Focus Systems Corporation

MonoCrypt AES Enhanced Crypto Library

FIPS 140-3 Non-Proprietary Security Policy

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

MonoCrypt AES Enhanced Crypto Library (MonoCrypt AES) is Software module. It runs on general-purpose computing system.
This module complies with FIPS140-3 requirement level 1 and is defined as Ver3.0.0.
The requirements of FIPS140-3 and the security levels for module are as follows.

1.2 Security Levels

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

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:
This product is a cryptographic software library. Applications running on a general-purpose computing system can use various cryptographic services by calling the C or C++ language API. This product is currently configured as a standalone module in the form of a shared library.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The cryptographic boundary for the Module is comprised of the following six files: CSL.h, CSL_AES.h, CSL_AES_KEY_WRAP.h, MonoCrypt.lib, MonoCrypt.dll, and libMonoCrypt.so.

Among these, the integrity test is performed only on the executable binary files, MonoCrypt.dll and libMonoCrypt.so, when they are loaded into RAM for execution. The header files (CSL.h, CSL_AES.h, CSL_AES_KEY_WRAP.h) and the import library (MonoCrypt.lib) are excluded from the integrity test. This exclusion is justified because these files are development-time resources used solely for compiling and linking the calling application; they do not contain any

executable cryptographic logic or sensitive security parameters that reside within the operational RAM during the module's execution.

