PQShield LTD

PQCryptoLib

Non-Proprietary FIPS 140-3 Security Policy

Document Version: v1.0.0

Date: 26 July 2024

Table of Contents

1. Generals 5
2. Cryptographic Module Specification 6
2.1 Operational Environment 6
2.2 Cryptographic Boundary 7
2.3 Modes of Operation 8
2.3.1 Configuration of the Approved mode of operation 8
2.3.2 Configuration of the non-Approved modes of operation 8
2.4 Security Functions 8
2.4.1 Industry protocols 15
2.5 Rules of Operation 15
3. Cryptographic Module Interfaces 17
4. Roles, Services and Authentication 18
4.1 Assumption of Roles and Related Services 18
4.2 Authentication Methods 21
4.3 Services 21
5. Software/Firmware Security 26
6. Operational Environment 27
7. Physical Security 28
8. Non-Invasive Security 28
9. Sensitive Security Parameter (SSP) Management 28
9.1 Sensitive Security Parameters (SSP) 29
9.2 DRBG Entropy Source 32
9.3 Zeroization of SSPs 33
10. Self-Tests 34
11. Life-Cycle Assurance 38
11.1 Installation, initialization, startup and operation of the Module 38
11.2 Maintenance requirements 38
11.3 Administrator and non-Administrator guidance 38
11.4 End of life procedure 38
12. Mitigation of Other Attacks 39

13. References and Definitions 40

List of Tables

Table 1 – Security Levels 5
Table 2– Tested Operational Environments 6
Table 3 Executable Code Sets - Software/Firmware/Hybrid 6
Table 4 – Vendor Affirmed Operational Environments 6
Table 5 - Approved Algorithms 8
Table 6 – Vendor Affirmed Algorithms 10
Table 7 – Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed 11
Table 8 - Security Function Implementations 11
Table 9 – Ports and Interfaces 17
Table 10 – Roles, Service Commands, Input and Output 18
Table 11 – Approved Services 21
Table 12 – Non-Approved Services 24
Table 13 – SSPs 29
Table 14 – Non-Deterministic Random Number Generation Specification 33
Table 15 – Error states and indicators 34
Table 16 – Pre-Operational Self-Test 34
Table 17 – Conditional Self-Tests 35
Table 18 – Service indicators 37
Table 19 – References 40
Table 20 – Acronyms and Definitions 41

List of Figures

Figure 1 - Module Block Diagram 7

1. General

This document defines the non-proprietary Security Policy for the PQShield PQCryptoLib 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.

The FIPS 140-3 security levels for the Module are as follows:

Table 1 – Security Levels

ISO/IEC 24759 section 6.	FIPS 140-3 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		1

2. Cryptographic Module Specification

The Module is classified as a software cryptographic module. It is a software library of cryptographic primitives with 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.

2.1 Operational Environment

A. Software module

PQCryptoLib cryptographic module is tested on the following operational environment.

Table 2 – Tested Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Ubuntu 20.04 LTS	Dell PowerEdge 740	Intel® Xeon® Platinum 8276 CPU (SkyLake)	with PAA
2	Ubuntu 20.04 LTS	Dell PowerEdge 740	Intel® Xeon® Platinum 8276 CPU (SkyLake)	without PAA

Table 3 – Executable Code Sets - Software/Firmware/Hybrid

#	Package/File Names	Software Version	Non-Security Relevant Distinguishing Features [Optional]	Hardware version if Hybrid [Optional]	Integrity Test Implemented
1	`libpqcrypto.so.1.0.0`	1.0.0	N/A	N/A	HMAC-SHA2-512

PQShield also performed testing of the module on the following Operational Environments:

Table 4 – Vendor Affirmed Operational Environments

#	Operating System	Hardware Platform
1	Debian 11 (bullseye)	Dell PowerEdge 740
2	Ubuntu 22.04 LTS	Dell PowerEdge 740

Operational Environment listed in Table 4 were not tested. No claim is made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

2.2 Cryptographic Boundary

The Module is a software library providing cryptographic services through application program interface (API) for use by Applications running in the user space of underlying operating system. The Module's embodiment is defined as *multi-chip standalone*.

The Figure 1 shows the cryptographic boundary of the Module, its interfaces with the tested operational environment's physical perimeter (TOEPP) and flow of information between the Module and operator (a calling function of an Application using services of the Module).

The software library is called `libpqcrypto.so.1.0.0` (software version 1.0.0) that is intended to link with the Application. The Module can run in a multi- threaded environment, it requires 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, tested operational environment and the CPU.

