Encryptics® Cryptographic Library
FIPS 140-2 Non-Proprietary Security Policy

Document Revision 0.7

02/28/2014

Prepared for:

Encryptics

5566 West Main Street, Suite 207, Frisco, TX 75033

Prepared By:

www.gossamersec.com

ENCRYPTICS

Revision History

Revision	Date	Authors	Summary
0.1	8/22/2013	Morris	Initial draft
0.2	8/29/2013	Morris	Minor updates
0.3	9/18/2013	Morris	Updates
0.4	10/15/2013	Morris	Updates
0.5	11/01/2013	Morris	Updates
0.6	12/05/2013	Morris	Updates to correct typos
0.7	02/28/2014	Morris	Updated to respond to CMVP comments

ENCRYPTICS

Table of Contents

1. Introduction 4
2. Encryptics® Cryptographic Library 5
2.1 Module Specification 5
2.1.1 Security Level 6
2.1.2 FIPS Mode of Operation 6
2.1.3 Approved Cryptographic Algorithms 7
2.1.4 Non-Approved Cryptographic Algorithms 8
2.2 Module Interfaces 8
2.3 Roles, Services and Authentication 8
2.4 Finite State Model 12
2.5 Physical Security 13
2.6 Operational Environment 13
2.7 Key Management 13
2.8 Electromagnetic Interference and Compatibility 14
2.9 Self-Tests 14
2.10 Guidance and Secure Operation 14
2.10.1 Crypto-office Guidance 14
2.10.2 User Guidance 15
2.11 Mitigation of Other Attacks 15

ENCRYPTICS

1. Introduction

This non-proprietary FIPS 140-2 security policy for the Encryptics Cryptographic Library details the secure operation of the Encryptics Cryptographic Library as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology(NIST) of the United State Department of Commerce. This document, the Cryptographic Module Security Policy (CMSP), also referred to as the Security Policy, specifies the security rules under which the Encryptics Cryptographic Library must operate.

The Encryptics Cryptographic Library underpins Encryptics' technology offerings and provides them with the protection afforded by industry-standard, government-approved algorithms to ensure that only authorized users and authorized devices are allowed to access private information stored within Encryptics .SAFE packages. Both Encryptics for Email and Encryptics Data Protection API products leverage the Encryptics Cryptographic Library to ensure use of FIPS 140-2 validated cryptography.

2. Encryptics® Cryptographic Library

2.1 Module Specification

The Encryptics Cryptographic Library (SW Version 1.0.3.0) (hereinafter referred to as the "Library", "cryptographic module" or the "module") is a software only cryptographic module composed of the Encryptics Cryptographic Library assembly and the Microsoft Windows Enchanced Cryptographic Provider (a supporting operating system library upon which the assembly relies) which together execute on a general-purpose computer system running Microsoft Windows.

The physical perimeter of the general-purpose computer (GPC) comprises the module's physical cryptographic boundary, while the combination of the Encryptics Cryptographic Library assembly and the Microsoft Windows Enhanced Cryptographic Provider (RSAENH)[1] constitute the module's logical cryptographic boundary.