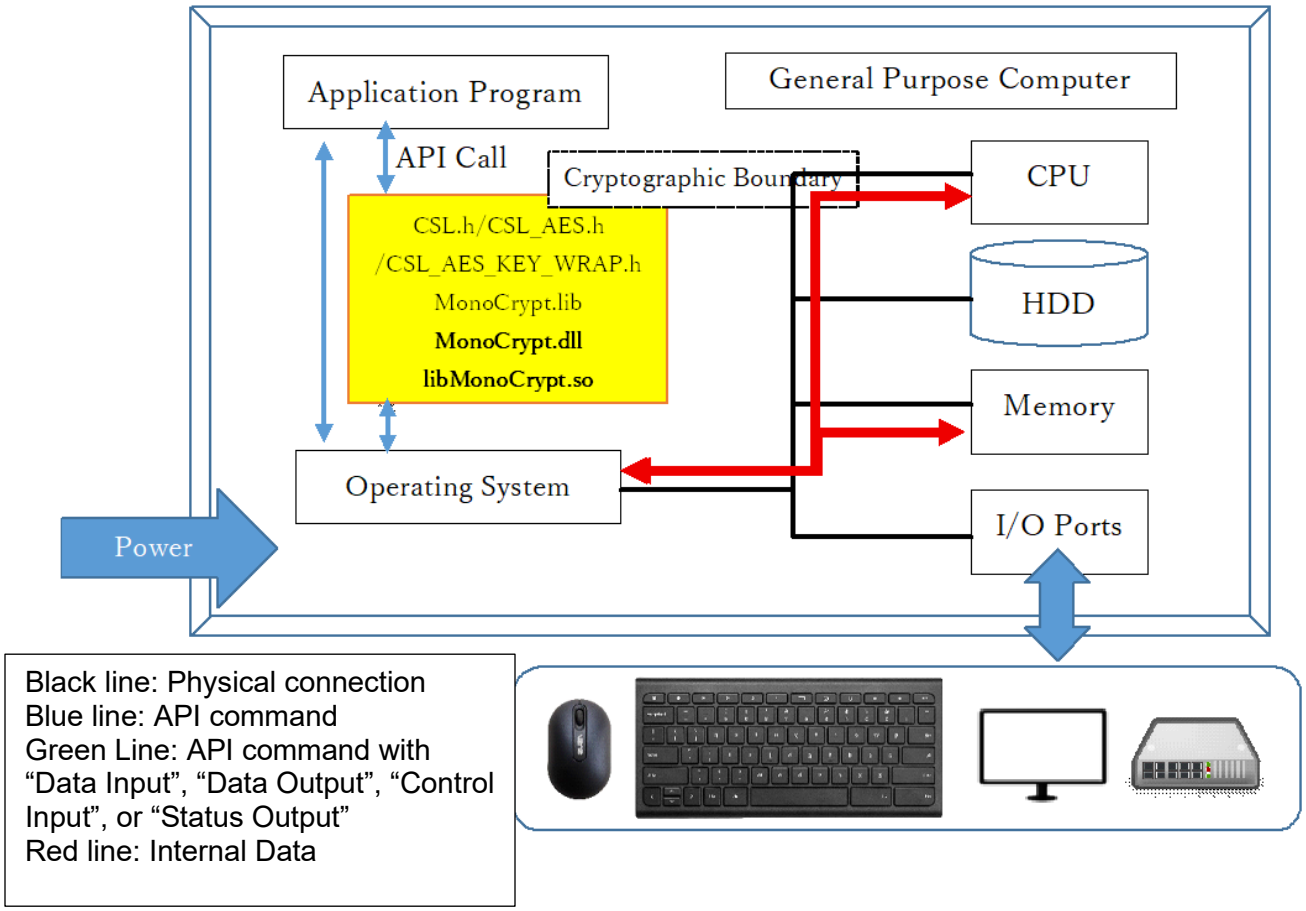

Figure 1: Block Diagram

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):
The "MonoCrypt.dll" is a module for Windows. It is separated into 32-bit and 64-bit versions.
The "libMonoCrypt.so" is a module for Linux and AIX. A 64-bit version is provided.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
MonoCrypt.dll (Windows_32bit)	3.0.0		HAMC-SHA2-256
MonoCrypt.dll (Windows_64bit)	3.0.0		HAMC-SHA2-256

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libMonoCrypt.so (AIX_64bit)	3.1.0		HAMC-SHA2-256
libMonoCrypt.so (Linux_64bit)	3.0.0		HAMC-SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Windows11 Pro	Dell Optiplex 5090	11th Gen Intel® Core(TM) i7-11700 @ 2.50GHz	No	N/A	3.0.0
Windows Server 2025	Dell PowerEdge R430	Intel® Xeon® CPU E5-2620 v4 @ 2.10GHz	No	Hyper-V10.0.14393.0 on Windows Server 2016 Standard	3.0.0
Redhat Enterprise Linux 10.1	Dell Optiplex 5090	11th Gen Intel® Core(TM) i7-11700 @ 2.50GHz	No	Hyper-V 10.0.22621.4249 on Windows 11 Pro	3.0.0
AIX 7.3	IBM Power System S814+	POWER8 3.026 GHz	No	N/A	3.0.0

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

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

N/A for this module.

2.3 Excluded Components

N/A for this module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	When the module is loaded and passes the pre-operational self-test, it operates in the Approved mode of operation.	Approved	CSL_OutputApprovalMode service called after a service call returns 0x00000102.
Non-Approved mode	When a service is called under the parameter conditions of the Non-Approved mode described in Section 2.5, the module implicitly transitions to the Non-Approved mode of operation.	Non-Approved	CSL_OutputApprovalMode service called after a service call returns 0x00000103.

Table 4: Modes List and Description

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A8008, A8009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CTR	A8008, A8009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - No Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A8008, A8009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-KW	A8008, A8009	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 192, 256, 320, 512, 1024	SP 800-38F
AES-KWP	A8008, A8009	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 8, 24, 64, 96, 1024	SP 800-38F
HMAC-SHA2-256	A8008, A8009	MAC - MAC: 128, 192, 256 Key Length - Key Length: 160, 192, 512, 576, 640	FIPS 198-1
SHA2-256	A8008, A8009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
AES-KW per RFC3394	A key wrapping functionality per RFC3394
AES-KWP per RFC5649	A key wrapping functionality per RFC5649