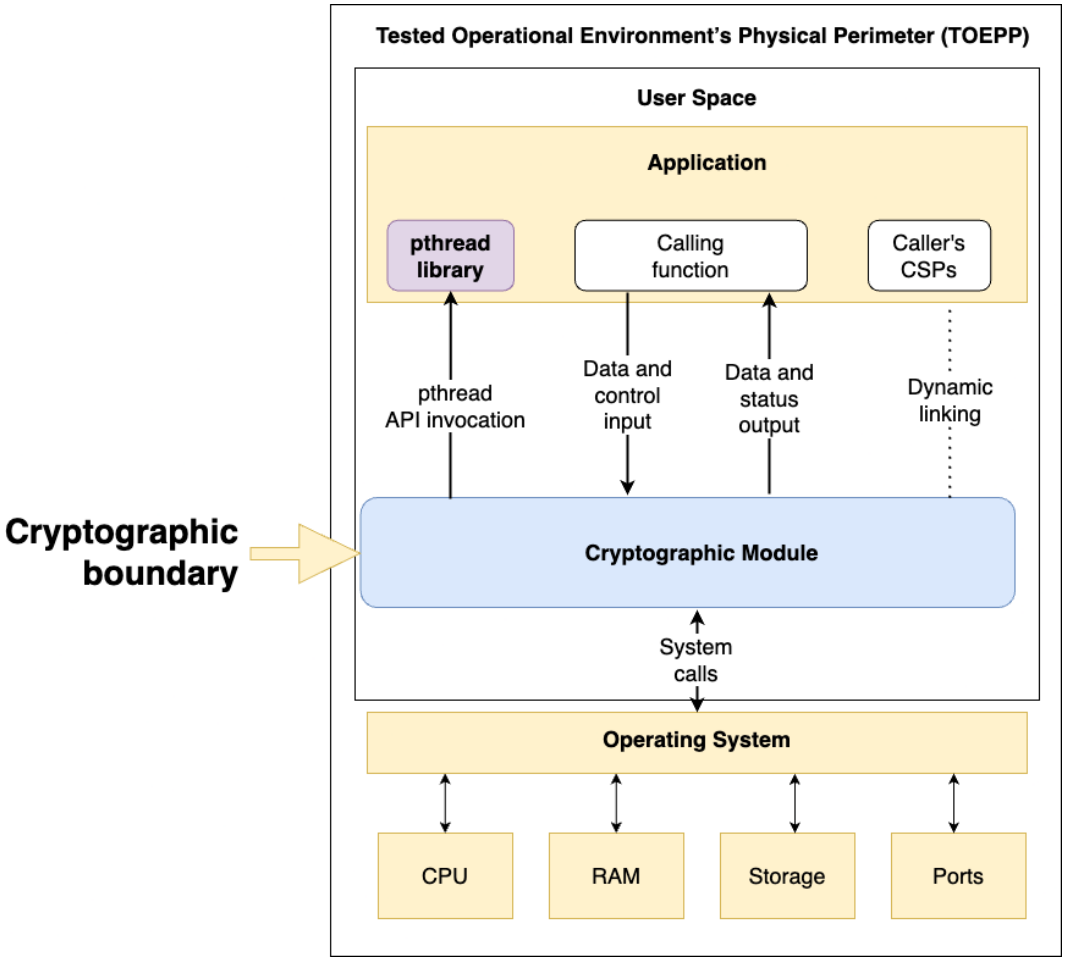

Figure 1 - Module Block Diagram:

Logical relationship of the Module to the other hardware and software components of the GPC.

2.3 Modes of Operation

The Module supports Approved mode of operation only. The Module does not support degraded mode and operates only in normal mode.

2.3.1 Configuration of the Approved Mode of Operation

The Approved mode of operation is configured after the CO loads the module into memory, all self-tests are completed successfully, and only Approved algorithms are invoked. See Table 5 below for the list of approved algorithms.

2.3.2 Configuration of the Non-Approved Modes of Operation

Non-approved mode of operation is not supported.

2.4 Security Functions

The Module implements the Approved and Non-Approved but Allowed cryptographic functions listed in the tables below.

Table 5 - Approved Algorithms

CAVP Cert	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use / Function
A3011	CVL: TLS [IG 2.4.B][1)]	TLS v1.3 KDF, DHE and PSK-DHE running modes	SHA2-(256, 384)	Key derivation
A3011	ECDSA [186]	Key pair generation	Curve: P-256, Security strength of 128-bits	Asymmetric Key Generation
A3011	ECDSA [186]	Public key verification	Curve: P-256, Security strength of 128-bits	Public Key Verification
A3011	ECDSA [186] [3)]	Signature generation Hash Algorithm: SHA2-256	Curve: P-256, Security strength of 128-bits	Signature generation
A3011	ECDSA [186] [3)]	Signature verification Hash Algorithm: SHA2-256	Curve: P-256, Security strength of 128-bits	Signature verification
A3011	Hash_DRBG [90A][2)]	Hash_DRBG [90A]	Security strength of 256 bits	Deterministic random bit generation
A3011	HMAC [198][4)]	SHA2-(224,256,384,512) SHA3-(224,256,384,512)	Security Strength of 192, 256 bits	Message authentication
A3011	KAS-ECC CDH-Component [56Ar3]	CDH-Component	Curve: P-256, Security strength of 128-bits	Shared secret computation

CAVP Cert	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use / Function
A3011	KAS-ECC-SSC [56Ar3] [5) 6)]	Ephemeral Unified	Curve: P-256, Security strength of 128-bits	Key generation, Shared secret computation
A3011	KDA HKDF [56Cr2]	SP 800-56Cr2 Section 5 HKDF [RFC5869] key derivation	Key sizes between 224 and 65336 (multiples of 8 bits)	Key based key derivation. Used by TLSv1.3 KDF
A3011	KDA Two-Step [56Cr2]	SP 800-56Cr2 Section 5 Two-step key derivation using KDF mode Feed-back	Key sizes between 224 and 65336 (multiples of 8 bits)	Key based key derivation.
A3011	KDF SP800-108 [108]	KDF SP800	HMAC SHA2 and SHA3 using 224, 256, 384 and 512 bits	Key based key derivation
A3011	SHA2 [180]	SHA2- (224,256,384,512)	Security strength of 112, 128, 192, 256 bits Message Length: 0-65536	Message digest generation
A3011	SHA3 [202]	SHA3- (224,256,384,512)	Security strength of 112, 128, 192, 256 bits Message Length: 0-65536 Large Message Sizes: 1GB	Message digest generation
A3011	SHAKE [202]	SHAKE-(128,256)	Max security strength either 128 or 256 bits	Variable size digest generation

1) No parts of this protocol, other than TLS v1.3 KDF, have been tested by the CAVP and CMVP. 2) 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. 3) ECDSA always uses SHA2-256 for hashing the input message 4) Truncated HMAC is CAVP-validated, but not used by the Module 5) ECDH implements ECC CDH primitive from [56Ar3] by using curve P-256. No KDF is applied on the output. 6) If run as a recipient, implementation performs partial public key validation compatible with an algorithm specified in [56Ar3], section 5.6.2.3.4.

Table 6 – Vendor Affirmed Algorithms

Algorithm	Algorithm Properties	OE	Reference
CKG	[133] Sections 4 and 5.1 Asymmetric signature key generation using unmodified DRBG output	Hardware Platform: Dell PowerEdge R740 Operating System: Ubuntu (version 20.04 LTS) Processor: Intel Xeon Platinum 8276	[133]
	[133] Sections 4 and 5.2 Asymmetric key establishment key generation using unmodified DRBG output	Hardware Platform: Dell PowerEdge R740 Operating System: Ubuntu (version 20.04 LTS) Processor: Intel Xeon Platinum 8276	[133]
	[133] Sections 4 and 6.1 Direct symmetric key generation using unmodified DRBG output	Hardware Platform: Dell PowerEdge R740 Operating System: Ubuntu (version 20.04 LTS) Processor: Intel Xeon Platinum 8276	[133]
	[133] Section 6.2.1 Derivation of symmetric keys from a key agreement shared secret	Hardware Platform: Dell PowerEdge R740 Operating System: Ubuntu (version 20.04 LTS) Processor: Intel Xeon Platinum 8276	[133]
	[133] Section 6.2.2 Derivation of symmetric keys from a pre-shared key	Hardware Platform: Dell PowerEdge R740 Operating System: Ubuntu (version 20.04 LTS) Processor: Intel Xeon Platinum 8276	[133]

NOTE: The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation.

Table 7 – Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

Algorithm	Caveat	Use / Function
Kyber KEM	No security claimed	Non-Approved cryptographic algorithm implementing non- security relevant service. This algorithm is meant to be used between two parties as a method to establish "*auxiliary shared secret T"* which is used for "hybrid" shared secret *Z'* as mentioned in [56Cr2, section 2]. The value Z' can only be used in the extraction step of the Two-Step Key Derivation process described in [56Cr2, section 5]. Internally, this step is implemented as HKDF-extract function as defined by [RFC5869]. Vendor claims no security over input values consumed by the algorithm and output values generated by the algorithm including value *T.* The algorithm uses Hash_DRBG for generating random byte strings. It also uses SHA3-{256,512} and SHAKE-{128,256} internally. The algorithm is not intended to be used as a security function and is not used internally by the Module. It must be accessed over API functions that are distinct from APIs used to access approved algorithms.

