Commvault Systems, Inc.

Commvault Crypto Library
Software Version: 3.0
FIPS 140-3 Non-Proprietary Security Policy
Level 1 Validation

November 2024

Prepared by:

www.acumensecurity.net

Table of Contents

List of Tables

List of Figures

1. General

Introduction

Federal Information Processing Standards Publication 140-3 — Security Requirements for Cryptographic modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS) runs the Cryptographic Module Validation Program (CMVP) program. The NVLAP accredits independent testing labs to perform FIPS 140-3 testing; the CMVP validates modules meeting FIPS 140-3 validation. Validated is the term given to a module that is documented and tested against the FIPS 140-3 criteria.

More information is available on the CMVP website at:
https://csrc.nist.gov/projects/cryptographic-module-validation-program.

About this document

This non-proprietary cryptographic module security policy for the Commvault Systems, Inc. Commvault Crypto Library, provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-3.

The Commvault Crypto Library may also be referred to as the “module” in this document.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Commvault Systems, Inc. shall have no liability for any error or damages of any kind resulting from the use of this document.

Notices

This document describes the cryptographic module security policy for the Commvault Crypto Library (also referred to as the "module" hereafter) with software version 3.0 from Commvault Systems, Inc. The module type is software and has a multi-chip standalone embodiment. It contains specification of the security rules, under which the cryptographic module operates, including the security rules derived from the requirements of the FIPS 140-3 standard.

The following table lists the level of validation for each area in FIPS 140-3:

ISO/IEC 24759 Section 6. [Number Below]	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

Table 1 – Security Levels

The module claims an overall Security Level 1.

2. Cryptographic module specification

The tested platforms are as follows:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Microsoft Windows Server 2019	Fujitsu RX2530 M5	Intel Xeon R Silver 4208	With AES-NI
2	Microsoft Windows Server 2019	Fujitsu RX2530 M5	Intel Xeon R Silver 4208	Without AES-NI
3	Red Hat Enterprise Linux 8.4	Fujitsu RX2530 M5	Intel Xeon R Silver 4208	With AES-NI
4	Red Hat Enterprise Linux 8.4	Fujitsu RX2530 M5	Intel Xeon R Silver 4208	Without AES-NI

Table 2 – Tested Operational Environments

The module is vendor-affirmed to operate on the platforms listed below. The CMVP makes no claim 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.

#	Operating System	Hardware Platform
1	Red Hat Enterprise Linux 8.5	Fujitsu RX2530 M5 with Intel Xeon Platinum 8370C
2	Red Hat Enterprise Linux 8.6	Fujitsu RX2530 M5 with Intel Xeon Platinum 8370C
3	Oracle Linux 8.5	Fujitsu RX2530 M5 with Intel Xeon Platinum 8370C
4	Oracle Linux 8.6	Fujitsu RX2530 M5 with Intel Xeon Platinum 8370C
5	Microsoft Windows Server 2022	Fujitsu RX2530 M5 with Intel Xeon Platinum 8370C

Table 3 – Vendor Affirmed Operational Environments

Module usage & cryptographic boundary

Commvault Crypto Library is a library-based software cryptographic module, providing cryptographic services to all Commvault products. Particularly Commvault Backup and Recovery and Metallic™. Commvault Backup and Recovery and Metallic is composed of several program modules allowing one to perform data backup, hierarchical storage management (data archiving) and continuous data replication. The **cvcl.dll** and **libcvcl.so**, version 3.0, binaries provide encryption services for Commvault Backup and Recovery and Metallic platforms and represent the cryptographic boundary of the module.

The module implements AES (data encryption, key protection), RSA (key generation, signing, verification), SHA-1, SHA2-256, SHA2-512, HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-512, and AES 256 CTR_DRBG algorithms in the approved mode. The module type is software and has a multichip stand-alone embodiment.

Commvault Crypto Library is packaged as a dynamic (shared) software module exporting the cryptographic API to any software that supports C calling conventions.

The cryptographic module's operational environment is a modifiable operational environment. The cryptographic boundary of the software module consists of **cvcl.dll** and **libcvcl.so**, per the image in Figure 1. No components have been excluded from the cryptographic boundary of the module.