Table 6: Non-Approved, Not Allowed Algorithms

If a non-null arbitrary value is set to “Integrity data” parameter for AES-KW or AES-KWP, the algorithm becomes Non-Approved, Not Allowed Algorithms.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
SW Integrity test	MAC	Used in the software integrity test		HMAC-SHA2-256: (A8008, A8009) SHA2-256: (A8008, A8009)
AES	BC-UnAuth	AES - Unauthenticated Cipher		AES-CBC: (A8008, A8009) AES-CTR: (A8008, A8009) AES-ECB: (A8008, A8009)
KeyWrap	BC-Auth	AES KeyWrap Cipher		AES-KW: (A8008, A8009) AES-KWP: (A8008, A8009)

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

2.8 RBG and Entropy

N/A for this module.

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.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Cryptographic Function API Input Data
N/A	Data Output	Cryptographic Function API Output Data
N/A	Control Input	API function calls
N/A	Status Output	The following data elements are output from the support function APIs: -Module SelfTest Status Information -Module Approved mode Status Information - Module Version Information

Table 8: Ports and Interfaces

This cryptographic module provides only logical interfaces via API. It does not provide any direct interface to the physical ports. The logical interface is a C language application programming interface (API), and its mapping is shown in Ports and Interfaces Table. When the module is performing self-tests or is in an error state, all output on the logical data output interface is prohibited.

The logical interfaces for the MonoCrypt AES cryptographic module are defined by the input arguments of its API functions. These interfaces are categorized into the following types of information flow:

- Data Input: Plaintext and ciphertext to be processed.
- Control Input: Control parameters that govern the module's operations.
- CSP Input: Critical Security Parameters (CSPs), such as key information. Data output during CSP Input is inhibited because the control is not returned to the superior application until the API processing is complete.

Data Flow and Memory Management
These inputs are passed to the cryptographic module via buffer areas in memory defined by the superior application. Similarly, the Data Output (such as processed results) and Status Output (such as error codes or operation status) are returned to the application through memory buffers pre-allocated by the application.
The application is responsible for subsequently directing this output to various physical devices as required.
The data flow is depicted in the block diagram provided in Section 2.1.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 9: Roles

Since the Module only defines the Crypto Officer (CO) role, all services are implicitly permitted to and associated with the CO role.
The Module operates in a logically separated single-user environment by leveraging the process separation mechanisms of the General-Purpose Operating System (GPOS). Even in scenarios where multiple users or processes are active on the OS, each instance of the Module is confined to a specific process memory space, ensuring that only one operator at a time can access that specific instance. Consequently, the Module does not support concurrent operators within a single cryptographic boundary instance.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
AES_Enc	AES Encryption	CSL_OutputApprovalMode service	-key -PlainText	Cipher Text	AES	Crypto