NOTE: The module does not implement any Non-Approved Algorithms Not Allowed in the Approved Mode of Operation.

Table 8 - Security Function Implementations

Name	Type	Description	SF Properties [Optional]	Algorithms	Algorithm Properties
CKG	Cryptographic Key Generation	Sections 4 and 6.1 Direct symmetric key generation using unmodified DRBG output	[133]	DRBG	Hash_DRBG
KAS1	KAS	Pair-wise Key-Establishment Schemes using Discrete Logarithm ECC Diffie-Hellman with TLS 1.3 KDF	KAS (KAS-SSC Cert. #A3011, CVL Cert. #A3011; SSP establishment methodology provides 128 bits of encryption strength), [56Ar3]1, [IG] 2.4.B, [IG] D.F	SHS DRBG KAS-SSC TLS 1.3 KDF	SHA2-256 SHA2-384 P-256 Curve Hash_DRBG

Name	Type	Description	SF Properties [Optional]	Algorithms	Algorithm Properties
KAS2	KAS	Pair-wise Key-Establishment Schemes using Discrete Logarithm ECC Diffie-Hellman with KDA HKDF	KAS (KAS-SSC Cert. #A3011, KDA Cert. #A3011; SSP establishment methodology provides 128 bits of encryption strength), [56Ar3]2, [56Cr2], [198], [IG] D.F, [IG] D.B, [133], [IG] C.C	HMAC SHA3 SHS KDA HKDF	P-256 Curve Hash_DRBG HMAC SHA2-224 HMAC SHA2-256 HMAC SHA2-384 HMAC SHA2-512 HMAC SHA3-224 HMAC SHA3-256 HMAC SHA3-384 HMAC SHA3-512
KAS3	KAS	Pair-wise Key-Establishment Schemes using Discrete Logarithm ECC Diffie-Hellman with KDA TwoStep	KAS (KAS-SSC Cert. #A3011, KDA Cert. #A3011; SSP establishment methodology provides 128 bits of encryption strength), [56Ar3]3, [56Cr2], [198], [IG] D.F, [IG] D.B, [133], [IG] C.C	HMAC SHA3 SHS KDA TwoStep	P-256 Curve Hash_DRBG HMAC SHA2-224 HMAC SHA2-256 HMAC SHA2-384 HMAC SHA2-512 HMAC SHA3-224 HMAC SHA3-256 HMAC SHA3-384 HMAC SHA3-512
KDA1	HKDF	Key Derivation Algorithm	[56Cr2] [198], [IG] C.C	HMAC SHA3 SHS KDA HKDF	HMAC SHA2-224 HMAC SHA2-256 HMAC SHA2-384 HMAC SHA2-512 HMAC SHA3-224 HMAC SHA3-256 HMAC SHA3-384 HMAC SHA3-512

Name	Type	Description	SF Properties [Optional]	Algorithms	Algorithm Properties
KDA2	TwoStep	Key Derivation Algorithm	[56Cr2] [198], [IG] C.C	HMAC SHA3 SHS KDA TwoStep	HMAC SHA2-224 HMAC SHA2-256 HMAC SHA2-384 HMAC SHA2-512 HMAC SHA3-224 HMAC SHA3-256 HMAC SHA3-384 HMAC SHA3-512
KDF1	Key Derivation Function TLS 1.3 KDF	Key Derivation using TLS 1.3 KDF	[IG] 2.4.B [IG] D.F [180]	SHS TLS 1.3 KDF	SHA2-256 SHA2- 384
KDF2	KDF [108]	Key Based Key Derivation Function using Feedback method	[108] [198] [IG] C.C	HMAC SHA3 SHS	KDF [108] – Feedback HMAC SHA2-224 HMAC SHA2-256 HMAC SHA2-384 HMAC SHA2-512 HMAC SHA3-224 HMAC SHA3-256 HMAC SHA3-384 HMAC SHA3-512
KeyGen	Key generation	ECDSA Key Generation	[186] [133]	ECDSA DRBG	Hash_DRBG P- 256 Curve

Name	Type	Description	SF Properties [Optional]	Algorithms	Algorithm Properties
KeyVer	Public Key Verification	ECDSA Public Key Verification	[186]	ECDSA	P-256 Curve
MAC	Message Authentication	Keyed-Hash Message Authentication Code	[198] [IG] C.B, [IG] C.C	HMAC SHS SHA3	HMAC SHA2-224 HMAC SHA2-256 HMAC SHA2-384 HMAC SHA2-512 HMAC SHA3-224 HMAC SHA3-256 HMAC SHA3-384 HMAC SHA3-512
Hash	Message Digest	Secure Hash Standard Permutation-Based Hash and Extendable-Output Functions	[180] [202] [IG] C.B [IG] C.C	SHS SHA3	SHA2-224 SHA2- 256 SHA2-384 SHA2-512 SHA3- 224 SHA3-256 SHA3-384 SHA3- 512 SHAKE-128 SHAKE-256
RNG	Deterministic Random Bit Generators	Random Number Generation	[90A] [IG] D.L [IG] D.R	DRBG	Hash_DRBG

Name	Type	Description	SF Properties [Optional]	Algorithms	Algorithm Properties
Shared Secret Establishment1	KAS ECC CDH-Component	Shared Secret Establishment Pair-wise Key-Establishment Scheme using Discrete Logarithm - ECC Diffie-Hellman	[56Ar3] [IG] D.F [IG] D.A [IG] D.B [133]	KAS ECC CDH DRBG	Hash_DRBG P- 256 Curve
Shared Secret Establishment2	KAS ECC -SSC	Shared Secret Establishment Pair-wise Key-Establishment Scheme using Discrete Logarithm - ECC Diffie-Hellman	[56Ar3] [IG] D.F [IG] D.B [133]	DRBG KAS-SSC	P-256 Curve Hash_DRBG
SigGen	Signature generation	Digital Signature Generation using ECDSA	[186] [133]	ECDSA DRBG SHS	Hash_DRBG P-256 Curve SHA2-256
SigVer	Signature verification	Digital Signature Verification using ECDSA	[186]	SHS	P-256 Curve SHA2-256

[1] Per [IG] D.F Scenario 2 path (2), [56Ar3] compliant key agreement scheme where testing is performed separately for the shared secret computation and a KDF compliant with [135] without key confirmation.

[2] Per [IG] D.F Scenario 2 path (2), [56Ar3] compliant key agreement scheme where testing is performed separately for the shared secret computation and a KDF compliant with KDA without key confirmation.

[3] Per [IG] D.F Scenario 2 path (2), [56Ar3] compliant key agreement scheme where testing is performed separately for the shared secret computation and a KDF compliant with KDA without key confirmation.

2.4.1 Industry Protocols

The Module supports KDF used by TLS protocol version 1.3.