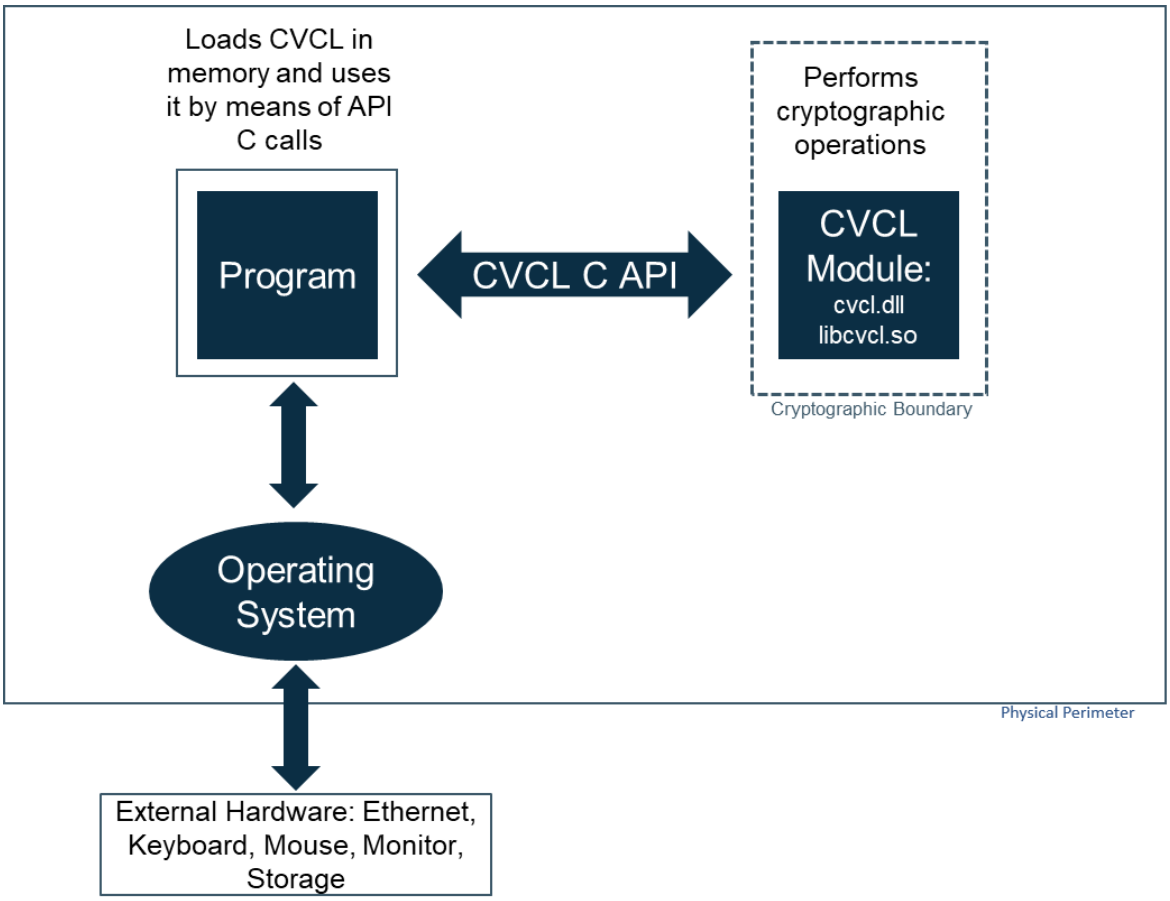
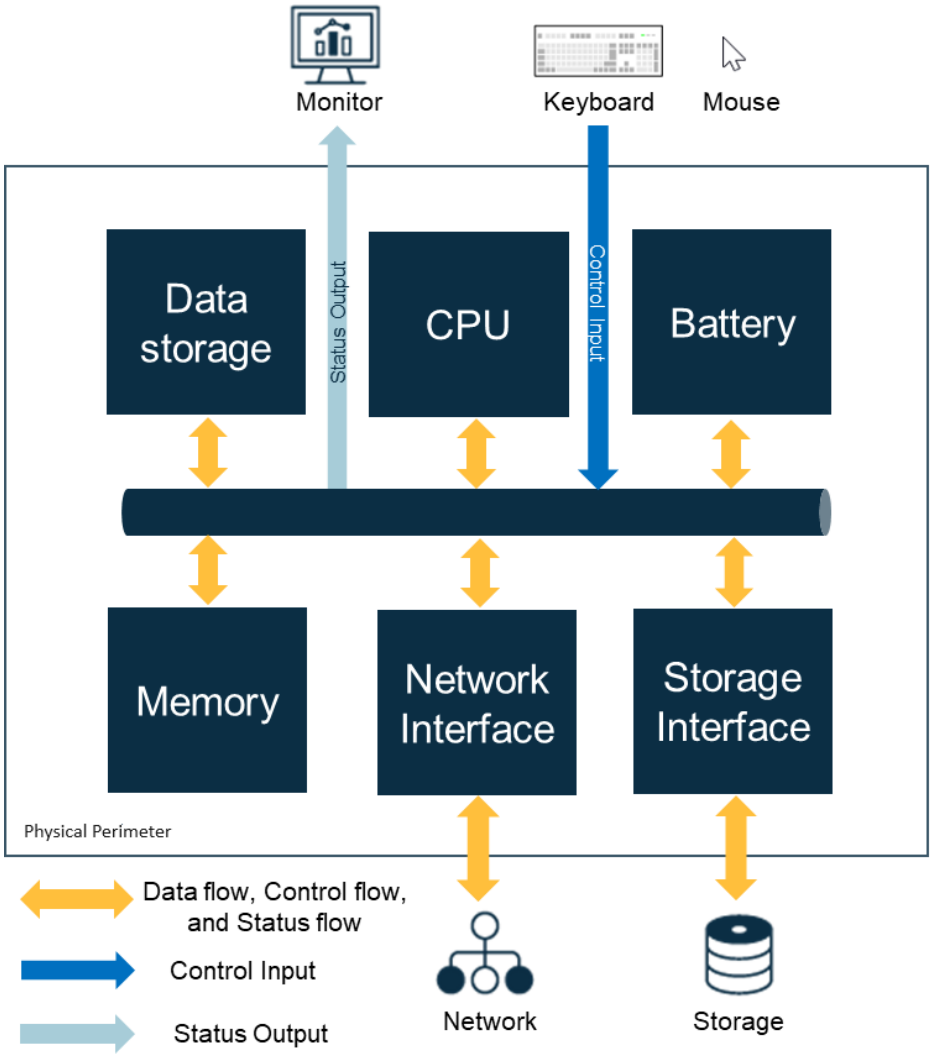