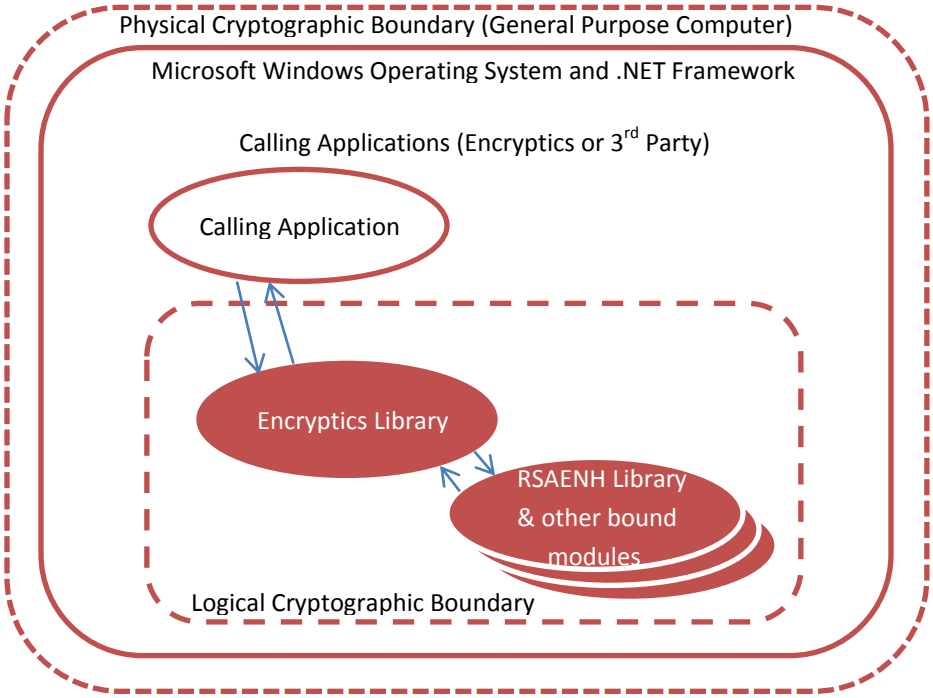

Figure 1 - Logical Diagram

[1] Please see section 2.6 for details regarding RSAENH's 140-2 validation certificates

2.1.1 Security Level

The Encryptics Cryptographic Library meets the overall requirements applicable to Level 1 security overall of FIPS 140-2 and the below specified section security levels.

Table 1 - Module Security Level Specification

#	FIPS 140-2 Section	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	3
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A
	Overall Level	1

2.1.2 FIPS Mode of Operation

The Encryptics Cryptographic Library utilizes FIPS-Approved and FIPS-Allowed services of the underlying 140-2 validated RSAENH module, and to ensure FIPS compliant operation, the operator must adhere to the Security Policy rules of the underlying RSAENH module (to which the module is bound). Those rules vary slightly depending upon which operating system RSAENH executes on and have been distilled into the following table.

Table 2 - Security Policy Rules

RSAENH (or module to which it is bound) Security Policy Rule	Applies if executing on: XP	Vista	Win7/ 2008
The operating system runs in “single user” mode where there is only one interactive user during a logon session.	X	X	X
Ensure the following registry valueis NOT set “HKLM\Software\Microsoft\Cryptography\ExpoOffload”	X	X	X
The operator must set the following registry DWORD to 1 “HKLM\System\CurrentControlSet\Control\Lsa\FIPSAlgorithmPolicy”	X		
Create the following registry keys and values “HKLM\Software\Microsoft\Cryptography\Defaults\Provider\Microsoft	X		

RSAENH (or module to which it is bound) Security Policy Rule	Applies if executing on: XP	Vista	Win7/ 2008
Enhanced RSA and AES Cryptographc Provider" with the following values "Image Path"="rsaenh.dll" "Type"=dword:00000018 "SigInFile"=dword:00000000			
The operator must install Microsoft Hotfix KB954059 to ensure the correct versions of RSAENH.dll and CI.dll are installed		X	
RSAENH is bound to, in turn, BOOTMGR, WINLOAD.EXE, CI.DLL.		X	
The operator must set the following registry DWORD to 1 "HKLM\System\CurrentControlSet\Control\Lsa\FIPSAlgorithmPolicy\Enabled"		X	X
Disable Debug mode and enable Driver Signing enforcement.		X	X
RSAENH is bound to BOOTMGR, WINLOAD.EXE, CI.DLL, and CNG.SYS.			X

Additionally, please note that while the underlying Microsoft RSAENH library provides a great number of algorithms, the Encryptics library utilizes only AES-256, RSA 2048-bit, HMAC-SHA-1, and SHA-256. However, the other algorithms of the underlying Microsoft RSA library are subject to the algorithm transition rule described SP 800-131A and FIPS 186-4.

Moreover, because the module relies upon the underlying Microsoft RSAENH library, the module generates cryptographic keys whose strengths are modified by available entropy.

2.1.3 Approved Cryptographic Algorithms

The module uses cryptographic algorithm implementations that have received the following certificate numbers from the Cryptographic Algorithm Validation Program.