2.5 Rules of Operation

The Module is intended to link with the Application. To initialize the Module, the code of an application that wishes to use the Module, must include `pqcl.h` and `fips.h` header files. This allows an application to use the *Module Initialization* service. The *Module Initialization* service must be used to initialize the Module. The Module is initialized after the *Module Initialization* service returns with success.

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 s*ervice runs conditional self-tests.

The *Zeroize* 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 method.
- 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.

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) and hence those ports are the same as those of the GPC it runs on.

The Module's ports and associated FIPS defined logical interface categories are listed in Table 9.

Table 9 – Ports and Interfaces

Physical port	Logical interface	Data that passes over port/interface
N/A	Data in	API input parameters that specify plaintext data; ciphertext or signed data; cryptographic keys, initialization vectors; kernel I/O.
N/A	Data out	API output parameters that receive plaintext data, ciphertext data, digital signatures cryptographic keys and initialization vectors. API Return values.
N/A	Control in	Function calls and control data (e.g., algorithms, algorithm modes, or module settings). Values stored in the CPU and read by the Module during initialization phase.
N/A	Control Out[1)]	N/A
N/A	Status out	API return values.
PC Power Supply Port	Power	N/A

1) Not supported by the Module.

All the services of the Module return either numeric values of type `pqcl_result_t` or nothing. The return values are defined and described in the header files of the module. The `PQCL_SUCCESS` is a return value indicates success and values prefixed with `PQCL_ERROR_` indicate some sort of failure. There is one exception, namely the *Status enquiry* service returns state of the library.

When the Module is performing self-tests, during zeroization or is in the error state, all output on the module's logical *data output* and interfaces are inhibited.

4. Roles, Services and Authentication

4.1 Assumption of Roles and Related Services

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.

Table 10 lists all operator roles supported by the Module and their related services.

Table 10 – Roles, Service Commands, Input and Output

Role	Service	Input	Output
CO	Disable PAA1	None	Status code
CO	Hash_DRBG operation context cleanup	Operation context. Algorithms: Hash_DRBG	Operation context with SSPs deleted
CO	Hash_DRBG pseudorandom byte stream generation	Operation context, output buffer, length of requested byte stream, optionally buffer with additional data. Algorithms: Hash_DRBG	Stream of pseudo-random bytes, status code
CO	Hash_DRBG reseeding	Operation context, buffers with entropy, nonce and personalization string together with sizes of those buffers, optionally buffer with additional data. Algorithms: Hash_DRBG	Status code
CO	Hash_DRBG instantiation and seeding	Operation context, buffers with entropy, nonce and personalization string together with sizes of those buffers. Algorithms: Hash_DRBG	Status code
CO	Hashing	Input data, input size. Hash functions: SHA2-(224, 256, 384, 512), SHA3-(224, 256, 384, 512).	Digest, status code.
CO	KEM context cleanup	Operation context. Algorithms: Kyber (non-approved)	Operation context with SSPs deleted
CO	KEM decapsulation	Operation context with KEM private key, ciphertext with byte length. Algorithms used: Kyber (non-approved)	Shared secret, status code.
CO	KEM encapsulation	Operation context with KEM public key. Algorithms used: Kyber (non-approved)	Shared secret with byte length, ciphertext with byte length, status code
CO	KEM key-pair export	Operation context, memory buffers to store public and/or private key. Algorithms: Kyber (non-approved)	Public and/or private key stored in memory buffer, status code.

Role	Service	Input	Output
CO	KEM key-pair generation	Operation context. Algorithms: Kyber (non-approved)	Key-pair stored in operation context. Status code.
CO	KEM key-pair import	Operation context, public key and/or private key in plaintext. Algorithms: Kyber (non-approved)	Key-pair stored in operation context, Status code.
CO	Key agreement	Operation context, peers public key with byte length. Algorithms. ECDH (curve P256).	Shared secret stored in a memory buffer, status code
CO	Key agreement key-pair cleanup	Operation context. Algorithms: ECDH (curve P-256).	Operation context with SSPs deleted
CO	Key agreement key-pair export	Operation context, memory buffers to store public and/or private key. Algorithms: ECDH (curve P-256).	Public and/or private key stored in memory buffer, status code.
CO	Key agreement key-pair generation	Operation context. Algorithms: ECDH (curve P-256)	Key-pair stored in operation context. Status code.
CO	Key agreement key-pair import	Operation context, public key and/or private key in plaintext. Algorithms: ECDH (curve P-256).	Key-pair stored in operation context, Status code.
CO	Key derivation	Operation context, key derivation key, salt, shared secret established by approved and auxiliary shared secret (56Cr2), algorithm ID, optional context binding value. Algorithms used: HKDF	Derived key, status code.
CO	Key expansion in two-step key derivation	Operation context, key-derivation key and its size, info label and its size. Algorithms used: KDF [108]	Derived key, status code.
CO	Keyed hash context cleanup	Operation context. Algorithms: HMAC	Operation context with SSPs deleted
CO	Keyed hash key import	Operation context, buffer containing key and its size. Key sizes: 112-bits or more (multiple of 8 bits). Algorithms: HMAC	Status code
CO	Keyed hash signing	Operation context, input data, input size, buffer storing output and output size. Algorithms: HMAC	An authentication tag, status code
CO	Keyed hash verification	Operation context, input data, input size. Algorithms: HMAC	Status code
CO	Module initialization [2)]	None	Status code
CO	Name enquiry	Pointer to the buffer	Text "pqcryptolib"

Role	Service	Input	Output
CO	Random number generation	Entropy, output size, additional input. Algorithms used: Hash_DRBG.	Stream of randomly generated bytes. Security strength of 256-bits.
CO	Randomness extraction	Operation context, shared secret established by approved key agreement method (56Cr2) and its size, salt value and its size. Algorithms used: HKDF	Key-derivation key, status code
CO	Randomness extraction with auxiliary shared secret	Operation context, shared secret established by approved key agreement method (56Cr2) and its size, salt value and its size, auxiliary shared secret and its size. Algorithms used: HKDF	Key-derivation key, status code
CO	Signature generation	Operation context, message to sign with byte length, memory buffer. Algorithms: ECDSA (curve P-256)	Signature stored in the memory buffer. Status code
CO	Signature verification	Operation context, message to verify with byte length. Algorithms: ECDSA (curve P-256)	Operation context with SSPs deleted
CO	Signing key-pair cleanup	Operation context. Algorithms: ECDSA (curve P-256).	Operation context with SSPs deleted
CO	Signing key-pair export	Operation context, memory buffers to store public and/or private key. Algorithms: ECDSA (curve P-256).	Public and/or private key stored in memory buffer. Status code.
CO	Signing key-pair generation	Operation context. Algorithms: ECDSA (curve P-256)	Key-pair stored in operation context. Status code.
CO	Signing key-pair import	Operation context, public key and/or private key in plaintext. Algorithms: ECDSA (curve P-256).	Key-pair stored in operation context. Status code.
CO	Status enquiry	None	Current state of the module
CO	Symmetric key generation	Buffer requested key size in bits. Algorithms: Hash_DRBG	Symmetric key, status code
CO	TLS v1.3 KDF	Operation context, key derivation key and its size, shared secret and authentication messages as defined by RFC8446, optionally pre-shared key and its size	Status code and all traffic and handshake keys resulting from TLS v1.3 key derivation
CO	Version enquiry	Pointer to the buffer	Text "1.0.0"
CO	XOF	Output size. Algorithms: SHAKE-128, SHAKE-256	Output generated by XOF.
CO	Zeroize	None	None

1) Ensures PAA is disabled and runs set of self-tests. 2) Initializes internal structures, enables PAA if available. It runs integrity checks, critical function testing and set of self-tests.