Figure 1 – Cryptographic Boundary Block Diagram



General Purpose Computing Platform

Figure 2 – Physical Perimeter

The module claims an overall Security Level of 1. The module does not implement any non-invasive security mitigations or mitigations of other attacks and thus the requirements per these sections are not applicable.

Modes of operation

The module supports both an approved and non-approved mode of operation. Both modes of operation are determined by this security policy, as the module does not enforce the rules associated with each mode. It is the responsibility of the caller to ensure that only approved algorithms and services are utilized. The module provides an indicator for each available algorithm, which specifies whether the algorithm is approved or non-approved.

The module must always be zeroised when switching between the approved mode of operation and the non-approved mode of operation and vice-versa.

Approved mode

To use the module in the approved mode, it must be installed and operated in accordance with the rules described in this section. After the module initializes, all services are available to the user, and it is the user's responsibility to only call approved or allowed services to ensure the module is operating in the approved mode of operation. Please see Section 11 of this document for secure setup and initialization requirements.

Module security policy rules

The module will operate in the approved mode only if approved cryptographic services are used. Since the module provides both approved and non-approved services, it is the application loading the module that is responsible to ensure that it is using only approved services. (Please refer to Table 9 for the list and classification of all module services.)

For the Backup and Recovery user, the correct data encryption method should be selected in the properties of each of the client machines in the CommCell.

Non-approved mode

If non-approved cryptography and services are selected, the module will be operating in the non-approved mode. These algorithms and services shall not be used when operating in the approved mode of operation.

Degraded operation

The module does not support a degraded mode of operation.

Overall security rules of operation

The module design corresponds to the security rules below. The term *shall* in this context specifically refers to a requirement for correct usage of the module in the Approved mode; all other statements indicate a security rule implemented by the module.

1. Self-tests do not require any operator action.

2. Data output is inhibited during SSP generation, self-test execution, zeroisation, and error states.
3. Status information does not contain SSPs or sensitive data that if misused could lead to a compromise of the module.
4. There are no restrictions on which SSPs are zeroised by the zeroisation service. Successful zeroisation (by either restart or function call procedure) for the operator is implicitly indicated by the successful completion of the procedural zeroisation service method.
5. The module does not support a maintenance interface or role.
6. The module does not output intermediate key values.
7. The module does not output plaintext CSPs.
8. The Crypto Officer shall retain control of the module while zeroisation is in process.
9. From the perspective of the Backup and Recovery user, the correct data encryption method should be selected in the properties of each of the client machines in the CommCell.

CAVP Cert[1]	Algorithm and Standard	Mode/Method	Description/Key Sizes/Key Strengths	Use/Function
A2412	AES-CBC-CS2 (FIPS 197, SP 800-38A, SP 800-38A Addendum)	AES-CBC-CS2	Key length: 128, 256 bits	Symmetric Encryption, Symmetric Decryption
A2412	AES-ECB (FIPS 197, SP 800-38A)	AES-ECB	Key length: 128, 256 bits	Symmetric Encryption, Symmetric Decryption
A2412	Counter DRBG (SP 800-90Ar1)	Counter DRBG	256 bits	Pseudo-Random Number Generation
A2412	HMAC-SHA-1 (FIPS 198-1)	HMAC-SHA-1	Key Length: 8-1024 bits	Message Authentication
A2412	HMAC-SHA2-256 (FIPS 198-1)	HMAC-SHA2-256	Key Length: 8-1024 bits	Message Authentication
A2412	HMAC-SHA2-512 (FIPS 198-1)	HMAC-SHA2-512	Key Length: 8-1024 bits	Message Authentication

[1] There are algorithms, modes, and key/moduli sizes that have been CAVP-tested but are not used by any approved service of the module. Only the algorithms, modes/methods, and key lengths/curves/moduli shown in this table are used by an approved service of the module.