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		called after this service returns 0x00000102	or CipherText -iv - Algorithm mode			Officer - AES Key: W,E, Z
AES_Dec	AES Decryption	CSL_OutputApprovalMode service called after this service returns 0x00000102	-key - PlainText or CipherText -iv - Algorithm mode	PlainText	AES	Crypto Officer - AES Key: W,E, Z
AES_Stream Initialize	Initialize in AES_Stream	CSL_OutputApprovalMode service called after this service returns 0x00000102	-key - PlainText or CipherText -iv - Algorithm mode	Object ID	AES	Crypto Officer - AES Key: W,E
AES_Stream Enc/Dec	Encryption or Decryption in AES_Stream	CSL_OutputApprovalMode service called after this service returns 0x00000102	-Object ID -PlainText or CipherText	Cipher Text or PlainText	AES	Crypto Officer - AES Key: E
AES_Stream Final	Finalization in AES_Stream	CSL_OutputApprovalMode service called after this service returns 0x00000102	Object ID	Cipher Text or PlainText	AES	Crypto Officer - AES Key: E,Z
AES_Stream Release Object	SSP Release in AES_Stream	CSL_OutputApprovalMode service called after this service returns 0x00000102	Object ID	-	None	Crypto Officer - AES Key: Z
AES_GetMAxLength	Calculate the maximum byte length	CSL_OutputApprovalMode service called after this	-Object ID -Input data length	Output data length	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	of the output data from AES encryption or decryption relative to the data byte length of the input data	service returns 0x00000102				
AES_KeyWrap	Encryption In AES_KeyWrap	CSL_OutputApprovalMode service called after this service returns 0x00000102	- key - PlainText or CipherText - Algorithm mode(KW/KWP)	Cipher Text	KeyWrap	Crypto Officer - KeyWrap Key: W,E, Z
AES_KeyUnWrap	Decryption In AES_KeyWrap	CSL_OutputApprovalMode service called after this service returns 0x00000102	- key - PlainText or CipherText - Algorithm mode(KW/KWP)	PlainText	KeyWrap	Crypto Officer - KeyWrap Key: W,E, Z
OnDemondSelftest	Selftest	OutputStatus service called after this service returns 0	-	-	SW Integrity test	Crypto Officer
OutputStatus	Get Status Indicator	-	-	Module SelfTest Status Information	None	Crypto Officer
ShowVersion	Get Module version	-	-	Module Version Information	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
CSL_OutputApprovalMode	Get Indicator for previously executed KeyWrap &AES	-	-	Module Approved mode Status Information	None	Crypto Officer
FIPS140_2_Zeroization	Zeroization executed before returning from AES_Enc service, AES_Dec service, AES_Stream Final service, AES_Stream Release Object service, AES_KeyWrap service, and AES_KeyUnWrap service.	-	-	-	None	Crypto Officer - AES Key: Z - KeyWrap Key: Z

Table 10: Approved Services

4.4 Non-Approved Services

Name	Description	Algorithms	Role
AES_KeyWrap_Non-Approved	Encryption or Decryption in AES_KeyWrap per RFC3394 or RFC5649	AES-KW per RFC3394 AES-KWP per RFC5649	Crypto Officer

Table 11: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not support external software loading.

5 Software/Firmware Security

5.1 Integrity Techniques

Since this cryptographic module is a software module, it handles software security.
To ensure software security within the cryptographic module, the integrity of the modules belonging to the cryptographic boundary except for CSL.h, CSL_AES.h, CSL_AES_KEY_WRAP.h, and MonoCrypt.lib is checked upon module load into the RAM to be executed as follows.

Since this cryptographic module is a software module, it handles software security requirements. To ensure software security within the cryptographic module, an integrity test is performed on the binary file that constitutes the logical cryptographic boundary.
The integrity of MonoCrypt.dll (and libMonoCrypt.so for UNIX-like systems) is checked upon module load into the RAM to be executed. The reference files used for application development, specifically header files (CSL.h, CSL_AES.h, CSL_AES_KEY_WRAP.h) and the import library (MonoCrypt.lib), are excluded from this integrity check. This is because these files are utilized only during the compilation and linking phases of the calling application and do not contain the executable cryptographic logic that resides within the cryptographic boundary during runtime.

・A MAC value is provided for integrity testing of this software library, calculated using the HMAC-SHA2-256 algorithm.
・This calculated MAC value for integrity testing is embedded as part of the library.
・The cryptographic library performs an integrity check when executing a pre-operational self-test upon loading or an on-demand self-test. This check compares the stored integrity test MAC value with the library's integrity test MAC value calculated in real-time using HMAC-SHA2-256.

Header files are excluded from the integrity test because they contain only function declarations, constant definitions, and structure definitions. As text-based source files used only during compilation, they do not contain executable code and are not loaded into RAM during module execution.

5.2 Initiate on Demand

OnDemondSelftest service is provided to execute an integrity test, allowing the state of the cryptographic module to be checked at any time.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

The Module runs in a Modifiable Operational Environment as defined by FIPS 140-3. The Module has been tested for operation on the specific operating systems and hardware platforms identified in Section 2.2.

