Advanced Product Design Sdn Bhd (737874-A)

Biocryptodisk BEncryptorLib Ver 6.2

MYCV Non-Proprietary Security Policy

Version 1.7

Oct 10, 2025

Table of Contents

List of Tables

List of Figures

Author(s)	Title	Date	Version	Description
Lee Kong Pheng	CEO	Apr 11, 2025	1.0	Initial Release
PQ	Software Engineer	May 20, 2025	1.1	Address Comment & formatting
PQ	Software Engineer	Jun 20, 2025	1.2	Address Comment & formatting
PQ	Software Engineer	Aug 01, 2025	1.3	Address Comment & formatting
PQ	Software Engineer	August 11, 2025	1.4	Address Comment & formatting
PQ	Software Engineer	August 18, 2025	1.5	Address Comment & formatting
PQ	Software Engineer	August 24, 2025	1.6	Address Comment & formatting
PQ	Software Engineer	October 10, 2025	1.7	Address comment from ISCB

Table 1 - Document History

1.0 General

1.1 Overview

This document is the non-proprietary MYCV security policy for Biocryptodisk BEncryptorLib Ver 6.2 (hereafter referred to as “the module").

It contains the security rules under which the module meets the requirement as specified in ISO/IEC 19790 Information technology – Security techniques – Security requirements for cryptographic modules.

The module is a software cryptographic module that is written in the cpp programming language. It supports Elliptic Curve Cryptography ECIES_KEM (P-256, P-384) for asymmetric encryption/decryption, ECDSA (P-256, P-384) for digital signing/digital signature verification, AES-256 CBC mode symmetric encryption/decryption, HMAC-SHA2-DRBG random number generator, SHA2 hashing (SHA2-256, SHA2-384, SHA2-512), SHA3 hashing (SHA3-512, SHAKE-256), Post Quantum Cryptography (ML-DSA-87, ML-KEM-1024)

The module had been ported to various platforms such as Windows, Mac, iPhone and Android for crypto interoperability and secure communication.

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 2 - Security Levels

2.0 Cryptographic Module Specification

2.1 Description

Purpose and Use:
The module is a software library implementing general purpose cryptographic algorithms. The module provides cryptographic services to applications running in the user space of the underlying operating system through an application program interface (API).

Module Type:
Software

Module Embodiment:
Multi-chip standalone

Module Characteristics [O]:
Not applicable

Cryptographic Boundary:
The module consists of the shared library file (BEncryptorLib.dll) which constitutes the cryptographic boundary. The block diagram in Figure 1 shows the cryptographic boundary of the module, its interfaces with the operational environment and the flow of information between the module and operator.

Tested Operational Environment's Physical Perimeter (TOEPP) [O]:
The TOEPP is the general-purpose computer on which the module is installed.