CAVP Cert[1]	Algorithm and Standard	Mode/Method	Description/Key Sizes/Key Strengths	Use/Function
A2412	RSA KeyGen (FIPS186-4)	RSA KeyGen (FIPS186-4) B.3.2	Modulo: 2048, 3072, 4096 bits with SHA2-512	Generate asymmetric key pairs
A2412	RSA SigGen (FIPS186-4)	RSA SigGen (FIPS186-4) ANSI X9.31, PKCS v1.5	Modulo: 2048, 3072, 4096 bits with SHA2-256, SHA2-512	Signature Generation
A2412	RSA SigVer (FIPS186-2)	SigVer [ANSI X9.31, PKCS v1.5]	Modulo: 1024, 2048, 3072, 4096 bits with SHA-1, SHA2-256, SHA2-512	Legacy Signature Verification
A2412	RSA SigVer (FIPS186-4)	RSA SigVer (FIPS186-4) ANSI X9.31, PKCS v1.5	Modulo: 1024, 2048, 3072, 4096 bits with SHA-1, SHA2-256, SHA2-512	Signature Verification
A2412	SHA-1 (FIPS 180-4)	SHA-1	Message Length: 0-65536	Message Digest
A2412	SHA2-256 (FIPS 180-4)	SHA2-256	Message Length: 0-65536	Message Digest
A2412	SHA2-512 (FIPS 180-4)	SHA2-512	Message Length: 0-65536	Message Digest
Vendor Affirmed[2]	CKG (SP 800-133rev2)	Counter DRBG	N/A	Symmetric key and asymmetric key seed generation in accordance with SP 800-133rev2 (Sections 4, 5.1, and 6.1) and IG D.H with B=U

Table 4 – Approved Algorithms

The module does not support any non-Approved algorithms in the Approved mode. (It supports neither ***Non-Approved Algorithms Allowed in the Approved Mode of Operation***, *nor* ***Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed***.)

Algorithm/Function	Use/Function
DES – ECB, CBC modes	Symmetric Data Encryption/Decryption

[2] Note: The resulting symmetric key and generated seed is an unmodified output from the implemented DRBG.

Triple-DES[3]-ECB, CBC modes (non-compliant)	Symmetric Data Encryption/Decryption
Blowfish – ECB, CBC modes with 128/256-bit keys	Symmetric Data Encryption/Decryption
Serpent – ECB, CBC modes with 128/256-bit keys	Symmetric Data Encryption/Decryption
Twofish – ECB, CBC modes with 128/256-bit keys	Symmetric Data Encryption/Decryption
MD5	Message Digest Algorithm
HMAC-MD5	Data Authentication
Asymmetric Data Encryption/Decryption with 2048, 3072, and 4096-bit keys and RSA 186-2 Key Generation	Asymmetric Data Encryption/Decryption and RSA Key Pair Generation
GOST with 256-bit keys	Symmetric Data Encryption/Decryption

Table 5 – Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

3. Cryptographic module interfaces

Physical port	Logical interface[4]	Data that passes over port/interface
N/A	Data input interface	The API C calls that accept input data for processing through their arguments
N/A	Data output interface	The API C calls that return by means of their return codes or arguments generated or processed data back to the caller
N/A	Control input interface	The API C calls that are used to initialize and control the operation of the CVCL module
N/A	Status output interface	The API C calls that are used to query the status of the CVCL module. Cvcl.log file where the status is being output to after completion of initialization and pre-operational self-tests

Table 6 – Ports and Interfaces

4. Roles, services, and authentication

The module supports a Crypto Officer (CO) role. The maintenance role is not supported, nor is authentication. The module also does not support concurrent operators. The Crypto-Officer role is implicitly assumed, and the module does not provide an authentication mechanism for the supported role. The module relies on the underlying operating system to implement fine-grained access control. The OS authentication mechanisms should be enabled by the administrator of the OS.

[3] The Triple-DES algorithm was CAVP tested but does not comply with the requirements of IG C.G, so is therefore classified as a non-Approved algorithm.

[4] The module does not support a control output interface.