The Module is implemented as a software-only cryptographic module in the form of a shared library (a DLL on Windows and a Shared Object on Linux and AIX), which operates under the control of a General-Purpose Operating System (GPOS). While the Module is executing, the GPOS is responsible for maintaining a logically separated, single-user operational environment for the process. The Module performs the functions described in Section 4 (Roles and Services).

7 Physical Security

N/A for this module.

8 Non-Invasive Security

N/A for this module.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Random Access Memory	Dynamic

Table 12: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Key Input	RAM of the superior application	RAM	Plaintext	Manual	Electronic	

Table 13: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Zero OverWriting	In the zeroization Internal function, the relevant	Zeroization is performed on the variables (buffers)	Zeroization is implemented as an internal function within the cryptographic services and,

Zeroization Method	Description	Rationale	Operator Initiation
	memory area is zeroed out (cleared) using the memset function. Following this zeroization, the buffer is then released.	that contain Critical Security Parameters (CSPs) when they are no longer needed, typically near the end of the cryptographic service execution.	therefore, does not require discretionary execution by the operator. Note that zeroization for AES_Stream Initialize service and AES_Stream Encryption/Decryption service is performed in AES_Stream Final service or AES_Stream Release Object service as a sequential service.

Table 14: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	AES key used for encryption and decryption	128,192.256 bits - 128,192.256 bits	Symmetric Key - CSP	Outside the module		AES
KeyWrap Key	KeyWrap key used for AES key encryption and AES key decryption	128,192.256 bits - 128,192.256 bits	Symmetric Key - CSP	Outside the module		KeyWrap

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	Key Input	RAM:Plaintext	input until Zero OverWriting	Zero OverWriting	
KeyWrap Key	Key Input	RAM:Plaintext	input until Zero OverWriting	Zero OverWriting	

Table 16: SSP Table 2

The Sensitive Security Parameters (SSPs) for the cryptographic module are input by the superior application, which must execute with the same privilege level as the cryptographic module.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
SW Integrity Test	HMAC-SHA2-256	MAC verification	SW/FW Integrity	OutputStatus service returns 0	The integrity of the modules belonging to the cryptographic boundary is checked upon module load into the RAM to be executed

Table 17: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB Encryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-ECB Decryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CBC Encryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CBC Decryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CTR Encryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						OnDemondSelftest service
AES-CTR Decryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KW Encryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KW Decryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KWP Encryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KWP Decryption (A8008)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
SHA2-256 (A8008)	Message:240bits	KAT	CAST	OutputStatus service returns 0	Message Digest Generation	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
HMAC-SHA2-	Key: 160bits	KAT	CAST	OutputStatus service returns 0	MAC verification	Performed on module load into the RAM to be

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
256 (A8008)						executed before the Integrity test or by calling OnDemondSelftest service
AES-ECB Encryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-ECB Decryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CBC Encryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CBC Decryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CTR Encryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-CTR Decryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-KW Encryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KW Decryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KWP Encryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Encryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
AES-KWP Decryption (A8009)	Key:128/192/256bits	KAT	CAST	OutputStatus service returns 0	Decryption	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
SHA2-256 (A8009)	Message:240bits	KAT	CAST	OutputStatus service returns 0	Message Digest Generation	Performed on module load into the RAM to be executed or by calling OnDemondSelftest service
HMAC-SHA2-256 (A8009)	Key: 160bits	KAT	CAST	OutputStatus service returns 0	MAC verification	Performed on module load into the RAM to be executed before the Integrity test or by calling OnDemondSelftest service

Table 18: Conditional Self-Tests

The Conditional Self-Tests are executed as KATs upon module load into the RAM to be executed.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SW Integrity Test	MAC verification	SW/FW Integrity	On demand	Manually