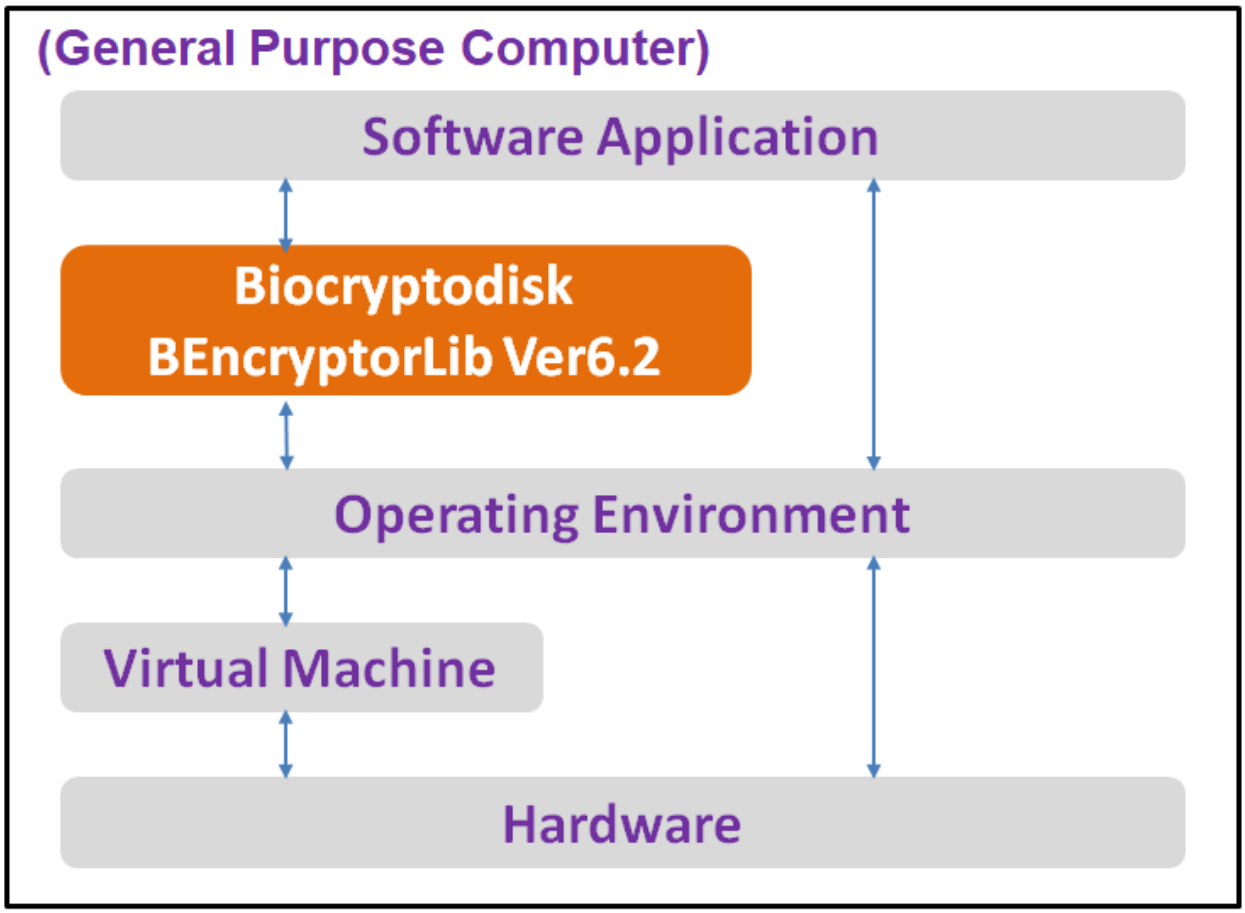

Figure 1: Module Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:
N/A for this module

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

Package or File Name	Software Version	Features	Integrity Test
BEncryptorLib	Ver 6.2	N/A	HMAC-SHA-256

Table 3 - Tested Module Identification – Software, Firmware, Hybrid

Tested Module Identification – Hybrid Disjoint Hardware:
N/A for this module

Tested Operational Environments - Software, Firmware, Hybrid:

Module	Platform	Processor	Operating Systems
Biocryptodisk BEncryptorLib Ver 6.2	Lenovo IdeaPad 3 151AU7	Intel i3-1215U	Windows OS 11 Home Single Language

Table 4 - Configuration tested by the Lab

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:
N/A for this module

2.3 Excluded Components

The module does not claim any excluded components.

2.4 Modes of Operation

The module operates exclusively in the approved mode, which provides all approved services and cryptographic functionalities.

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	In this mode, all approved services are available.	Approved	OPERATION

Table 5 - Modes List and Description

Mode Change Instructions and Status [O]:
The module does not implement a mode change instructions of operation.

Degraded Mode Description [O]:
The module does not implement a degraded mode of operation.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAV Cert	Properties	Reference
AES-CBC	**MyCV-CAV005**	Direction: Encrypt, Decrypt Key Length: 256	ISO/IEC 18033-3:2010 FIPS 197
ECDSA KeyGen	**MyCV-CAV005**	Curve: P-256, P-384 Secret Generation Mode: extra bits, testing candidates	FIPS 186-4 ISO/IEC 14888-3:2016
ECDSA SigGen	**MyCV-CAV005**	Curve: P-256, P-384 Hash Algorithm: SHA2-256, SHA2-384	FIPS 186-4 ISO/IEC 14888-3:2016
ECDSA SigVer	**MyCV-CAV005**	Curve: P-256, P-384 Hash Algorithm: SHA2-256, SHA2-384	FIPS 186-4 ISO/IEC 14888-3:2016
ECIES KEM	**MyCV-CAV005**	Direction: Encrypt, Decrypt Random number algorithm: HASH-DRBG Key derivation algorithm: ANSI X9.63 KDF Encryption algorithm: AES-CBC-256 Mac algorithm: HMAC-SHA-256	ISO 18033-2
ML-KEM-1024 KeyGen	**MyCV-CAV005**	Hash Algorithm: SHA3-512	FIPS 203
ML-KEM-1024	**MyCV-CAV005**	Direction: Encapsulate, Decapsulate Hash Algorithm: SHA3-512	FIPS 203
ML-DSA-87 KeyGen	**MyCV-CAV005**	Hash Algorithm: SHAKE-256	FIPS 204
ML-DSA-87 SigGen	**MyCV-CAV005**	Hash Algorithm: SHAKE-256 Message Length: 8-65536 Increment 8 Context Length: 0-2040 Increment 8 Signature Interfaces: internal, external Pre Hash: prehash, pure Deterministic: Yes, No External Mu: Yes, No	FIPS 204
ML-DSA-87 SigVer	**MyCV-CAV005**	Hash Algorithm: SHAKE-256 Message Length: 8-65536 Increment 8 Context Length: 0-2040 Increment 8 Signature Interfaces: external, internal Pre Hash: preHash External Mu: Yes, No	FIPS 204
HASH-DRBG	**MyCV-CAV005**	Mode: SHA2-256 Prediction Resistance: No Reseed: Yes, No Entropy Input: 256 Nonce: 128 Personalization String Length: 256 Additional Input: 256 Returned Bits: 1024	ISO/IEC 18031:2011