Table 3 – FIPS-Approved Algorithm Certificates

Algorithm	CAVP Certificate when operating on Microsoft Windows XP	Vista	Windows 7	Server 2008
AES-256 CBC	781	739	1168	1168
RSA 2048, PCKS#1, Sign/Verify	371	353 & 354	557 & 559	568 & 559
HMAC-SHA-1	428	407	673	687
SHA-1/256	783	753	1081	1081
RNG	447	N/A	N/A	N/A
DRBG	N/A	V/A[2]	23	23

[2] Vendor-Affirmed

2.1.4 Non-Approved Cryptographic Algorithms

The module uses the following non-FIPS 140-2 approved, but allowed, algorithms.

- RSA encrypt/decrypt with a 2048-bit key (key wrapping; key establishment methodology provide 112-bits of encryption strength).

2.2 Module Interfaces

The module is classified as a multiple-chip standalone module for FIPS 140-2 purposes. As such, the module's physical cryptographic boundary encompasses the general-purpose computer running the Microsoft Windows operating system and interfacing with the computer peripherals (USB devices [keyboard and mouse], video devices [monitors, screens, camera], optical drives, audio devices [speakers, headset, and microphone], network devices [Ethernet and Wireless adapters], and power adapter).

However, the module provides only a logical interface via an Application Programming Interface (API) and does not interface or communication with or across any of the physical ports of the GPC. This logical interface exposes service that operators (calling applications) may use directly.

The API interface provided by the module is mapped onto the four FIPS 140-2 logical interfaces: data input, data output, control input, and status output. It is through this logical API that the module logically separates them into distinct and separate interfaces. The mapping of the module's API to the four FIPS 140-2 interfaces is as follows.

- Data input – input arguments to all constructors and methods specifying input parameters
- Data output – modified input arguments (those passed by reference) and return values for all constructors, methods, and properties modifying input arguments and returning values
- Control input – invocation of all methods and properties
- Status output – information returned by the IsValidState and VersionInfo methods and any exceptions thrown by constructors, methods, and properties

2.3 Roles, Services and Authentication

The module supports both of the FIPS 140-2 required roles, the Crypto-officer and the User role, and supports no additional roles. An operator implicitly selects the Crypto-officer role when loading (or causing loading of) the library and selected the User role when soliciting services from the module through its API. The module requires no operator authentication, and the below table enumerates the module's services.

Table 4 - Service Descriptions for Crypto-officer and User Roles

Service	Type[3]	Description
Crypto-Officer services		
Library Loading	N/A	The process of loading the assembly
User services		
Aes256CryptoServiceProviderFactory Class Services		
Aes256CryptoServiceProviderFactory()	C	Initializes a new instance of the Class
GetProvider()	M	Returns a default Aes256CryptoProvider instance
GetProvider(Byte[],Byte[])	M	Returns an Aes256CryptoProvider instance based upon the key and IV supplied
Aes256CryptoProvider Class Services		
Aes256CryptoProvider()	C	Default constructor returns AES-256 CBC
Aes256CryptoProvider(Byte[],Byte[])	C	Constructor accepts supplied keys and iv
Clear	M	Zeroizes any existing keys
Decrypt	M	Decrypts supplied data using AES key
Encrypt	M	Encrypts supplied data using AES key
GenerateNewKey	M	Generates a new AES key using values specified during instantiation.
Algorithm	P	Returns "AES"
BlockSize	P	Returns 128
CipherMode	P	Returns the current cipher feedback mode as an integer (only CBC currently supported).
CryptoAlgorithmId	P	Returns “AES”
CryptoMode	P	Returns the CipherMode as a string.
IV	P	Returns the current IV
Key	P	Returns the current AES key value
KeySize	P	Returns the current AES key size as an int (currently only 256-bit is supported)
PaddingMode	P	Returns the current padding mode.
SessionCryptoMode	P	Returns the CipherMode as a string.
Rsa2048CryptoServiceProviderFactory Class		
Rsa2048CryptoServiceProviderFactory	C	Initializes a new instance of the Class
GetProvider()	M	Returns a default Rsa2048CryptoProvider instance
GetProvider(String)	M	Returns an Rsa2048CryptoProvider instance based upon the XML Key String
GetProvider(RSACryptoServiceProvider)	M	Returns an Rsa2048CryptoProvider instance from the supplied instance
Rsa2048CryptoProvider Class		
Rsa2048CryptoProvider()	C	Default constructor generates 2048 RSA key pair.
Rsa2048CryptoProvider(String)	C	Constructor accepting supplied RSA 2048 key as XML String
Rsa2048CryptoProvider (RSACryptoServiceProvider)	C	Constructor accepting an RSACryptoServiceProvide instance.