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB Encryption (A8008)	KAT	CAST	On demand	Manually
AES-ECB Decryption (A8008)	KAT	CAST	On demand	Manually
AES-CBC Encryption (A8008)	KAT	CAST	On demand	Manually
AES-CBC Decryption (A8008)	KAT	CAST	On demand	Manually
AES-CTR Encryption (A8008)	KAT	CAST	On demand	Manually
AES-CTR Decryption (A8008)	KAT	CAST	On demand	Manually
AES-KW Encryption (A8008)	KAT	CAST	On demand	Manually
AES-KW Decryption (A8008)	KAT	CAST	On demand	Manually
AES-KWP Encryption (A8008)	KAT	CAST	On demand	Manually
AES-KWP Decryption (A8008)	KAT	CAST	On demand	Manually
SHA2-256 (A8008)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A8008)	KAT	CAST	On demand	Manually
AES-ECB Encryption (A8009)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB Decryption (A8009)	KAT	CAST	On demand	Manually
AES-CBC Encryption (A8009)	KAT	CAST	On demand	Manually
AES-CBC Decryption (A8009)	KAT	CAST	On demand	Manually
AES-CTR Encryption (A8009)	KAT	CAST	On demand	Manually
AES-CTR Decryption (A8009)	KAT	CAST	On demand	Manually
AES-KW Encryption (A8009)	KAT	CAST	On demand	Manually
AES-KW Decryption (A8009)	KAT	CAST	On demand	Manually
AES-KWP Encryption (A8009)	KAT	CAST	On demand	Manually
AES-KWP Decryption (A8009)	KAT	CAST	On demand	Manually
SHA2-256 (A8009)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A8009)	KAT	CAST	On demand	Manually

Table 20: Conditional Periodic Information

The module implements an OnDemondSelftest service and OutputStatus service to allow the user to execute the self-tests and obtain the operational status at any time.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Known Answer Self-test Error	The Error State resulting from the Conditional Self-Test failure of the cryptographic algorithm self-test.	An error occurred during the cryptographic algorithm test.	Reload the Module into Memory Power Cycle or Reboot	OutputStatus service returns 0x40000000

Name	Description	Conditions	Recovery Method	Indicator
Integrity Self-test Error	The Error State resulting from the Pre-Operational Self-Test failure of the integrity self-test.	An error occurred during the module's integrity test	Reload the Module into Memory Power Cycle or Reboot	OutputStatus service returns 0x80000000

Table 21: Error States

10.5 Operator Initiation of Self-Tests

The operator can initiate the Self-Tests by calling the OnDemondSelftest service (inclusive of software integrity verification).

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

(1) Windows Version

1. Folder Access Permissions Setting
   The cryptographic administrator sets the user access permissions for the folder where the module is to be placed.
   - Permission granting must follow the Windows OS folder access permission settings.
   - Execute, Read, and Write permissions are required.
2. Module Placement (Deployment)
   Copy the module to the specified folder.
   - Restrictions apply to the module's placement. Refer to "(4) Restrictions".
3. Module Access Permissions Setting
   The same access permissions as the destination folder are required for the execution of this module.
   - Access permissions can be set using methods such as the following example:
     - *(Example) Setting via Properties:* Set the same access permissions as the destination folder for the OS user who acts as the cryptographic administrator role, under the module's "Properties" → "Security" tab.

. Module Activation (Startup)
   This cryptographic module does not operate as a standalone DLL. The cryptographic functionality is activated by executing a user application that has been compiled and linked to utilize the module.
   - Therefore, the execution of the user application constitutes the activation of this cryptographic module.
     - *(Example) For a command-line executable user application:*
       - User application name: Encrypt.exe
       - Installation folder: c:\Programfiles\Focus
       - i) Folder Navigation: Execute the following command in the command prompt to move to the location of the executable file:
       - cd c:\Programfiles\Focus
       - ii) Program Startup: Execute the following command in the command prompt:
       - c:\Programfiles\Focus>Encrypt.exe

- When the above command is executed, and this cryptographic module is loaded and the self-test succeeds, the cryptographic functionality becomes available to the user application.