Algorithm	CAV Cert	Properties	Reference
SHA2	**MyCV-CAV005**	Mode: 384, 512 Message Length: 8-65536 Increment 8	ISO/IEC 10118-3:2018
SHA3	**MyCV-CAV005**	Mode: 512 Message Length: 8-65536 Increment 8	ISO/IEC 10118-3:2018
HMAC-SHA2	**MyCV-CAV005**	MAC Length: 32-256 Increment 8 Key Length: 8-448 Increment 8	ISO/IEC 9797-2

Table 6 - Approved Algorithms

NOTE: Only the algorithms specified in this section are supported by the module in approved mode of operation. No operational use of an algorithm may be performed until the corresponding CAST has passed.

Neutral Algorithms:

Algorithm	CAV Cert	Properties	Reference
SHA2	**MyCV-CAV005**	Mode: 256 Message Length: 0-1024 Increment 8	ISO/IEC 10118-3:2018
SHA3	**MyCV-CAV005**	Mode: SHAKE-256 Supports Empty Message Output Length: 16-1024 Increment 8	FIPS PUB 202 (SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions)

Table 7 - Neutral Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
DRBG	DRBG	Deterministic Random Number Generator	-	SHA2-256 HASH-DRBG
Message Authentication	MAC	Hash-Based Message Authentication Code	-	HMAC-SHA2-256
Secure Hash	SHA	Secure Hash Function	-	SHA2-256 SHA2-384 SHA3-512 SHA3-SHAKE-256
ECC Asymmetric Key-Pair Generation	CKG	Generate an ECC Asymmetric Key-Pair	-	ECDSA KeyGen HASH-DRBG
ECC asymmetric encryption/decryption	KEM	Perform ECIES asymmetric encryption	-	ECIES KEM AES-CBC HMAC-SHA2
ECC Digital Signature Generation	DigSig - SigGen	Digital Signature Generation	-	ECDSA SigGen SHA2-256 SHA2-384

Name	Type	Description	Properties	Algorithms
ECC Digital Signature Verification	DigSig-SigVer	Digital Signature Verification	-	ECDSA SigVer SHA2-256 SHA2-384
ML-KEM Asymmetric Key-Pair Generation	CKG	Generate an ML-KEM Asymmetric Key-Pair	-	ML-KEM-1024 KeyGen SHA3-512
ML-KEM Encapsulation and Decapsulation	KEM	ML-KEM Encapsulation and Decapsulation	-	ML-KEM-1024
ML-DSA Asymmetric Key-Pair Generation	CKG	Generate an ML-DSA-87 Asymmetric Key-Pair	-	ML-DSA-87 KeyGen SHAKE-256
ML-DSA Digital Signature Generation	DigSig-SigGen	Digital Signature Generation	-	ML-DSA-87 SigGen Deterministic: true SHAKE-256
ML-DSA Digital Signature Verification	DigSig-SigVer	Digital Signature Verification	-	ML-DSA-8 SigVer SHAKE-256
UnAuth Block Cipher	BC-UnAuth	Unauthenticated block cipher	-	AES-CBC

Table 8 - Security Function Implementations

2.7 Algorithm Specific Information

The conditions for using the Module in the Approved mode of operation are:

1) ECDSA P-256 must be used with SHA2-256 hash function.
2) ECDSA P-384 must be used with SHA2-384 hash function.
3) Data output is inhibited during self-tests, zeroization, and error states.

2.8 RBG and Entropy

N/A for this module

2.9 Key Generation

N/A for this module

2.10 Key Establishment

N/A for this module

2.11 Industry Protocols

N/A for this module

2.12 Additional Information [O]

N/A for this module

3.0 Cryptographic Module Interfaces

3.1 Ports and Interfaces

The module is a software only module that operates on a general purpose computing (GPC) platform. The physical ports and logical interfaces are consistent with a GPC operating environment. The logical interfaces of the module is implemented via an Application Programming Interface (API). The module supports the following ISO/IEC 19790 logical interfaces.