[3] (C)onstructor, (M)ethod, or (P)roperty

Service	Type[3]	Description
Clear	M	Zeroizes the RSA key pair.
Decrypt	M	Decrypts using the RSA key pair
Encrypt	M	Encrypts using the RSA key pair
GenerateNewKeyPair	M	Generates a new key pair
Sign	M	Generates a signature of the supplied message using the RSA private key.
Verify	M	Verifies the supplied signature and message using the RSA public key
EncryptAlgorithmNum	P	Returns “NLKey.Types.NLKeyAlgorithm.RSA” as a 32 bit integer.
KeyPairXml	P	Returns the public and private key (if available).
KeySize	P	Returns the RSA KeySize.
PublicKey	P	Return the RSA Public Key as an XML String
ShaValue	P	Returns the signing alg string name (always "SHA256").
SecureEnvelopeFactory Class		
SecureEnvelopeFactory	C	Initializes a new instance of the Class
SecureEnvlope	P	Returns the default instance of an ISecureEnvelope
SecureEnvelope Class		
SecureEnvelope	C	Initializes a new instance of the Class
BuildSecureEnvelope	M	This method builds an NLSecureEnvelope which encapsulates a signed hash of the data passed via the first argument. The data is signed using the asymmetric signing provider, the signing server, the communicaitn key and content key all of which are passed in.
BuildSelfSignable	M	This method builds an NLSelfSignable object.
GetSecureEnvelopeContents	M	This method retrieves the contents of a SecureEnvelope
SelfCheck Class		
SelfCheck	C	Initializes a new instance of the Class
Reset	M	This method resets the state of the power up self-tests.
VerifyModule	M	This is the Module Initializer method. It is automatically called when the assembly is loaded into memory.
IsValidState	P	This property returns the current power up self check state of the module.
VersionInfo	P	This property returns the version of the assembly.
Sha256CryptoServiceProviderFactory Class		
Sha256CryptoServiceProviderFactory	C	Initializes a new instance of the Class
Hasher	P	Returns an instance of the Sha256Hasher class
Sha256CryptoServiceProvider Class		
Sha256CryptoServiceProvider	C	Initializes a new instance of the Class
Digest	M	Returns a hash of the data passed in.

Table 5 - Service Inputs and Outputs

Service	Data Input	Data Output	CSP	Access[4]	Status Out
Crypto-Officer services					
Library Loading	N/A	N/A	N/A	N/A	Pass/Fail
User services					
Aes256CryptoServiceProvider Factory Class Services					
Aes256CryptoServiceProvider Factory()	N/A	Instance Ref	N/A	N/A	Exception
GetProvider()	N/A	Instance Ref	N/A	N/A	Exception
GetProvider(Byte[],Byte[])	AES Key and IV	Instance Ref	AES Key	W	Exception
Aes256CryptoProvider Class Services					
Aes256CryptoProvider()	N/A	Instance Ref	N/A	N/A	Exception
Aes256CryptoProvider(Byte[], Byte[])	AES Key and IV	Instance Ref	AES Key	W	Exception
Clear	N/A	N/A	AES Key	Z	Exception
Decrypt	Ciphertext	Plaintext	AES Key	X	Exception
Encrypt	Plaintext	Ciphertext	AES Key	X	Exception
GenerateNewKey	N/A	N/A	AES Key	G	Exception
Algorithm	N/A	AES Alg ID	N/A	N/A	Exception
BlockSize	N/A	AES Block size	N/A	N/A	Exception
CipherMode	N/A	CBC	N/A	N/A	Exception
CryptoAlgorithmId	N/A	AES Alg ID	N/A	N/A	Exception
CryptoMode	N/A	CBC	N/A	N/A	Exception
IV	N/A	IV Value	N/A	N/A	Exception
Key	N/A	Key Value	AES Key	R	Exception
KeySize	N/A	256-bit only	N/A	N/A	Exception
PaddingMode	N/A	Padding mode	N/A	N/A	Exception
SessionCryptoMode	N/A	CBC	N/A	N/A	Exception
Rsa2048CryptoServiceProvideFactory Class					
Rsa2048CryptoServiceProvideFactory	N/A	Instance Ref	N/A	N/A	Exception
GetProvider()	N/A	Instance Ref	RSA Keys	G	Exception
GetProvider(String)	RSA Keys	Instance Ref	RSA Keys	W	Exception
GetProvider(RSACryptoServiceProvider)	RSA Keys	Instance Ref	RSA Keys	W	Exception
Rsa2048CryptoProvider Class					
Rsa2048CryptoProvider()	N/A	Instance Ref	RSA Keys	G	Exception
Rsa2048CryptoProvider(String)	RSA Keys	Instance Ref	RSA Keys	W	Exception
Rsa2048CryptoProvider (RSACryptoServiceProvider)	RSA Keys	Instance Ref	RSA Keys	W	Exception