4.2 Authentication Methods

The Module does not support authentication.

4.3 Services

All security services implemented by the Module are listed in Table 11 and Table 12 below.

The SSPs modes of access are defined as:

- **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 updated, 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

The "Indicator" column describes security service indicator (described in [IG] 2.4.C) used by the service. All the indicators are defined below in Table 18.

Table 11 – Approved Services

Service	Description	Approved Security Functions	Keys and /or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Disable PAA	Switch off PAA	N/A	N/A	CO	N/A	N/A
Hash_DRBG operation context cleanup	Zeroize Hash_DRBG SSPs	Hash_DRBG	DRBG-S	CO	Z	DSI
Hash_DRBG pseudorandom byte stream generation	Generate pseudo random byte strings and keys	Hash_DRBG	DRBG-S	CO	ERW	DSI
Hash_DRBG reseeding	Seed Hash_DRBG instance with externally supplied entropy	Hash_DRBG	DRBG-EI, DRBG-N, DRBG-S, DRBG-SEED	CO	W, R, R, GW	DSI
Hash_DRBG instantiation and seeding	Seed Hash_DRBG instance with externally supplied entropy	Hash_DRBG	DRBG-EI, DRBG-N, DRBG-S, DRBG-SEED	CO	W, R, R, GW	DSI

Service	Description	Approved Security Functions	Keys and /or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Hashing	Calculate a message digest	SHA2-(224,256,384,512) SHA3-(224,256,384,512)	N/A	CO	N/A	HSI
Key agreement	Calculate shared secret key from private and public keys	ECDH	ECDH-PRV, ECDH-PUB, ECDH-K-Z	CO	ER, ER, GW	XSI
Key agreement key-pair cleanup	Zeroize SSPs	ECDH	ECDH-PRV, ECDH-PUB	CO	Z, Z	XSI
Key agreement key-pair export	Export a public and/or private ECDH key(s)	ECDH	ECDH-PRV, ECDH-PUB	CO	R, R	XSI
Key agreement key-pair generation	Generate a key-pair for key agreement	ECDH	ECDH-PRV, ECDH-PUB	CO	EGW, GW,	XSI
Key agreement key-pair import	Import a public and/or private ECDH key(s)	ECDH	ECDH-PRV, ECDH-PUB	CO	W, W	XSI
Key derivation	Symmetric key derivation from shared secret key	KDA-HKDF SHA2-(224,256,384,512) SHA3-(224,256,384,512), CKG	HKDF-K, HKDF-KDK	CO	ER, EGWZ	KSI
Key expansion in two-step key derivation	PRF-based key-derivation function [108]	KDA-HKDF, KDF-SP800-108 SHA2-(224,256,384,512) SHA3-(224,256,384,512), CKG	HKDF-KDK	CO	RE	KSI

Service	Description	Approved Security Functions	Keys and /or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Keyed hash context cleanup	Zeroize HMAC SSPs	SHA2-(224,256,384,512) SHA3-(224,256,384,512)	HMAC-K	CO	Z	MSI
Keyed hash key import	Import a HMAC symmetric key into operation context	HMAC, SHA2-(224,256,384,512), SHA3-(224,256,384,512)	HMAC-K	CO	ER	MSI
Keyed hash signing	Calculate an authentication tag on data	HMAC, SHA2-(224,256,384,512) , SHA3-(224,256,384,512)	HMAC-K	CO	ER	MSI
Keyed hash verification	Validate an authentication tag on data	HMAC, SHA2-(224,256,384,512), SHA3-(224,256,384,512)	HMAC-K	CO	ER	MSI
Module initialization	Initializes internal structure of the Module	Hash_DRBG HMAC	DRBG-EI, DRBG-SEED, DRBG-S, DRBG-N	CO	RE, WE, W, W	N/A
Name enquiry	Returns name of the Module	N/A	N/A	CO	N/A	N/A
Random number generation	Generate random byte strings and keys from the Module's internal instance of Hash_DRBG	Hash_DRBG	DRBG-S	CO	ERW	RSI
Randomness extraction	The randomness extraction step in two- step key derivation procedure as specified by [56Cr2]	KDA-HKDF SHA2-(224,256,384,512) SHA3-(224,256,384,512)	HKDF-K, HKDF-KDK	CO	ER, GW	KSI

Service	Description	Approved Security Functions	Keys and /or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Randomness extraction with auxiliary shared secret	The randomness extraction step in two- step key derivation procedure with auxiliary shared secret T as specified by [56Cr2]	KDA-HKDF SHA2-(224,256,384,512) SHA3-(224,256,384,512)	HKDF-K, HKDF-KDK	CO	ER, GW	KSI
Signature generation	Calculate digital signature using a private key	SHA2-(256), ECDSA	ECDSA-PRV	CO	RE	SSI
Signature verification	Verify digital signature using a public key	SHA2-(256), ECDSA	ECDSA-PUB	CO	RE	SSI
Signing key-pair cleanup	Zeroize SSPs and deletes operation context	ECDSA	ECDSA-PRV, ECDSA-PUB	CO	Z, Z	SSI
Signing key-pair export	Export a public and/or private ECDSA key(s)	ECDSA	ECDSA-PRV, ECDSA-PUB	CO	R, R	SSI
Signing key-pair generation	Generate a key-pair for signing and verification	ECDSA	ECDSA-PRV, ECDSA-PUB	CO	EGW, GW	SSI
Signing key-pair import	Import a public and/or private asymmetric key into operation context	ECDSA	ECDSA-PRV, ECDSA-PUB	CO	W, W	SSI
Status enquiry	Returns state of the Module	N/A	N/A	CO	N/A	N/A

Service	Description	Approved Security Functions	Keys and /or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Symmetric key generation	Generates symmetric key of size between 112 and 256 bits, multiple of 8.	Hash_DRBG, CKG	HMAC-K	CO	GW	DSI
TLS v1.3 KDF	Expands key-derivation- key into secret key as specified by the RFC8446	TLS KDF, CKG	ECDH-K-Z, TLS-PSK, TLS-ES, TLS-HS, TLS-MS, TLS-EBK, TLS-RBK, TLS-CETS, TLS-EEMS, TLS-CHTS, TLS-SHTS, TLS-CATS, TLS-SATS, TLS-EMS, TLS-RMS, TLS-ZERO	CO	ER, ER, GW, GW, GW, GW, GW, GW, GW, GW, GW, GW, GW, GW, ER	TSI
Version enquiry	Returns version of the Module	N/A	N/A	CO	N/A	N/A
XOF	Calculate output from XOF	SHAKE-(128, 256)	N/A	CO	N/A	HSI
Zeroize	Destroys internal state of Hash_DRBG	Hash_DRBG	DRBG-S	CO	Z	ZSI

The following table contains non-approved, allowed services running in approved mode of operation. All those services implement Key Encapsulation Mechanism but are non-security relevant. Namely we do not declare any security on the keys generated by those algorithms.