Physical Port	Logical Interface	Data that passes over port/interface
N/A	Data Input Interface	API input parameters for data
N/A	Data Output Interface	API output parameters for data
N/A	Control Input Interface	API function calls
N/A	Control Output Interface	The module does not output any commands, signals or control data used to control another module.
N/A	Status Output Interface	API return values under log messages

Table 9 - Ports and Interfaces

3.2 Trusted Channel Specification [O]

N/A for this module

3.3 Control Interface Not Inhibited [O]

N/A for this module

3.4 Additional Information [O]

N/A for this module

4.0 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 10 - Roles

The module does not include an authentication mechanism for its roles. The Crypto Officer (CO) role is implicitly assumed when utilising the cryptographic services as provided by the module. Additionally, the module does not support designated roles for maintenance or general users.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
ShowVersion	Provide crypto module info.	Successful completion of the service	None	Module name and version number		CO
ShowStatus	Provide module status	Successful completion of the service	None	Status return		CO
Self-Test (dll integrity)	Perform library integrity test	Successful completion of the service (return 0)	None	Status return	Message Authentication	CO
Self-Test (KAT & PCT)	Perform designated KAT & PCT test	Successful completion of	None	Status return		CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		the service (return 0)				
Random	Generate DRBG random number	Successful completion of the service	None	Random number	DRBG	CO Seed: G,Z, DRBG_E: W,E DRBG_N: W,E Internal state DRBG_V: W,E DRBG_C: W,E
Generate Keypair	Generate asymmetric keypairs	Successful completion of the service	ECC curve: P256, P384 ML-KEM: None (default: 1024-256) ML-DSA: None (default 87-256)	ECC_PUBKEY, ECC_PRIKEY MLK_ECSKEY, MLK_DCSKEY MLD_PUBKEY MLD_PRIKEY	ECC Asymmetric Key-Pair Generation, ML-KEM Asymmetric Key-Pair Generation, ML-DSA Asymmetric Key-Pair Generation	ECC_PUBKEY: G,R,Z ECC_PRIKEY: G,R,Z MLK_ECSKEY: G,R,Z MLK_DCSKEY: G,R,Z MLD_PUBKEY: G,R,Z MLD_PRIKEY: G,R,Z
Message digest	Generate hashing value	Successful completion of the service	Message, message length	Digest value	Secure Hash	CO
Symmetric cipher	Perform symmetric encryption and decryption	Successful completion of the service	AES_KEY, Initial vector, Plaintext or ciphertext	Plaintext or ciphertext	UnAuth Block Cipher	CO AES_KEY: W,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Asymmetric cipher	Perform asymmetric encryption and decryption	Successful completion of the service	ECC: Encryption (ECC_PRIKEY, plaintext), Decryption (ECC_PUBKEY, cipher, sharedInfo, mac) ML-KEM: Encapsulation (MLK_ECSKEY) Decapsulation (ciphertext, MLK_DCSKEY)	ECC: Encryption (ciphertext, sharedInfo, mac) Decryption (plaintext) ML-KEM: Encapsulation(ciphertext, sharedSecretKey) Decapsulation (sharedSecretKey)	ECC asymmetric encryption/ decryption, ML-KEM Encapsulation and Decapsulation	CO ECC_PUBKEY: W,E,Z ECC_PRIKEY: W,E,Z MLK_ECSKEY: W,E,Z MLK_DCSKEY: W,E,Z
Digital Signature	Generate and verify digital signature	Successful completion of the service	ECDSA: Sign: (flag, ECC_PRIKEY, message, message length), Verify: (flag, ECC_PUBKEY, signature, signature size) ML-DSA: Sign: (MLD_PRIKEY, message, message length,	Status return Sign: Signature Verify: 0:match, -1: fail	ECC Digital Signature Generation, ECC Digital Signature Verification ML-DSA Digital Signature Generation, ML-DSA Digital Signature Verification	CO ECC_PUBKEY: W,E,Z ECC_PRIKEY: W,E,Z MLD_PRIKEY: W,E,Z MLD_PUBKEY: W,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
			random (optional)) Verify: (MLD_PUBKEY, signature, signature size)			
Zeroise	Free computer memory	Successful completion of the service	All variables consist of SSPs			CO: All SSPs: Z
HMAC	Compute tag of encrypted message in ECIES encryption and decryption.	Successful completion of the service	ECC asymmetric encryption / decryption: (encrypted message, derived share secret as mac key)	Hash value	Message Authentication	CO: HMAC_KEY: W,E

Table 11 - Approved Services

The table above lists the approved services. For each service, the table lists the associated cryptographic algorithm(s), the role to perform the service, the cryptographic keys or SSPs involved, and their access type(s). The following convention is used to specify access rights to a SSP:

- 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 = Zeroise: The module zeroises the SSP.