[4] (G)enerate, (R)ead, (W)rite, e(X)excute, (Z)eroize

Service	Data Input	Data Output	CSP	Access[4]	Status Out
Clear	N/A	N/A	RSA Keys	Z	Exception
Decrypt	Ciphertext	Plaintext	RSA Keys	X	Exception
Encrypt	Plaintext	Ciphertext	RSA Keys	X	Exception
GenerateNewKeyPair	N/A	N/A	RSA Keys	G	Exception
Sign	Message	N/A	RSA Keys	X	Exception
Verify	Signature	N/A	RSA Keys	X	Exception
EncryptAlgorithmNum	N/A	Alg ID	N/A	N/A	Exception
KeyPairXml	N/A	Key Value	RSA Keys	R	Exception
KeySize	N/A	Key size	N/A	N/A	Exception
PublicKey	N/A	Public Key	N/A	N/A	Exception
ShaValue	N/A	SHA size	N/A	N/A	Exception
SecureEnvelopeFactory Class					
SecureEnvelopeFactory	N/A	Instance Ref	N/A	N/A	Exception
SecureEnvlope	N/A	Instance Ref	N/A	N/A	Exception
SecureEnvelope Class					
SecureEnvelope	N/A	Instance Ref	N/A	N/A	Exception
BuildSecureEnvelope	RSA Keys and data	Secure Envelope	RSA Keys	X	Exception
BuildSelfSignable	RSA keys and data	Signed data	RSA Keys	X	Exception
GetSecureEnvelopeContents	RSA Keys and Secure Envlope	Decrypted contents	RSA & AES Keys	X	Exception
SelfCheck Class					
SelfCheck	N/A	Instance Ref	N/A	N/A	Exception
Reset	N/A	N/A	N/A	N/A	Exception
VerifyModule	N/A	N/A	HMAC Integrity Key	X	Exception
IsValidState	N/A	N/A	N/A	N/A	Current State
VersionInfo	N/A	N/A	N/A	N/A	Module Version
Sha256CryptoServiceProvider Factory Class					
Sha256CryptoServiceProvider Factory	N/A	Instance Ref	N/A	N/A	Exception
Hasher	N/A	Instance Ref	N/A	N/A	Exception
Sha256CryptoServiceProvider Class Sha256CryptoServiceProvider	N/A	Instance Ref	N/.A	N/A	Exception
Digest	Message	Hash	N/A	N/A	Exception

2.4 Finite State Model

The module has a Finite State Model (FSM) that descrbes the module's behavior and transitions based upon its current state and the command received. The module's FSM was reviewed as part of the overall FIPS 140-2 validation.

2.5 Physical Security

The physical security requirements does not apply to the module. The module is a software-only module that executes upon a general-purpose computer

2.6 Operational Environment

The module executes on a general purpose operating system running in single user mode that segregates processes into separate process spaces. Thus, the operating system separates each process space from all others. The below table listed the specific Microsoft Windows operating systems and their associated FIPS 140-2 certificate number for the Microsoft Enhanced Cryptographic Provider (RSAENH) upon which the Encryptics Cryptograhic Library relies.