Role	Service	Input	Output
Crypto Officer	Initialization of the module	Process startup	CVCL.log
	Power-on self-test	Power-up	Status Success/Error
	Key Generation	Entropy data RSA modulus bits (asymmetric)	Random bits RSA object (asymmetric)
	Key Zeroisation	API Call	None
	Show Status	API Call	State
	Show Version	API Call	Version
	Symmetric Data Encryption/Decryption	Plain/Cipher text, key	Cipher/Plain text
	Digest Algorithms	Buffer	Digest
	Message Authentication	Digest	Result
	Signature Generation/Verification	Message/Signature, key	Signature, result
	Pseudo-Random Number Generation	Entropy	Buffer

Table 7 – Roles, Service Commands, Input and Output

Role	Authentication Method	Authentication Strength
Crypto Officer	Not Implemented	N/A

Table 8 – Roles and Authentication

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSP's	Indicator
Initialization of the module	The module is initialized	N/A	None	CO	N/A	N/A
Power-on self-test	The module's pre-operational self-tests	N/A	None	CO	N/A	**API:** cvcl_check_state2()

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSP's	Indicator
	and conditional known answer tests are run					**Return code:** 7- Error state 6 – Not initialized 0 – Initialized If the return code is none of the above, then self-tests completed successfully
Key Generation	Generation of symmetric and asymmetric keys	• AES ECB, CBC 128, 256-bit, HMAC SHA-1, HMAC SHA2-256, HMAC-SHA2-512 • RSA 2048, 3072, 4096-bit KeyGen (FIPS 186-4) • CTR_DRBG with AES 256-bits • CKG	Symmetric AES Keys, RSA Public Keys, RSA Private Keys	CO	GRWE	**API:** cvcl_aes_is_fips_approved() cvcl_sha512_is_fips_approved() cvcl_rsa_sign_is_fips_approved () **Return code:** 1- Approved, 0 - Non-approved
Key Zeroisation	Zeroises the SSPs of the module by process completion, power cycle or the module's destroy function	N/A	Symmetric AES Keys, RSA Public Keys, RSA Private Keys, HMAC Keys, AES-CTR DRBG V and Key, AES-CTR DRBG Seed, Entropy Input	CO	Z	N/A
Show Status	Queries the current state of the module	N/A	N/A	CO	N/A	**API:** cvcl_check_state2() **Return code:** 7- Error state 6 – Not initialized 0 - Initialized
Show Version	Queries the version of the module	N/A	N/A	CO	N/A	**API:** cvcl_get_version()

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSP's	Indicator
Symmetric Data Encryption/ Decryption	Encrypts or decrypts data	• AES-ECB, CBC modes with 128-bit and 256-bit keys	Symmetric AES Keys	CO	RE	API: cvcl_aes_is_fips_approved() **Return code:** 1 - Approved, 0 – Non-approved
Digest Algorithms	Generates a message digest	• SHA-1 • SHA2-256 • SHA2-512	None	CO	N/A	**API:** cvcl_sha1_is_fips_approved() cvcl_sha256_is_fips_approved() cvcl_sha512_is_fips_approved() **Return Code:** 1 - Approved, 0 – Non-approved
Message Authentication	Generates or verifies data integrity using HMAC	• HMAC-SHA-1 • HMAC-SHA2-256 • HMAC-SHA2-512	HMAC keys	CO	RE	**API:** cvcl_sha1_is_fips_approved() cvcl_sha256_is_fips_approved() cvcl_sha512_is_fips_approved() **Return Code:** 1 - Approved, 0 - Non-approved

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSP's	Indicator
Signature Generation/Verification	Generates or verifies digital signatures using RSA	• RSA signature generation and verification (1024[6], 2048, 3072, 4096)	RSA Public Keys, RSA Private Keys	CO	RE	**API:** cvcl_rsa_sign_is_fips_approved () **Return Code:** 1 - Approved, 0 - Non-approved
Pseudo-Random Number Generation	Generates random bits using the Approved DRBG	• CTR-DRBG-AES-256	Entropy Input, AES-CTR DRBG Seed	CO	WE	**API:** cvcl_frand_is_fips_approved() **Return Code:** 1 - Approved, 0 – Non-approved
			AES-CTR DRBG V, AES-CTR DRBG Key	CO	GE	

Table 9 – Approved Services

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.