4.4 Non-Approved Services

N/A for this module

4.5 External Software/Firmware Loaded

N/A for this module

4.6 Bypass Actions and Status [O]

N/A for this module

4.7 Cryptographic Output Actions and Status [O]

N/A for this module

4.8 Additional Information [O]

N/A for this module

5.0 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing the HMAC-SHA-256 value using predefined MAC key. The computed HMAC value is XORed with a 32-byte constant to produce the final embedded HMAC and is stored at the end of the module. It was computed at build time for each software component of the module. When service is started, it computes an HMAC-SHA-256 hash over the module file, excluding the embedded HMAC value and the magic number. The computed HMAC is then compared against the embedded HMAC value that stored to the library file. If the HMAC values do not match, the test fails and the module enters the error state.

5.2 Initiate on Demand

The crypto officer can initiate the integrity test and KAT on demand by reloading the Module or by calling the API DLLIntegrityTest() and SelfTest() (invoked using Module's library handle) at any time after power on.

5.3 Open-Source Parameters [O]

N/A for this module

5.4 Additional Information [O]

N/A for this module

6.0 Operational Environment

6.1 Operational Environment Type and Requirements

The module is a software module, which is operated in a modifiable operational environment per ISO/IEC 19790 level 1 specifications. The module's software version running on each tested platform is version 6.2.

Type of Operational Environment:
Modifiable

How Requirements are Satisfied [O]:
The module should be compiled and installed as stated in section 11. If properly installed, the operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.
The module does not support concurrent operators.

6.2 Configuration Settings and Restrictions [O]

Not applicable

6.3 Additional Information [O]

Not applicable

7.0 Physical Security

The module is comprised of software only and therefore this section is not applicable.

7.1 Mechanisms and Actions Required [O]

N/A for this module

7.2 User Placed Tamper Seals [O]

Number:

Placement:

Surface Preparation:

Operator Responsible for Securing Unused Seals:

Part Numbers:
N/A for this module

7.3 Filler Panels [O]

N/A for this module

7.4 Fault Induction Mitigation [O]

N/A for this module

7.5 EFP/EFT Information [O]

N/A for this module

7.6 Hardness Testing Temperature Ranges [O]

N/A for this module

7.7 Additional Information [O]

N/A for this module

8.0 Non-Invasive Security

This module does not implement any non-invasive security mechanism, and therefore this Section is not applicable.

8.1 Mitigation Techniques [O]

N/A for this module

8.2 Effectiveness [O]

N/A for this module

8.3 Additional Information [O]

N/A for this module

9.0 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
S1	RAM(Memory)	Dynamic

Table 12 - Storage Areas

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in RAM in plaintext form. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type
INP	Call API input parameter	Cryptographic Module	Plaintext	Manual	Electronic
OUTP	Cryptographic Module	Call API output parameter	Plaintext	Manual	Electronic

Table 13 - SSP Input-Output Methods

The module does not support manual SSP entry or intermediate SSP generation output. The SSPs are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form within the physical perimeter of the operational environment. This is allowed by [FIPS 140-3_IG] 9.5.A, according to the “CM Software to/from App via TOEPP Path” entry on the Key Establishment Table.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation

Zero1	Zeroizes the SSPs contained	Overwrite with zero.	mySecureZeroMemory()
Zero2	Unload crypto module.	clear the allocated handle object from S1.	ReleaseHandle()

Table 14 - SSP Zeroization Methods

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The application that is acting as the CO is responsible for calling the appropriate zeroization functions provided in the module's API and listed in the above table. Calling mySecureZeroMemory(), which will zeroize the SSPs and also invoke the corresponding API functions listed in the above table to zeroize SSPs. The zeroization functions overwrite the memory occupied by SSPs with “zeros” and deallocate the memory with the regular memory deallocation operating system call. The completion of a zeroization routine(s) will indicate that a zeroization procedure succeeded.

9.4 SSPs

The table below describes the cryptographic keys used by the module.

