(*CryptoServicesRegistrar.isInApprovedOnlyMode()*). If the module is configured to allow approved and non-approved mode operation, a call to *CryptoServicesRegistrar.setApprovedMode(true)* will switch the current thread of user control into approved mode.

In FIPS-approved mode, the module will not provide non-approved algorithms, therefore, exceptions will be called if the user tries to access non-approved algorithms in the Approved Mode.

2.1.3 Module Configuration

In default operation, the module will start with both approved and non-approved mode enabled.

If the module detects that the system property *com.safelogic.cryptocomply.fips.approved_only* is set to *true* the module will start in approved mode and non-approved mode functionality will not be available.

If the underlying JVM is running with a Java Security Manager installed, the module will be running in approved mode with secret and private key export disabled.

Use of the module with a Java Security Manager requires the setting of some basic permissions to allow the module HMAC-SHA-256 software integrity test to take place as well as to allow the module itself to examine secret and private keys. The basic permissions required for the module to operate correctly with a Java Security Manager are indicated by a Y in the **Req** column of Table 2 – Available Java Permissions.

Permission	Settings	Req	Usage
RuntimePermission	“getProtectionDomain”	Y	Allows checksum to be carried out on jar
RuntimePermission	“accessDeclaredMembers”	Y	Allows use of reflection API within the provider
PropertyPermission	“java.runtime.name”, “read”	N	Only if configuration properties are used
SecurityPermission	"putProviderProperty.BCFIPS"	N	Only if provider installed during execution
CryptoServicesPermission	“unapprovedModeEnabled”	N	Only if unapproved mode algorithms required

Permission	Settings	Req	Usage
CryptoServicesPermission	"changeToApprovedModeEnabled"	N	Only if threads allowed to change modes
CryptoServicesPermission	"exportSecretKey"	N	To allow export of secret keys only
CryptoServicesPermission	"exportPrivateKey"	N	To allow export private keys only
CryptoServicesPermission	"exportKeys"	Y	Required to be applied for the module itself. Optional for any other codebase.
CryptoServicesPermission	"tlsNullDigestEnabled"	N	Only required for TLS digest calculations
CryptoServicesPermission	"tlsPKCS15KeyWrapEnabled"	N	Only required if TLS is used with RSA encryption
CryptoServicesPermission	"tlsAlgorithmsEnabled"	N	Enables both NullDigest and PKCS15KeyWrap
CryptoServicesPermission	"defaultRandomConfig"	N	Allows setting of default SecureRandom
CryptoServicesPermission	"threadLocalConfig"	N	Required to set a thread local property in the CryptoServicesRegistrar
CryptoServicesPermission	"globalConfig"	N	Required to set a global property in the CryptoServicesRegistrar

Table 2 – Available Java Permissions

2.1.4 Approved Cryptographic Algorithms

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

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
4702	**AES**	FIPS 197 SP 800-38A	ECB, CBC, OFB, CFB8, CFB128, CTR	128, 192, 256	Encryption, Decryption
Based on 4702	**AES-CBC Ciphertext Stealing (CS)**	Addendum to SP 800-38A, Oct 2010	CBC-CS1, CBC-CS2, CBC-CS3	128, 192, 256	Encryption, Decryption
4702	**CCM**	SP 800-38C	AES	128, 192, 256	Authenticated Encryption, Authenticated Decryption
4702 (AES) 2494 (Triple-DES)	**CMAC**	SP 800-38B	AES, Triple-DES	AES with 128, 192, 256 Triple-DES with 2-key[1], 3-key	MAC Generation, MAC Verification
4702	**GCM/GMAC[2]**	SP 800-38D	AES	128, 192, 256	Authenticated Encryption, Authenticated Decryption, MAC Generation, MAC Verification

[1] 2-key Triple-DES is for legacy operations only (decryption, CMAC verification)
[2] GCM with an IV generated internally to the module, see section 3.1.3 of this document concerning external IVs. IV generation is compliant with IG A.5.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
1600	**DRBG**	SP 800-90A	Hash DRBG, HMAC DRBG, CTR DRBG	112, 128, 192, 256	Random Bit Generation
1244	**DSA**[3]	FIPS 186-4	PQG Generation, PQG Verification, Key Pair Generation, Signature Generation, Signature Verification	1024, 2048, 3072 bits (1024 only for SigVer)	Digital Signature Services
1160 1343 (CVL)	**ECDSA**	FIPS 186-4	Signature Generation Component, Key Pair Generation, Signature Generation, Signature Verification, Public Key Validation	P-192*, P-224, P-256, P-384, P- 521, K-163*, K-233, K-283, K-409, K-571, B-163*, B- 233, B-283, B-409, B-571 * Curves only used for Signature Verification and Public Key Validation	Digital Signature Services

[3] DSA signature generation with SHA-1 is only for use with protocols.

2.2 Critical Security Parameters and Public Keys

2.2.1 Critical Security Parameters

The table below provides a complete list of Critical Security Parameters used within the module:

CSP	Description / Usage
AES Encryption Key	[FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) encrypt key[19]
AES Decryption Key	[FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) decrypt key
AES Authentication Key	[FIPS-197] AES (128/192/256) CMAC/GMAC key
AES Wrapping Key	[SP 800-38F] AES (128/192/256) key wrapping key
DH Agreement key	[SP 800-56A-rev2] Diffie-Hellman (>= 2048) private key agreement key
DRBG(CTR AES)	V (128 bits) and AES key (128/192/256), entropy input (length dependent on security strength)
DRBG(CTR Triple-DES)	V (64 bits) and Triple-DES key (192), entropy input (length dependent on security strength)
DRBG(Hash)	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
DRBG(HMAC)	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
DSA Signing Key	[FIPS 186-4] DSA (2048/3072) signature generation key
EC Agreement Key	[SP 800-56A-rev2] EC (All NIST defined B, K, and P curves >= 224 bits) private key agreement key
EC Signing Key	[FIPS 186-4] ECDSA (All NIST defined B, K, and P curves >= 224 bits) signature generation key
HMAC Authentication Key	[FIPS 198-1] Keyed-Hash key (SHA-1, SHA-2). Key size determined by security strength required (>= 112 bits)
IKEv2 Derivation Function Secret Value	[SP 800-135] Secret value used in construction of key for the specified IKEv2 PRF