Table 6 - Validated Operational Environments

#	Operating System and Test Platform	140-2 Cert.#
1	Microsoft Windows XP SP3 with .NET Framework 3.5 running on a Dell SC430 (single-user mode)	989
2	Microsoft Windows Vista SP1 (x64 version) with .NET Framework 3.5 running on a Dell SC430 (single-user mode)	1002
3	Microsoft Windows 7 SP1 (x64 version) with .NET Framework 3.5 running on an HP Compaq dc7600 (single-user mode)	1330
4	Microsoft Windows Server 2008 R2 SP1 (x64 version) with .NET Framework 4.0 running on an HP Compaq dc7600 (single-user mode)	1337

2.7 Key Management

The module possesses only one key, its HMAC-SHA-1 self-integrity test key. Beyond that key, the module does not store any other keys persistently, and it is the calling applications responsibility to appropriately manage keys. The module can generate keys (both AES-256 symmetric keys and RSA 2048-bit asymmetric key pairs), can accept key entered by an operator, and affords an operator the ability to zeroize keys held in RAM. The module also provides key establishment through its SecureEnvelope Class by employing RSA key wrapping to wrap AES keys, and the module's key establishment methodology provide 112-bits of encryption strength. The following table describes the module's Security Relevant Data Items (SRDI's) including asymmetric and symmetric keys.

Table 7 - Module Keys

Key	Type	Size	Description	Origin	Stored	Zeroized

AES Keys	AES	256 bits	Symmetric keys intended to encrypt and decrypt User data	Entered, Established, or Generated	RAM / plaintext	Clear method
RSA Keys	RSA	2048 bits	Asymmetric keys used either for PKCS#1 sign and verify operations or for RSA key wrapping	Entered or Generated	RAM /plaintext	Clear method
Self-check Key	HMAC	86 bytes	HMAC-SHA-1 key used by the module for it's power up integrity test	Compiled into the module	Module image / plaintext & obfuscated	N/A (see 140-2 IG 7.4)

2.8 Electromagnetic Interference and Compatibility

The module meets level 3 security for FIPS 140-2 EMI/EMC requirements as the Encryptics Cryptographic Library passed validation executing upon general-purpose computers that confirm to the EMI/EMC requirements specific by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B (i.e., for home use).

2.9 Self-Tests

The module automatically performs a complete set of power-up self-tests during library load to ensure proper operation, thus an operator has no access to cryptographic functionality unless the power-up self-tests pass and the library load succeeds. The power-up self-tests include an integrity check of the module's software using an HMAC-SHA-1 value calculated over the module's file image. Should the module fail a self-test, the module will throw an exception and unload itself from memory. Additionally, the underlying RSAENH module performs both power-up and conditional self-tests for its cryptographic algorithms. Finally, an operator may invoke the power-up self-tests at any time by power-cycling the GPC and then reloading module.

2.10 Guidance and Secure Operation

The Encryptics Cryptographic Library meets overall Level 1 requirements for FIPS PUB 140-2. The sections below describe the Crypto-offier and User guideance.

2.10.1 Crypto-office Guidance

The Crypto-officer or operator responsible for configuring the operational environment upon which the module runs must ensure to FIPS compliant operation (as described in section 2.1.2, FIPS Mode of Operation, of the Security Policy).

Additionally, the Crypto-officer is defined to be the operator responsible for loading the library, thus when invoked by a calling application (either at library load or dynamically), the operating system loader will load the module, causing it to automatically perform its power-up self-tests. Should the module fail its power-up self-tests, the module will output a LoadException to the Crypto-officer.

2.10.2 User Guidance

Once the operating system has been properly configured by the Crypto-officer (if needed), the Encryptics Cryptographic Library requires no special usage to operate in a FIPS-compliant manner. The module utilizes only FIPS-Approved or FIPS-Allowed cryptographic algorithms. The User must assume responsibility for managing all keys, as the module does not provide any persistent key storeage.

2.11 Mitigation of Other Attacks

The Encryptics Cryptographic Library does not claim to mitigate any attacks beyond the FIPS 140-2 Level 1 requirements for validation.