[6] 1024-bit modulo only used for signature verification

Service	Description	Algorithms Accessed	Role	Indicator
Symmetric Data Encryption/Decryption	Encrypts or decrypts data	• DES – ECB, CBC modes • Triple-DES (non-compliant)– ECB, CBC modes • Blowfish with 128-bit and 256-bit keys – ECB, CBC modes • Serpent with 128-bit and 256-bit keys – ECB, CBC modes • Twofish with 128-bit and 256-bit keys – ECB, CBC modes GOST with 256-bit keys	CO	**API:** cvcl_3des_is_fips_approved() cvcl_des_is_fips_approved() cvcl_blowfish_is_fips_approved() cvcl_serpent_is_fips_approved() cvcl_twofish_is_fips_approved() **Return code:** 0 – Non-approved
Digest Algorithms	Generates a message digest	• MD5	CO	**API:** cvcl_md5_is_fips_approved() **Return code:** 0 – Non-approved
Asymmetric Data Encryption/Decryption	Encrypts or decrypts data	• RSA with 2048, 3072 and 4096-bit keys RSA 186-2 Key Generation	CO	**API:** cvcl_rsa_encdec_is_fips_approved() **Return code:** 0 – Non-approved
Data Authentication	Generates or verifies data integrity using HMAC	• HMAC-MD5	CO	**API:** cvcl_md5_is_fips_approved() **Return code:** 0 – Non-approved

Table 10 – Non-Approved Services

5. Software/Firmware security

The module is shipped with a precomputed RSA signature file, (cvcl.dll.sig or libcvcl.so.sig). The signature is generated using an RSA EMSA-PKCS1-v1_5 4096-bit with SHA2-512 private key. The public/private RSA 4096-bit key pair is generated during manufacturing. The public key is hard coded in the module and is used for signature verification. The RSA public key used for the integrity test is not considered an SSP. As part of the software integrity check, the entire cvcl.dll Windows module or libcvcl.so Unix based module shared library is verified against this signature. If the signature verification fails, the module transitions to the error state, and finally back to the non-initialized state, where no further cryptographic operations are possible.

6. Operational environment

The module is designed to work on a General-Purpose Computer with one of the supported modifiable operating systems:

- Microsoft Windows Server 2019 (Tested on Intel Xeon R Silver 4208 with and without AES-NI)
- Red Hat Enterprise Linux 8.4 (Tested Intel Xeon R Silver 4208 with and without AES-NI)

The operating system segregates the computing environment into processes spaces. Though all processes share the same hardware resources, they are logically separated from each other by the operating system. Each process has an independent pool of virtual memory, and the module functions completely within the address space of the process that loads it. The module does not attempt to communicate with other processes or other components of the operating system, and the operating system shouldn't allow other processes to interfere with the module.

It is the administrator's responsibility to configure the operating system authentication mechanisms and ensure that only authorized users have access to the software that is loading the module.

Since the Commvault Crypto Library is a software module executing within a general-purpose computer, and the physical cryptographic perimeter is drawn through the hardware interfaces of the computer, (i.e.: Ethernet or SCSI Bus), all major data paths within the computer must be protected.

The replacement or modification of the module by unauthorized parties is explicitly prohibited. There are no restrictions on the configuration of the operational environment.

7. Physical security

Physical security is not applicable to this software module at Security Level 1.

8. Non-invasive security

The module does not implement non-invasive security measures.

9. Sensitive security parameter management

Implemented algorithms

The module provides low-level key generation and management routines for all algorithms listed in Table 4 of this Security Policy. Since the module implements both approved and non-approved algorithms, it is the responsibility of the user to ensure that only the approved services are being used (please refer to the sub-sections “Secure setup and initialization” and “Module security policy rules” for more details).

Key generation

The module provides services to generate pseudo-random symmetric and asymmetric keys and is passively receiving entropy. The module sets a function callback whenever entropy is required using “**void cvcl_set_getentropy_cb(cvcl_get_rand_t get_rand)**” API. The callback function requires a minimum of 384 bits of entropy. Symmetric keys are generated using the direct output of the CTR_DRBG. For asymmetric key generation, a provable prime method is used in accordance with FIPS 186-4 standard.

Key entry and output

The module does not import or export keys across the physical cryptographic perimeter. It is the responsibility of the application that loads the module to protect keys when they’re being exported or imported across the physical cryptographic perimeter. It is also the responsibility of the application to ensure that only Approved cryptographic algorithms are being used for key protection. The module accepts and passes keys across the cryptographic boundary as parameters via API calls.

Key storage

The module does not provide any long-term key storage. Keys stored in the NVRAM are protected from unauthorized disclosure, access or modification by the operating system that is responsible for allocating isolated and independent virtual memory for the module and the process using it.

Zeroisation of keys

All data output is inhibited during the zeroisation of SSPs. The following precautions are taken to make sure that all keys and seeds are being destroyed properly:

- CVCL module zeroises all intermediate security sensitive material.
- CVCL module zeroises all keys passed to the authorized services when the services are no longer needed and are being destroyed.
- The following API is used to call init and call destroy:

  CVCL_API void * cvcl_XXX_init(int * error_num, char * error_buf, int error_len);
  CVCL_API void cvcl_XXX_destroy(void * XXX);
  Note: “XXX” is the algorithm being used.