(2) AIX Version

1. Directory Access Permissions Setting
   The cryptographic administrator sets the user access permissions for the directory where the module is to be placed.
   - Permission granting must follow the AIX OS directory access permission settings.
   - Execute, Read, and Write permissions are required.
2. Module Copying
   Copy the module to the specified directory.
   - Restrictions apply to the module's placement. Refer to "(4) Restrictions".

3. Module Access Permissions Setting
   The same access permissions as the destination directory are required for the execution of this module.
   - Access permissions can be set using methods such as the following example:
     - *(Example) Executing the permission setting command (chmod):*
     - chmod 755 ./home/application/libMonoCrypt.so
4. Library Path Specification
   Depending on the environment, it may be necessary to set the library path as an environment variable to enable the user application to locate the module. Since there are multiple ways to set the library path, configure it using the method recommended for the specific OS.
   - *(Example) Setting with LIBPATH:*
   - export LIBPATH=/home/application/libMonoCrypt.so:$LIBPATH
   - The use of an absolute path is highly recommended.
5. Module Activation (Startup)
   This cryptographic module does not operate as a standalone shared object. The cryptographic functionality is activated by executing a user application that has been compiled and linked to utilize the module.
   - Therefore, the execution of the user application constitutes the activation of this cryptographic module.
     - *(Example) For a command-line executable user application:*
       - User application name: Crypt.exe
       - Installation directory: /home/application/Focus
       - i) Directory Navigation: Execute the following command in the shell to move to the location of the executable file:
       - $ cd /home/application/Focus
       - ii) Program Startup: Execute the following command in the shell:
       - $ ./Crypt.exe
     - When the above command is executed, and this cryptographic module is loaded and the self-test succeeds, the cryptographic functionality becomes available to the user application.

(3) Linux Version

1. Directory Access Permissions Setting
   The cryptographic administrator sets the user access permissions for the directory where the module is to be placed.
   - Permission granting must follow the Linux OS directory access permission settings.

- Execute, Read, and Write permissions are required.

2. Module Copying
   Copy the module to the specified directory.
- Restrictions apply to the module's placement. Refer to "(4) Restrictions".

3. Module Access Permissions Setting
   The same access permissions as the destination directory are required for the execution of this module.
- Access permissions can be set using methods such as the following example:
  - *(Example) Executing the permission setting command (chmod):*
  - chmod 755 ./user/application/libMonoCrypt.so

4. Library Path Specification
   Depending on the environment, it may be necessary to set the library path as an environment variable to enable the user application to locate the module. Since there are multiple ways to set the library path, configure it using the method recommended for the specific OS.
- *(Example) Setting with LD_LIBRARY_PATH:*
- export LD_LIBRARY_PATH=/application/libMonoCrypt.so:$LD_LIBRARY_PATH

5. Module Activation (Startup)
   This cryptographic module does not operate as a standalone shared object. The cryptographic functionality is activated by executing a user application that has been compiled and linked to utilize the module.
- Therefore, the execution of the user application constitutes the activation of this cryptographic module.
  - *(Example) For a command-line executable user application:*
    - User application name: Crypt.exe
    - Installation directory: /user/application/Focus
    - i) Directory Navigation: Execute the following command in the shell to move to the location of the executable file:
    - $ cd /user/application/Focus
    - ii) Program Startup: Execute the following command in the shell:
    - $ ./Crypt.exe
  - When the above command is executed, and this cryptographic module is loaded and the self-test succeeds, the cryptographic functionality becomes available to the user application.

(4) Restrictions

The placement location of this module is generally arbitrary, but the following restrictions apply.

- This module cannot be used when placed in a folder or directory on a different drive from the user application.
- If this module and the user application are not located in the same folder or directory, since this module performs a pre-operational self-test, it may take a significant amount of time before this module becomes available.
- Exclusive to the AIX version, the cryptographic library MUST be located in either /usr/lib, /home, or the current directory. Furthermore, duplicating and placing the library with the same name in multiple different locations is strictly prohibited.