Name	Description	Size-Strength	Type Category	Generated By	Established by	Used By
AES_KEY	AES key used for symmetric encryption. Modes: CBC	256 – 256 bits	Symmetric Key - CSP	CryptGenRandom (windows) or SYS_random (linux)		AES-CBC
ECC_PUBKEY_EXT	ECC public key used for asymmetric encryption and digital signature verification.	256-128 bits, 384-192 bits	Asymmetric Key - PSP	Generated outside the module boundary		ECIES KEM enc, ECDSA SigVer
ECC_PRIKEY_EXT	ECC private key used for asymmetric decryption and digital signing.	256-128 bits, 384-192 bits	Asymmetric Key - CSP	Generated outside the module boundary		ECIES KEM dec, ECDSA SigGen
ECC_PUBKEY	ECC public key used for asymmetric encryption and digital signature verification	256-128 bits, 384-192 bits	Asymmetric Key - PSP	ECC Generate Keypair		ECIES KEM enc, ECDSA SigVer
ECC_PRIKEY	ECC private key used for asymmetric decryption and digital signing.	256-128 bits, 384-192 bits	Asymmetric Key - CSP	ECC Generate Keypair		ECIES KEM dec, ECDSA SigGen
HMAC_KEY	HMAC key used to computes the MAC of	256-128 bits	Message Authentication Key - CSP		Established in ECIES	HMAC-SHA2, ECIES KEM

	encrypted message in asymmetric encryption. Uses the MAC to check the compute tag in asymmetric decryption.					
Seed	Seed-materials consist of entropy, nonce and personal string.	Size in bytes: 256	Seed value - CSP	CryptGenRandom (windows) or SYS_random (linux)		HASH-DRBG
DRBG_E	Entropy input loaded from the external source	256-bits-256-bit	Entropy - CSP	Seed		HASH-DRBG
DRBG_N	Nonce input loaded from the external source	128-bit-256-bits	Nonce - CSP	Seed		HASH-DRBG
DRBG_V	HASH-DRBG Internal State Secret V	440-bits-256-bit	Secret V - CSP	Generate		HASH-DRBG
DRBG_C	HASH-DRBG Internal State Secret C	440-bits-256-bit	Secret C - CSP	Generate		HASH-DRBG
MLK_ECSKEY	ML_KEM key used for encapsulation and shared secret generation.	Mode: 1024-256	Asymmetric encapsulation Key - PSP	ML-KEM Generate Keypair		ML-KEM-1024
MLK_DCSKEY	ML_KEM key used for decapsulation and compute shared secret with given	Mode: 1024-256	Asymmetric decapsulation Key - CSP	ML-KEM Generate Keypair		ML-KEM-1024

	ciphertext.					
MLD_PUBKEY	ML_DSA public key used for digital signature verification.	Mode: 87-256	Asymmetric private key - PSP	ML-DSA Generate Keypair		ML-DSA-87 SigVer
MLD_PRIKEY	ML_DSA private key used for digital signing.	Mode: 87-256	Asymmetric private key - CSP	ML-DSA Generate Keypair		ML-DSA-87 SigGen
Shared secret/ shared info	Output value established from ECIES encryption and ML-KEM encapsulation.	-	CSP	-	Established via EC shared secret computation & ML_KEM encapsulation.	ECIES dec, ML-KEM decapsulation.

Table 15 - SSP Table 1

Name	Input-Output	Storage	Storage Duration	Zeroization	Related SSPs
AES_KEY	INP	S1:Plaintext	call in lifetime	Zero1	
ECC_PUBKEY_EXT	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : ECC_PRIKEY_EXT ECIES KEM enc: use with sharedInfo, HMAC_KEY
ECC_PRIKEY_EXT	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : ECC_ PUBKEY_EXT ECIES KEM dec: use with

					sharedInfo, HMAC_KEY
ECC_PUBKEY	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : ECC_PRIKEY ECIES KEM enc: use with sharedInfo, HMAC_KEY
ECC_PRIKEY	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : ECC_ PUBKEY ECIES KEM dec: use with sharedInfo, HMAC_KEY
HMAC_KEY	INP	S1:Plaintext	call in lifetime	Zero1	ECIES KEM : use with Shared info
Seed	INP	S1:Plaintext	call in lifetime	Zero1	
DRBG_E	INP	S1:Plaintext	call in lifetime	Zero1	Derived from Seed
DRBG_N	INP	S1:Plaintext	call in lifetime	Zero1	Derived from Seed
DRBG_V	INP	S1:Plaintext	call in lifetime	Zero1	Derived from Seed
DRBG_C	INP	S1:Plaintext	call in lifetime	Zero1	Derived from Seed
MLK_ECSKEY	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : MLK_DCSKEY ML_KEM enc: use with Shared secret
MLK_DCSKEY	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : MLK_ECSKEY ML_KEM dec: use with Shared secret
MLD_PUBKEY	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : MLK_PRIKEY

MLD_PRIKEY	INP	S1:Plaintext	call in lifetime	Zero1	Paired With : MLK_PUBKEY
Shared secret/ shared info	INP	S1:Plaintext	call in lifetime	Zero1	shared info Used with : . ECC_PUBKEY_EXT ECC_PRIKEY_EXT ECC_PUBKEY ECC_PRIKEY Shared secret Used with : MLK_ECSKEY MLK_DCSKEY

Table 16 - SSP Table 2

Keys can be provided to the module via API input parameters. The module does not enter or output keys outside its physical boundary. Zeroization is performed using power cycle.