- The zeroisation is performed by overwriting the SSPs with zeroes.

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
Symmetric AES Keys	128, 256 bits	AES-CBC, ECB, CKG (A2412)	Internally computed using the SP 800-90Ar1 DRBG	Import and Export via API Call (Electronic Entry)	N/A	Volatile memory (plaintext)	At the end of the operation, power cycle or the destroy function	Used in data encryption/decryption and key protection
RSA Public Keys	80, 112, 128, 152 bits	RSA SigVer, CKG (A2412)	Internally computed using provable prime method from FIPS 186-4	Import and Export via API Call (Electronic Entry)	N/A	Volatile memory (plaintext)	At the end of the operation, power cycle or the destroy function	Used to perform Signature verification
RSA Private Keys	112, 128, 152 bits	RSA KeyGen, SigGen, CKG (A2412)	Internally computed using provable prime method from FIPS 186-4	Import and Export via API Call (Electronic Entry)	N/A	Volatile memory (plaintext)	At the end of the operation, power cycle or the destroy function	Used to perform Signature generation
HMAC keys	112 - 256 bits	HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-512, CKG (A2412)	Internally computed using the SP 800-90Ar1 DRBG	Import and Export via API Call (Electronic Entry)	N/A	Volatile memory (plaintext)	At the end of the operation, power cycle or the destroy function	Used to generate and verify HMAC
AES-CTR DRBG V (IG D.L compliant)	256 bits	Counter DRBG (A2412)	Internally computed	The internal state is not imported or exported	N/A	Volatile memory (plaintext)	The internal state is zeroised by the cvcl_destroy() command	Used in random bit generation

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
AES-CTR DRBG Key (IG D.L compliant)	256 bits	Counter DRBG (A2412)	Internally computed	The internal state is not imported or exported	N/A	Volatile memory (plaintext)	The internal state is zeroised by the cvcl_destroy() command	Used in random bit generation
Entropy Input (IG D.L compliant)	384 bits	Counter DRBG (A2412)	N/A	Entropy input Imported via API Call (Electronic Entry)	N/A	Volatile memory (plaintext)	The entropy input data is zeroised at the end of the operation	Used in random bit generation
AES-CTR DRBG Seed (IG D.L compliant)	256 bits	Counter DRBG (A2412)	Internally computed	The internal state is not imported or exported	N/A	Volatile memory (plaintext)	The internal state is zeroised by the cvcl_destroy() command	Used in random bit generation
DRBG Output	256 bits	Counter DRBG (A2412)	AES-CTR DRBG	Import and Export via API Call (Electronic Entry)	N/A	Volatile memory (plaintext)	At the end of the operation, power cycle or the destroy function	Used as seed in asymmetric key pair generation or symmetric key

Table 11 – SSPs

Entropy sources	Minimum number of bits of entropy	Details
Passive (external) [IG 9.3.A, Scenario 2b]	384-bits	The module is passively receiving the entropy while exercising no control over the amount or the quality of the obtained entropy. The following caveat applies to the module: No assurance of the minimum strength of generated SSPs (e.g., keys)

Table 12 – Non-Deterministic Random Number Generation Specification

10. Self-tests

The module is a software shared library that can be loaded by one of the Commvault applications. Before the library can be used in cryptographic operations, it must be initialized. The loading application performs initialization by calling the CVCL initialization function, which internally executes the self-tests. Upon successful completion, the library is switched to the Initialized State.

While self-tests are running, data input and data output interfaces are inhibited. To run the self-tests on-demand, the module must be re-instantiated. Prior to the execution of the pre-operational software integrity test, a conditional CAST is performed for RSA.

The module performs the following self-tests:

Pre-operational self-tests:

Software Integrity Test: RSA 4096 with SHA-512 signature verification computed on the cvcl.dll file for Windows and libcvcl.so file for UNIX and makes sure that the signature matches the precomputed value provided in the signature file (cvcl.dll.sig or libcvcl.so.sig). To initiate the software integrity test on-demand, explicitly call the following API that performs the test:

api - cvcl_verify_sig(dll_fname, error_buf, error_len)

Conditional self-tests:

Conditional Cryptographic Algorithm Self-tests (CAST) (#A2412):

- AES-CBC 256 bits Encrypt KAT
- AES-CBC 256 bits Decrypt KAT
- AES-ECB 256 bits Encrypt KAT
- AES-ECB 256 bits Decrypt KAT
- RSA 2048 bits SHA2-256 Sign KAT
- RSA 3072 bits SHA2-256 Sign KAT
- RSA 4096 bits SHA2-256 Sign KAT
- RSA 1024 bits SHA2-256 Verify KAT
- RSA 2048 bits SHA2-256 Verify KAT