11.2 Administrator Guidance

The Crypto Officer is responsible for installing the module and granting usage rights.

For the installation of the module, please refer to the 11.1 Install, Initialization, and Startup Procedures guidance.
The Crypto Officer may utilize cryptographic functions in both approved and non-approved modes of operation.
The Cryptographic Officer (CO) may utilize various support functions to aid in the cryptographic operations. Examples of available support functions include: performing self-tests and retrieving the result status, retrieving the operational mode status, and retrieving version information.

11.3 Non-Administrator Guidance

N/A for this module.

11.4 Design and Rules

1. The module shall provide a Cryptographic Officer role.
2. The following configuration is required for the Key Wrap function to operate in the Approved Mode within this module:
   Integrity (ig) argument must be set to NULL.

11.5 Maintenance Requirements

N/A for this module.

11.6 End of Life

The method for disposing of the Cryptographic Module depends on the specific Operating System (OS).
Fundamentally, the module shall be removed using the deletion commands provided by the respective OS, ensuring that the data in the hard disk drive (HDD) area is overwritten to prevent restoration. This process constitutes the zeroization of the Module's non-volatile storage components (if any) and fulfills the End-of-Life requirements for the software module.

(1) Windows Version

1.Standard File Deletion

The operator shall delete the Module file(s) using the standard operating system method (e.g., selecting the file and pressing the Delete key), and subsequently empty the Recycle Bin.

2.Execute the cipher Command for Secure Wipe

After the standard deletion, the operator must execute the cipher command to securely overwrite the free space on the drive where the file was deleted. This ensures the permanent erasure of the deleted data.

Launch Command Prompt or PowerShell as an Administrator.

Input the following command, replacing C: with the letter of the drive from which the file was deleted, and execute it:

(Example) cipher /w:C:

This command performs a secure wipe of the free disk space, effectively ensuring the complete deletion of files hidden in the space made available after the normal deletion process.

(2) AIX version

The operator shall use the OS dd command to overwrite the module file (libMonoCrypt.so) with zero or random data, followed by the rm command for final deletion.

1. Secure Overwrite using dd command (Zeroization):
   The dd command is used to overwrite the file content with data from a source of random data (/dev/urandom), ensuring the file's data is irretrievably erased:

   (Example) dd if=/dev/urandom of=libMonoCrypt.so bs=1M conv=notrunc

   - if=/dev/urandom: Specifies the input file source of random data (or /dev/zero for zeros).
   - of=libMonoCrypt.so: Specifies the output file, which is the module file.
   - bs=1M: Sets the block size to 1 megabyte (for faster operation).
   - conv=notrunc: Ensures the output file is not truncated, overwriting only the existing content.
2. Final Deletion using rm command:
   After the secure overwrite (zeroization) is complete, the file is removed from the file system:

   (Example) rm libMonoCrypt.so

   This combined process ensures the irretrievable zeroization and removal of the Cryptographic Module's software components from the disk, satisfying the End-of-Life requirements.

(3) Linux Version

The operator shall use the shred command provided by the OS to securely delete the shared library file, specifically referencing the file name such as libMonoCrypt.so.
Command Execution: Execute the shred command with appropriate options to ensure multiple passes of overwriting are performed:

(Example) shred -n 10 -uvz libMonoCrypt.so

This command performs the following actions for secure erasure:
-n 10: Overwrites the file contents 10 times with non-repeatable patterns.
-u: Deallocates and removes the file after overwriting.
-v: Shows the progress of the operation (verbose).
-z: Adds a final overwrite with zeros to mask the shredding.

This secure deletion process ensures the irretrievable zeroization of the Cryptographic Module's software components from the disk.

12 Mitigation of Other Attacks

N/A for this module.