Module** is initialized, the PAA can be disabled by calling the Disable PAA service. This service can be called only after successful initialization of the Module. The Module Initialization service runs pre-operational and conditional self-tests and the Disable PAA service runs conditional self-tests.

The Close service zeroizes the state of internal DRBGs. It must be explicitly called by the CO before the **Module** is unloaded from the application using it. In addition, the CO can zeroize individual SSPs as described in Section 9.4.

Overall security design

- The Module provides one distinct operator role, namely Crypto Officer (CO).

- The Module does not provide authentication.
- The Module allows the operator to initiate power-up self-tests by power cycling power or resetting the Module.
- Pre-Operational self-tests do not require any operator action.
- Data output is inhibited during key generation, self-tests, zeroization, and error states.
- Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the Module.
- There are no restrictions on which keys or SSPs are zeroized by the zeroization service.
- The Module does not support concurrent operators.
- The Module does not support a maintenance interface or role.
- The Module does not support manual SSP establishment methods.
- The Module does not have any proprietary external input/output devices used for entry/output of data.
- The Module stores plaintext CSPs
- The Module does not output intermediate key values.
- The Module does not provide bypass services or ports/interfaces.

11.5 Maintenance Requirements

As a software module running on GPC, there are no specific requirements regarding maintenance.

The **Module** is disposed of by deleting the binary file of the **Module**.

11.6 End of Life

The **Module** does not persistently store SSPs. No sanitization procedure exists or is necessary.

12 Mitigation of Other Attacks

The **Module** is not designed to mitigate against other attacks.

References and Definitions

The following standards are referred to in this Security Policy.

Abbreviation	Full Specification Name
FIPS 140-3	Security Requirements for Cryptographic Modules, March 22, 2019
FIPS 140-3 IG	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program, October 7, 2022
SP 800-108r1	NIST Special Publication 800-108r1, Recommendation for Key Derivation Using Pseudorandom Functions (Revised), August 2022
SP 800-133r2	NIST Special Publication 800-133, Recommendation for Cryptographic Key Generation, Revision 2, June 2020
SP 800-135r1	National Institute of Standards and Technology, Recommendation for Existing Application-Specific Key Derivation Functions, Special Publication 800-135rev1, December 2011.
FIPS 186-5	National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-5, February 2023.
FIPS 198-1	National Institute of Standards and Technology, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198-1, July, 2008
FIPS 180-4	National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standards Publication 180-4, August, 2015
FIPS 202	FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, SHA3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, August 2015
FIPS 203	Federal Information Processing Standards Publication, Module-Lattice-Based Key-Encapsulation Mechanism Standard, August 2025
FIPS 204	Federal Information Processing Standards Publication, Module-Lattice-Based Digital Signature Standard, August 2024
SP 800-56Ar3	NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, April 2018
SP 800-56Cr2	NIST Special Publication 800-56C Revision 2, Recommendation for Key-Derivation Methods in Key-Establishment Schemes, August 2020
SP 800-90A	National Institute of Standards and Technology, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90A, Revision 1, June 2015.
RFC 5869	Internet Engineering Task Force specification of “HMAC-based Extract-and-Expand Key Derivation Function” by H. Krawczyk, P. Eronen, May 2010. Cited by [56Cr2].
RFC 8446	Internet Engineering Task Force specification of "The Transport Layer Security (TLS) Protocol Version 1.3“, https://datatracker.ietf.org/doc/html/rfc8446

Table 21: References

Acronym	Definition
AES	Advanced Encryption Standard
APT	Adaptative Proportion Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining (AES mode)
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSP	Critical Security Parameter
DKM	Derived Keying Material. Output of expansion step in two-step key derivation procedure
DRBG	Deterministic Random Bit Generator
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
FSM	Finite State Machine
GPC	General Purpose Computer
HKDF	HMAC-based Key Derivation Function
HMAC	Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Know Answer Test
KDA	Key Derivation Algorithm
KDF	Key Derivation Function
KDK	Key Derivation Key. Output from extraction step in two-step key derivation procedure
KEM	Key Encapsulation Mechanism
LTS	Long Term Support
MAC	Message Authentication Code
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
POSIX	Portable Operating System Interface
PSK	Pre-shared key as defined by [RFC8446]
PSP	Public Security Parameters
RAM	Random Access Memory
RCT	Repetition Count Test
SHA	Secure Hash Algorithm

Acronym	Definition
SSP	Sensitive Security Parameter
TLS	Transport Layer Security
TOEPP	Tested Operational Environment Physical Perimeter
XOF	eXtendable-Output Function

Table 22: Acronyms and Definitions