Table 12 – Non-Approved Services

Service	Description	Algorithm Accessed	Role	Indicator
KEM context cleanup	Non-security relevant service	None	CO	ESI
KEM decapsulation	Non-security relevant service	SHA3-{256,512}, SHAKE-{128, 256}, Kyber	CO	ESI
KEM encapsulation	Non-security relevant service	Hash_DRBG, SHA3-{256,512}, SHAKE-{128, 256}, Kyber	CO	ESI
KEM key-pair export	Non-security relevant service	Kyber	CO	ESI
KEM key-pair generation	Non-security relevant service	Hash_DRBG, SHA3-{256,512}, SHAKE-{128, 256}, Kyber	CO	ESI
KEM key-pair import	Non-security relevant service	Kyber	CO	ESI

5. Software/Firmware Security

The Module is composed of the software component delivered as a library (dynamic loadable shared object library) in a binary form. The software component is protected with the HMAC-SHA2-512 integrity testing technique described in **Table 17**. The integrity tests are always performed upon module initialization phase (described in §2.5) and can be performed on demand by power cycling the Module. In case the integrity test fails, the library is moved into ERROR state and the Module needs to be unloaded from the memory of the Application that uses the Module.

The method used for integrity testing is detailed in Table 17.

6. Operational Environment

The Module operates under a modifiable operational environment as per the FIPS 140-3 definitions. The tested operational environment is listed in Table 2 above. In addition, PQShield claims that the Module can be ported on the operational environment listed in Table 4; no statement is made regarding the correct operation of the Module on the Vendor Affirmed Operational Environments. The Module runs on a GPC running one of the tested operational environments. Each tested 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 Parameter (SSP) Management

The SSPs generation column shown in **Table 13** are defined as:

- G1 = FIPS 186-4 compliant ECDSA key generation by testing candidate methods described in [186], subsection B.4.2.
- G2 = Symmetric key generated by internal CAVP validated Hash_DRBG
- G3 = Generated by the entropy source
- G4 = Derived from internal state of Hash_DRBG
- G5 = Generated by Hash_df hash derivation function of Hash_DRBG
- G6 = KDK generated in two-step key derivation with HMAC used as PRF [56Cr2]
- G7 = DKM generated by TLS v1.3 KDF [RFC8446]
- G8 = Counter initialized to 0 and incremented during execution of an algorithm
- G9 = Value modified during initialization and execution of the module

The SSPs establishment column shown in **Table 13** are defined as:

- A1 = SP800-56A rev3 compliant ECDH key agreement (128 bit)
- A2 = Derived using SP 800-108 compliant KDF

The SSPs storage column shown in **Table 13** are defined as:

- S1 = Only stored in volatile memory (RAM) in plaintext
- S2 = Publicly known value stored within the module code, validated by integrity check

The SSPs import/export column shown in **Table 13** are defined as:

- E1 = Input in plaintext by the calling application
- E2 = Public key output in plaintext
- E3 = Secret key output in plaintext

The SSPs zeroization column shown in **Table 13** are defined as:

- Z1 = Zeroized by Module power cycle or hard reset.
- Z2 = Zeroized by the internal zeroization function by overwriting with a fixed pattern, that is octet-string filled with 0.
- Z3 = Overwriting by new working state of Hash_DRBG during reseeding process

9.1 Sensitive Security Parameters (SSP)

All SSPs used by the Module are described in this section. All usage of these SSPs by the Module is described in the services detailed in 4.3.

Table 13 – SSPs

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use	Related keys
DRBG-EI	256	N/A	G3	E1	N/A	S1	Z1 Z2	Hash_DRBG [90A] entropy input	DRBG-S, DRBG-N
DRBG-SEED	256	DRBG (#A3011)	G5	N/A	N/A	S1	Z1 Z2 Z3	Hash_DRBG [90A] initialization and reseeding	DRBG-EI DRBG-N DRBG-S
DRBG-S	256	DRBG (#A3011)	G4	N/A	N/A	S1	Z1 Z2	Hash_DRBG [90A] working state (values V and C) derived from the seed	DRBG-EI DRBG-SEED
DRBG-N	256	DRBG (#A3011)	G3	E1	N/A	S1	Z1 Z2	Hash_DRBG [90A] nonce	DRBG-EI DRBG-SEED
DRBG-AI	128	DRBG (#A3011)	N/A	E3	N/A	S1	Z1 Z2	DRBG additional input. When used for ECDSA signature generation, the input is a ECDSA secret key.	ECDSA-PRV
HMAC-K	Between 128 and 256. Multiple of 8 bits	HMAC (#A3011)	G2	E1	N/A	S1	Z1 Z2	HMAC [198] authentication key and OPAD value	DRBG-S
HKDF-K	Between 224 and 65536. Multiple of 8 bits	KDA-HKDF, Two-Step (#A3011)	G2	E1	A1	S1	Z1 Z2	KDA-HKDF secret key	DRBG-S

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use	Related keys
HKDF-KDK	Between 8 and 4096. Multiple of 8 bits	KDA-{HKDF, Two-Step} (#A3011)	N/A	E3	A2	S1	Z1 Z2	Key-derivation key resulting from the randomness-extraction step that is used in the key-expansion step during the execution of the key-derivation procedure specified in the [56Cr2] and/or [108]	HMAC-K ECDH-K-PRV HKDF-CNT
ECDSA-PRV	128	ECDSA (#A3011)	G1	E1 E3	N/A	S1	Z1 Z2	ECDSA signature generation key (P-256)	DRBG-S DRBG-N DRBG-AI
ECDH-PRV	128	EC Diffie Hellman Shared Secret Computation (#A3011)	G1	E1 E3	N/A	S1	Z1 Z2	ECDH key agreement private key (P-256)	DRBG-S DRBG-N ECC-PAR
ECDH-K-Z	128	EC Diffie Hellman Shared Secret Computation (#A3011)	N/A	E1 E3	A1	S1	Z1 Z2	ECDH shared secret (P-256) used to derive session encryption key	ECDH-K-PRV ECC-PAR
TLS-PSK	Between 112 and 256. Multiple of 8 bits	TLS v1.3 KDF (#A3011)	G6	E1	N/A	S1	Z1 Z2	A TLS v1.3 pre-shared key, established externally or derived from TLS-RMS.	TLS-RMS
TLS-ES	256 or 384 bits	TLS v1.3 KDF (#A3011)	G6	N/A	N/A	S1	Z1 Z2	TLS v1.3 early secret	TLS-PSK ECDH-K-Z
TLS-HS	256 or 384 bits	TLS v1.3 KDF (#A3011)	G6	N/A	N/A	S1	Z1 Z2	TLS v1.3 handshake secret	TLS-PSK ECDH-K-Z