9.5 Transitions [O]

Not Applicable

9.6 Additional Information [O]

Not Applicable

10.0 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm	Test Properties	Test Method	Test Type	Indicator
HMAC-SHA2-256	256-bit	Message Authentication	Software Integrity	SELF-TEST

Table 17 - Pre-Operational Self Tests

The module performs the following power-up and conditional self-tests. Upon failure or

power-up or conditional self-test the module halts its operation.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC	256-bit	KAT	CAST	SELF-TEST	Encrypt	Performed on module load
AES-CBC	256-bit	KAT	CAST	SELF-TEST	Decrypt	Performed on module load
ECC	curve: P256; hashType: SHA2-256	KAT	CAST	SELF-TEST	Key generation	Performed on module load
ECIES	curve: P256; hashType: SHA2-256	KAT	CAST	SELF-TEST	Encrypt	Performed on module load
ECIES	curve: P256; hashType: SHA2-256	KAT	CAST	SELF-TEST	Decrypt	Performed on module load
ECDSA	curve: P256; hashType: SHA2-256	KAT	CAST	SELF-TEST	Sign	Performed on module load
ECDSA	curve: P256; hashType: SHA2-256	KAT	CAST	SELF-TEST	Verify	Performed on module load
SHA2	512-bit	KAT	CAST	SELF-TEST	Hashing	Performed on module load
HASH-DRBG	Mode: SHA2-256	KAT	CAST	SELF-TEST	Random number generation	Performed on module load
HMAC SHA 256	hash type: SHA256; key length: 40 bytes	KAT	CAST	SELF-TEST	MAC	Performed on module load
ML-KEM	hashType: SHA3-512	KAT	CAST	SELF-TEST	Key generation	Performed on module load
ML-KEM	hashType: SHA3-512	KAT	CAST	SELF-TEST	Encapsulation	Performed on module

						load
ML-KEM	hashType: SHA3-512	KAT	CAST	SELF-TEST	Decapsulation	Performed on module load
ML-DSA	hashType: SHAKE-256	KAT	CAST	SELF-TEST	Key generation	Performed on module load
ML-DSA	hashType: SHAKE-256	KAT	CAST	SELF-TEST	Sign	Performed on module load
ML-DSA	hashType: SHAKE-256	KAT	CAST	SELF-TEST	Verify	Performed on module load
ECC-PCT	curve size: P256, P384	Pairwise consistency test	CAST	SELF-TEST	Condition self-test performed by the module	Performed when module performing key generation
ML-KEM-PCT	hashType: SHA3-512	Pairwise consistency test	CAST	SELF-TEST	Condition self-test performed by the module	Performed when module performing key generation
ML-DSA-PCT	hashType: SHAKE-256	Pairwise consistency test	CAST	SELF-TEST	Condition self-test performed by the module	Performed when module performing key generation

Table 18 - Conditional Self Tests

The self-test will trigger once module is powered on by using library handle and available on demand which can be executed by crypto officer (CO) at any time. The CO may optionally invoke any CAST after the module has passed the first operational self-test. The CAST will no longer run automatically after it has passed the first time on module load.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC SHA 256	Integrity test	SW integrity	On module load, CO can invoke manually thereafter	Automatic or Manually

Table 19 - Pre-operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
AES-CBC	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ECC	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ECIES	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ECIES	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ECDSA	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ECDSA	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
SHA2	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
HASH-DRBG	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
HMAC SHA 256	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ML-KEM	KAT	CAST	On module load, CO can invoke manually thereafter	Manually

ML-KEM	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ML-KEM	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ML-DSA	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ML-DSA	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ML-DSA	KAT	CAST	On module load, CO can invoke manually thereafter	Manually
ECC-PCT	PCT	CAST	Invoked by automatically during key generation	Automatic
ML-KEM-PCT	PCT	CAST	Invoked by automatically during key generation	Automatic
ML-DSA-PCT	PCT	CAST	Invoked by automatically during key generation	Automatic

Table 20 - Periodic Self-Test

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	The module will return an error code to indicate the error and will enter the Error state.	Error during load library, initialisation and execution of the crypto API error. Failure of integrity test, any of the CASTs, any of the PCTs.	The module must be reloaded except for errors cause by PCT. The PCT error condition will be recovered after a successful zeroisation of the keypair.	ERROR1