- RSA 3072 bits SHA2-256 Verify KAT
- RSA 4096 bits SHA2-256 Verify KAT
- SHA-1 KAT
- SHA2-256 KAT
- SHA2-512 KAT
- HMAC-SHA-1 KAT
- HMAC-SHA2-256 KAT
- HMAC-SHA2-512 KAT
- CTR_DRBG (AES256) KAT (Generate, Instantiate and Reseed)

Conditional Pairwise Consistency Self-tests:

- Pairwise consistency test when generating RSA key pairs (for signature generation/verification)

At power-up, the pre-operational and conditional self-tests are carried out. If any self-test fails, the module transitions to the error state and back to the non-initialized state, where the applicable error code (6) is returned to the program that attempted to load the module. While the self-tests are being executed, and when the module is in an error state, the data input and data output interfaces are inhibited.

When conditional self-tests fail, the module transitions to the error state (return code 7), and no further cryptographic operations can be executed until the controlling application successfully reinitializes the module.

In the event of a pre-operational or conditional self-test failure, the user should retry initialization and if this doesn't fix the error, uninstall, and re-install the module from the original installation media.

11. Life-cycle assurance

This section describes design, coding and testing practices used at Commvault during the production of the module. Please refer below for Crypto Officer and Commvault administrative end user guides detailing the correct installation, configuration, management, and usage procedures that are required for the secure operation of the module.

Secure setup and initialization

The module is bundled with the Commvault software platform, distributed and installable using the Commvault download manager. To install the module correctly, you must follow the installation instructions provided in the Commvault documentation. You must have enough space on the hard disk, enough memory, and possess administrative privileges on the computer where the product is being installed. The operating system must have an authentication mechanism configured that will allow only authorized users to use the product (and module) after it has been installed. The host operating system must be configured by the administrator such that only the administrator's account is enabled, while all other user accounts (including guest accounts) are disabled.

Once the Commvault product is installed, there are no other procedures for secure installation, initialization and startup of the cryptographic module. Once the hardware platform is powered on, the module will be in initialized state.

To validate that the module has been installed successfully and is operating in the approved mode, the following action is required:

- Navigate to the software's Base folder.
- Check the module version from the command line by opening the command line window on the GPC where Commvault is installed. Execute cvcl_ver.exe executable (Windows) or cvcl_ver (UNIX) and make sure that it displays "CVCL version 3.0" of the module. The module version 3.0 is returned by the API cvcl_get_version().

Check that the module successfully loaded and entered the approved mode by opening the cvcl.log file on the applicable computers and searching for the message "CVCL: Running in FIPS". Also, check that the individual algorithms are reporting whether they are approved algorithms or not.

Please note that the module identifier is also obtained by accessing the cvcl.log mentioned above, which contains "CVCL" in the first line of output "CVCL: Running in FIPS". It can be used to correlate the output "CVCL" to "Commvault Crypto Library", which is the name of the module.

The following is an example of the possible log output:

```
8924  3d68  06/28 16:44:21 ### Cvcl::init() - CVCL: Running in FIPS
8924  3d68  06/28 16:44:21 ### Cvcl::init() - 3DES: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - AES: FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - Blowfish: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - DES: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - GOST: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - RSA sign/verify: FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - RSA encrypt/decrypt: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - Serpent: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - Twofish: Not FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - SHA1: FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - SHA256: FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - SHA512: FIPS-approved
8924  3d68  06/28 16:44:21 ### Cvcl::init() - MD5: Not FIPS-approved
```

Figure 3 – Log Output Example

Configuration management

A Concurrent Versioning System (CVS) is used by the module development team for configuration management, change control and version control. All modifications made to the module are logged, along with the comments that accompanied them, and a complete history of changes is maintained in the CVS repository. Every individual change is assigned a unique version number. In addition, for every

software release, a release-specific symbolic tag is assigned to all source files and documents thereby creating a snapshot of the components included in the release.

Development

The module is written using high level language “C” with several time-critical pieces optimized using architecture-specific low-level assembler instructions on some platforms (AES-NI).

A software defect tracking software called Silk Radar is being used to log defects discovered during testing and to keep track of their resolution by the development team.

As part of the build sequence the module is compiled, linked, self-tested and signed. For every build released to the system test, a series of acceptance tests is conducted to verify at a higher level that cryptographic operations are working as expected.

End-of-life

The module can be sanitized by resetting the host platform on which it is run, zeroising all SSPs.

12. Mitigation of other attacks

The module does not mitigate against any specific attacks outside of the scope of FIPS 140-3.