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use	Related keys
TLS-MS	256 or 384 bits	TLS v1.3 KDF (#A3011)	G6	N/A	N/A	S1	Z1 Z2	TLS v1.3 master secret	TLS-PSK ECDH-K-Z
TLS-EBK	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 binder secret for external PSKs	TLS-ES
TLS-RBK	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 binder key for resumption PSKs	TLS-ES
TLS- CETS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 early traffic secret	TLS-ES
TLS- EEMS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 early master secret	TLS-ES
TLS- CHTS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 client handshake traffic secret	TLS-HS
TLS- SHTS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 server handshake traffic secret	TLS-HS
TLS- CATS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 client application traffic secret	TLS-MS
TLS- SATS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 server application traffic secret	TLS-MS
TLS-EMS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 exporter master secret	TLS-MS
TLS- RMS	128	TLS v1.3 KDF (#A3011)	G7	E3	N/A	S1	Z1 Z2	TLS v1.3 resumption master secret	TLS-MS
GLOB	N/A	N/A	G9	N/A	N/A	S1	Z1	Global state of the cryptographic module	N/A

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use	Related keys
HKDF- CNT	N/A	KDA-{HKDF, Two-Step} (#A3011)	G8	N/A	N/A	S1	Z1 Z2	Counter used by KDF in feedback mode [108]	HKDF-KDK
ECC-PAR	N/A	ECDSA and EC Diffie Hellman Shared Secret Computation (#A3011)	N/A	N/A	N/A	S2	Z1	ECC domain parameters	ECDSA-PUB ECDH-PUB ECDSA-PRV ECDH-PRV
ECDSA-PUB	128	ECDSA (#A3011)	G1	E1, E2	N/A	S1	Z1 Z2	ECDSA signature verification key (P-256)	ECDSA-PRV ECC-PAR
ECDH-PUB	128	EC Diffie Hellman Shared Secret Computation (#A3011)	G1	E1, E2	N/A	S1	Z1 Z2	ECDH key agreement public key (P-256)	ECDH-PRV ECC-PAR
TLS-ZERO	N/A	TLS v1.3 KDF (#A3011)	G9	N/A	N/A	S1	Z1	Internal value used by TLS v1.3 KDF in case PSK is not provided. 48-byte long buffer filled with 0.	All TLS-* SSPs, except TLS-PSK

9.2 DRBG Entropy Source

The RNG module leverages two different entropy sources, one provided by the CPU and the other provided by operational environment. The RDSEED is a CPU instruction, used to get entropy directly from the Intel CPU. Additionally, the Module uses the entropy from environmental noise. This is done by using `getrandom` system call. Returned bytes from both sources are XORd together and provided to the RNG initialization function as an entropy. In case 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 approved mode operation. The module conforms to FIPS 140-3 IG 9.3.A scenario 2b, thus the following caveat is applicable:

No assurance of the minimum strength of generated SSPs (e.g., keys).

Table 14 – Non-Deterministic Random Number Generation Specification

Entropy Sources	Minimum number of entropy bits	Details
RDSEED	256	The RDSEED is CPU instruction, that provides an access to the implementation of XOR-NRBG construction (as per [90C]). It uses AES/128-CBC-MAC conditioner (vetted conditioning component as per [90B]) and internal implementation of DRBG based on AES-CTR to produce full entropy output. RDSEED is used as the main entropy source, which provides 256 bits of entropy to seed DRBG. Construction is CAVP certified (Cert. #A1791).
getrandom()	8	Additional entropy source.

9.3 Zeroization of SSPs

SSPs are zeroized when the appropriate operation context is destroyed, after *Zeroize* service has completed an operation or on power cycling/reboot of host platform. Input and output interfaces are inhibited while zeroization is performed. The ZSI indicator must be used to check if zeroization was successful.

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.

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. The module uses critical functions, namely hash-based DRBG and HMAC-SHA2-512. The critical functions are tested both during pre-operational and conditional testing run. There is only one self-tests error state, and it is described in the table below:

Table 15 – Error States and Indicators

Error state	Description	Indicator
ES1	The Module fails a KAT or PCT self-test. The module does not perform any cryptographic functions and all data output is inhibited in the error state. The Module needs to be power cycled to clear the error.	The Module enters the ERROR state and outputs status of PQCL_STATE_ERROR

The Module performs the following pre-operational self-tests:

Table 16 – Pre-Operational Self-Test

Security Function	Method	Type	Description	Error state
DRBG	KAT	Critical function test	DRBG Critical Function Tests (Instantiate, Generate and Reseed)	ES1
HMAC	Software integrity	Software integrity test	Integrity check of cryptographic module, using HMAC with SHA2-512 with fixed 256-bit key over various continuous segments of the Module binary image; tag compared against reference value stored in the binary	ES1

The Module performs the following conditional self-tests:

Table 17 – Conditional Self-Tests

Security Function	Method	Type	Description	Error state
DRBG	KAT	CAST	Uses Hash_DRBG based on SHA2-256 for 256-bit security strength. Includes instantiate, generate, generate with additional input, reseed and reseed with additional input KATs. Doesn't include prediction resistance. Performed before the first random data generation. Self-test is performed before being used in the Integrity test.	ES1
ECDH Key Generation	PCT	PCT	ECDH P-256 Key Generation Pairwise Consistency Test	ES1
ECDSA	KAT	CAST	ECDSA P-256 with SHA2-256 signature generation and verification. Uses 32-byte long message.	ES1
ECDSA Key Generation	PCT	PCT	ECDSA P-256 Key Generation Pairwise Consistency Test	ES1
HMAC	KAT	CAST	HMAC-SHA2-512 KAT with 16-byte long key over 32-byte long message. Self-test is performed before being used in the Integrity test.	ES1
HMAC	KAT	CAST	HMAC-SHA2-512 KAT with 16-byte long key over 32-byte long message.	ES1
KAS-SSC	KAT	CAST	ECC Diffie-Hellman shared secret generation with P-256 as per [IG] D.F	ES1