Fatal Error State	The module will abort and will not be available.	TOEPP memory overflow or power outage or multiple APIs are called by a single library handle at the same time.	The error can be recovered by reloading of the module.	ERROR1

Table 21 - Error States

After the pre-operational self-tests and the CASTs succeed, the module becomes operational. If any of the pre-operational self-tests or any of the CASTs fail an error message is returned, and the module transitions to the error state.
When the module fails any pre-operational self-test or conditional test, the module will return an error code to indicate the error and will enter the Error state. Any further cryptographic operation is inhibited. The calling application can obtain the module state by calling the GetFiniteState() API function. In the Error state, the error can be recovered if transit from the Operation state. The error cannot be recovered if transit from the Power On state, Init state and Self-Test state, thereafter will transit to Power Off state to free library from the system memory. If multiple operations are requested concurrently by a single class handle, the module will transition to Fatal Error state, the module will abort and will not be available.

10.5 Operator Initiation of Self-Tests [O]

The software integrity tests and the CASTs can be invoked relying on the DLLIntegrityTest () and SelfTest() API function call. The PCTs can be invoked on demand by requesting the Key Generation service.

10.6 Additional Information [O]

Not Applicable

11.0 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

As the module is an integrated component of the Biocryptodisk Vidcall Version 8.2 software solution, module operators have no ability to independently load the module onto the target platform. The module and its calling application are to be installed on a platform specified in section 2.2 or one where portability is maintained.

11.2 Administrator Guidance

All the functions, ports and logical interfaces described in this document are available to the Crypto Officer.

This module is designed to support Biocryptodisk Vidcall Version 8.2 software applications, and these applications are the sole consumers of the cryptographic services provided by the module. No end-user action is required to initialize the module for operation; the calling application performs any actions required to initialize the module.

The pre-operational integrity test and cryptographic algorithm self-tests are performed automatically via a default entry point (DEP) when the module is loaded for execution, without any specific action from the calling application or the end-user. End-users have no means to short-circuit or bypass these actions. Failure of any of the initialization actions will result in a failure of the module to load for execution.

11.3 Non-Administrator Guidance

The module implements only the Crypto Officer. There are no requirements for non-administrator guidance

11.4 Design and Rules [O]

Not Applicable

11.5 Maintenance Requirements [O]

Not Applicable

11.6 End of Life [O]

For secure sanitization of the cryptographic module, the end user has obligation to zeroize all declared variable in application that consists of sensitive data in volatile memory.

The module does not possess persistent storage of SSPs, so further sanitization steps are not required.

11.7 Additional Information [O]

Not Applicable

12.0 Mitigation of Other Attacks

12.1 Attack List [O]

Not Applicable

12.2 Mitigation Effectiveness [O]

Not Applicable

12.3 Guidance and Constraints [O]

Not Applicable

12.4 Additional Information [O]

Not Applicable

13.0 References

Reference	Specification
[ISO/IEC 18033-3:2010]	Information technology — Security techniques — Encryption algorithms Part 3: Block ciphers
[ISO/IEC 10118-3:2004]	Information technology — Security techniques — Hash-functions Part 3: Dedicated hash-functions
[ISO/IEC 10118-3:2018]	IT Security techniques — Hash-functions Part 3: Dedicated hash-functions
[ISO/IEC 18031:2011]	Information technology — Security techniques — Random bit generation
[ISO/IEC 9797-2: 2021]	Information security — Message authentication codes (MACs)Part 2: Mechanisms using a dedicated hash-function
[ISO/IEC 14888-3:2018]	Digital Signatures with Appendix, Part 3: Discrete Logarithm-Based Mechanisms

[ISO/IEC 18033-2:2006]	Information technology — Security techniques — Encryption algorithms Part 2: Asymmetric ciphers
[ANSIX 9.31]	Standards for Random Number Generator
[ANSIX 9.63]	Key Derivation Function
[FIPS 140-2]	Security Requirements for Cryptographic Modules
[FIPS 180-4]	Secure Hash Standard (SHS)
[FIPS 186-2/4]	Digital Signature Standard
[FIPS 197]	Advanced Encryption Standard (AES)
[FIPS 198-1]	The Keyed-Hash Message Authentication Code (HMAC)
[FIPS 202]	SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions
[FIPS 203]	Module-Lattice-Based Key-Encapsulation Mechanism Standard
[FIPS 204]	Module-Lattice-Based Digital Signature Standard
[ISO/IEC 19790]	Security Requirements for Cryptographic Modules
[ISO/IEC 24759]	Information technology — Security techniques — Test requirements for cryptographic modules
[SP 800-38A]	Recommendation for Block Cipher Modes of Operation: Three Variants of Cipher text Stealing for CBC Mode
[SP 800-90A]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators

Table 22 - References