Security Function	Method	Type	Description	Error state
KDF [108]	KAT	CAST	KDF with HMAC-SHA2-256 is used as PRF in feedback mode with an 8-bit counter located after fixed data, an 8-byte long Label string, and a 32-byte long IV set to buffer filled with random data.	ES1
KDA HKDF	KAT	CAST	HKDF with HMAC-SHA2-256 as an auxiliary function. An input is a 32-byte long value Z, 16-byte long salt and 52-byte long info string as described by the [56Cr2]. Output (DKM) is a 32-byte long derived key.	ES1
KDA Two-Step	KAT	CAST	Two-step KDF with HMAC-SHA2-384 as an auxiliary function and concatenation of shared Z and auxiliary secret T as described by the [56Cr2]. An input to the extract step is a 32-byte long value Z and a 16-byte value T. Extraction uses a 128-byte long salt filled with 0. It produces a 48-byte-long KDK. An expansion step uses feedback mode with 16-byte long IV and it does not use counter. An input to the expansion step is a KDK and info string. Info string is formatted as a concatenation of two, 16 bytes long buffers, value T, and a 4-byte long buffer containing a length of output in bits. Output (DKM) is a 32-byte long key.	ES1
SHA3/SHAKE	KAT	CAST	SHAKE128 using 16-byte message. Two separated self-tests with the same parameters run for AVX2 optimized and non- optimized implementation.	ES1
SHS	KAT	CAST	SHA2-256 KAT using 32-byte message.	ES1
TLS KDF v1.3	KAT	CAST	Uses random 34-byte long shared secret Z (as defined in the [56Cr2]) and ClientHello, ServerHello, client Finished and server Finished (as defined in the [TLSACVP] and [RFC8446]) – all 34 byte-long. All key schedule secrets are generated and validated against the expected value.	ES1

The Module provides the operator dedicated query function to determine whether the current security service in use is approved. Security services implemented by the Module are grouped by type. The Module provides one query function for each type of service. Details of usage are described in the product documentation.

Table 18 defines a mapping between the identifier of the service indicator (used in the tables above) and the type of service.

Table 18 – Service Indicators

Service Indicator	Type of security service
DSI	Deterministic Random Number Generation
HSI	Hashing
MSI	Message Authentication Code
KSI	Key Derivation Function
ESI	Key Encapsulation Mechanism
XSI	Key Agreement
RSI	Random number generation
SSI	Digital Signature
TSI	TLS v1.3 KDF
ZSI	Zeroization

11. Life-Cycle Assurance

11.1 Installation, Initialization, Startup and Operation of the Module

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. The library is delivered in a tarball file, that contains linkable software library and the API declared in the header files, as well as documentation.

To install the library, an operator needs to unpack the tarball to target directory of its choice. Following command should be used:

```
mkdir -p /opt/pqshield/pqcryptolib
tar xvf pqcryptolib_v1.0.0.0_amd64_linux.tar.gz \
      -C /opt/pqshield/pqcryptolib
```

An application that wishes to use the Module includes `pqcl.h` and `fips.h` header files in its source code and link the application against 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 *Zeroization* service.

The initialization process automatically detects and enables PAA if available. To disable PAA operator must call the pqcl_disable_hwa() service. The service can be called only after successful initialization of the library.

11.2 Maintenance Requirements

This software module has no specific requirements regarding maintenance. The module is disposed by deleting the binary file of the Module.

11.3 Administrator and Non-Administrator Guidance

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

11.4 End of life Procedure

When the Module is at end of life, the customers of the Module are informed via PQShield's customer service capabilities. After the 6-month window, the access rights to the FIPS branch in a repository storing the source code of the Module are change to more restrictive, so that only administrators can access and read from the FIPS branch of the Module. This effectively makes it impossible to release new version of the Module.
The Module does not possess persistent storage of SSPs. The SSP value only exists in volatile memory and that value vanishes when the Module is powered off. The secure sanitization of the Module is done by powering the Module off. The deprecation of the Module is done by upgrading it to the newer version. During upgrade process the old version of the Module is removed and replaced with a new version.

12. Mitigation of Other Attacks

The Module is not designed to mitigate against other attacks.

13. References and Definitions

The following standards are referred to in this Security Policy.

Table 19 – References

Abbreviation	Full Specification Name
[FIPS140-3]	*Security Requirements for Cryptographic Modules*, March 22, 2019
[ISO19790]	*International Standard, ISO/IEC 19790, Information technology — Security techniques — Test requirements for cryptographic modules, Third edition, March 2017*
[ISO24759]	*International Standard, ISO/IEC 24759, Information technology — Security techniques — Test requirements for cryptographic modules, Second and Corrected version, 15 December 2015*
[IG]	*Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program, October 7, 2022*
[108]	*NIST Special Publication 800-108r1, Recommendation for Key Derivation Using Pseudorandom Functions (Revised), August 2022*
[133]	*NIST Special Publication 800-133, Recommendation for Cryptographic Key Generation, Revision 2, June 2020*
[135]	*National Institute of Standards and Technology, Recommendation for Existing Application-Specific Key Derivation Functions, Special Publication 800-135rev1, December 2011.*
[186]	*National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-4, July 2013.*
[198]	*National Institute of Standards and Technology, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198-1, July, 2008*
[180]	*National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standards Publication 180-4, August, 2015*
[202]	*FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, SHA3 Standard: Permutation-Based Hash and Extendable-Output Functions,* FIPS PUB 202, August 2015
[56Ar3]	*NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography,* April 2018
[56Br2]	*NIST Special Publication 800-56B Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Finite Field Cryptography, March 2019*
[56Cr2]	*NIST Special Publication 800-56C Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, August 2020*

Abbreviation	Full Specification Name
[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.
[90B]	*National Institute of Standards and Technology, Recommendation for the Entropy Sources Used for Random Bit Generation, Special Publication 800-90B, January 2018.*
[90C]	*Recommendation for Random Bit Generator (RBG) Constructions (2nd Draft),* Special Publication 800-90C (2nd Draft), April 2016
[RFC5869]	*Internet Engineering Task Force specification of "HMAC-based Extract-and-Expand Key Derivation Function"* by H. Krawczyk, P. Eronen, May 2010. Citated by [56Cr2].
[TLSACVP]	*"ACVP TLS Key Derivation Function JSON Specification",* https://pages.nist.gov/ACVP/draft-hammett-acvp-kdf-tls-v1.3.html
[RFC8446]	Internet Engineering Task Force specification of *"The Transport Layer Security (TLS) Protocol Version 1.3", https://datatracker.ietf.org/doc/html/rfc8446*

Table 20 – Acronyms and Definitions

Acronym	Definition
APT	Adaptative Proportion Test
DKM	Derived Keying Material. Output of expansion step in two-step key derivation procedure
GPC	General Purpose Computer
KAT	Know Answer Test
KDK	Key Derivation Key. Output from extraction step in two-step key derivation procedure
KEM	Key Encapsulation Mechanism
PCT	Pair-wise Consistency Test
POSIX	Portable Operating System Interface
PSK	Pre-shared key as defined by [RFC8446]
RCT	Repetition Count Test
SSP	Sensitive Security Parameter