[19] The AES-GCM key and IV are generated randomly per IG A.5, and the Initialization Vector (IV) is a minimum of 96 bits. In the event module power is lost and restored, the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

2.3 Module Interfaces

The figure below shows the module's physical and logical block diagram:

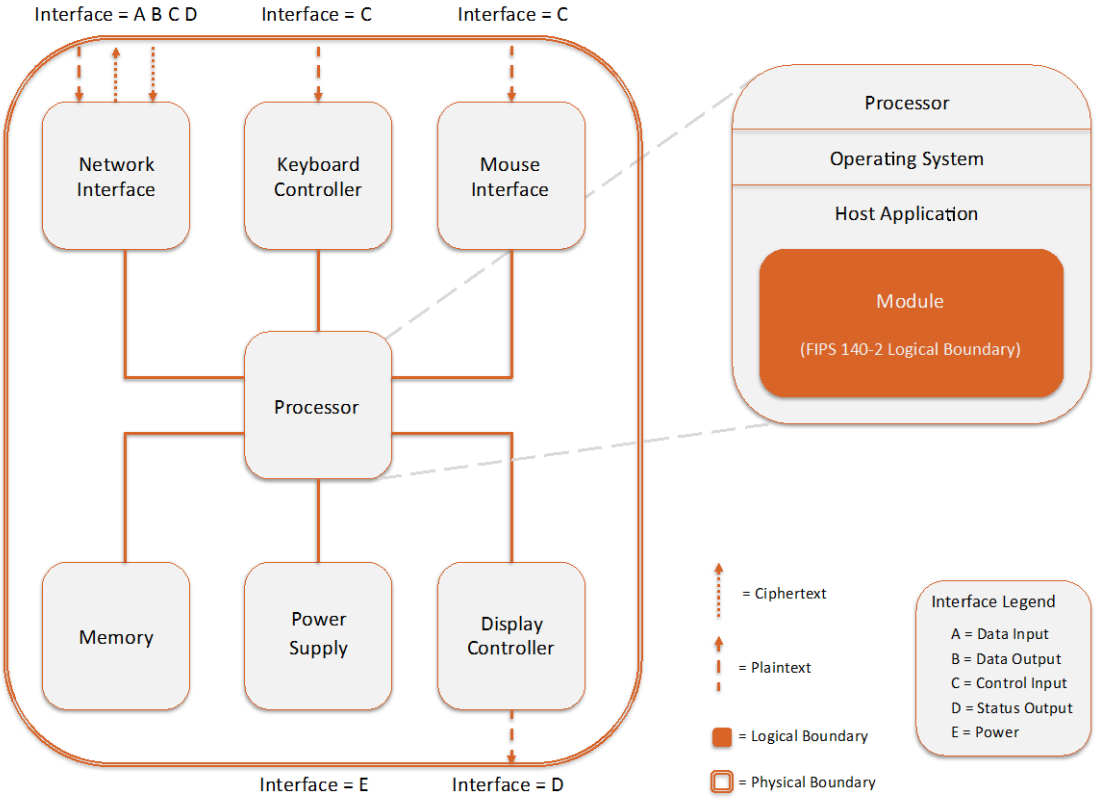

Figure 1 – Module Boundary and Interfaces Diagram

The interfaces (ports) for the physical boundary include the computer keyboard port, mouse port, network port, USB ports, display and power plug. When operational, the module does not transmit any information across these physical ports because it is a software cryptographic module. Therefore, the module's interfaces are purely logical and are provided through the Application Programming Interface (API) that a calling daemon can operate. The logical interfaces expose services that applications directly call, and the API provides functions that may be called by a referencing application (see Section 2.4 – Roles, Services, and Authentication for the list of available functions). The module distinguishes between logical interfaces by logically separating the information according to the defined API.

The API provided by the module is mapped onto the FIPS 140- 2 logical interfaces: data input, data output, control input, and status output. Each of the FIPS 140- 2 logical interfaces relates to the module's callable interface, as follows:

FIPS 140-2 Interface	Logical Interface	Module Physical Interface
Data Input	API input parameters – plaintext and/or ciphertext data	Network Interface
Data Output	API output parameters and return values – plaintext and/or ciphertext data	Network Interface
Control Input	API method calls- method calls, or input parameters, that specify commands and/or control data used to control the operation of the module	Keyboard Interface, Mouse Interface
Status Output	API output parameters and return/error codes that provide status information used to indicate the state of the module	Display Controller, Network Interface
Power	None	Power Supply

Table 9 – Logical Interface / Physical Interface Mapping

As shown in Figure 1 – Module Boundary and Interfaces Diagram and Table 11 – Module Services, Roles, and Descriptions, the output data path is provided by the data interfaces and is logically disconnected from processes performing key generation or zeroization. No key information will be output through the data output interface when the module zeroizes keys.

2.4 Roles, Services, and Authentication

2.4.1 Assumption of Roles

The module supports two distinct operator roles, User and Crypto Officer (CO). The cryptographic module implicitly maps the two roles to the services. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

The module does not support a Maintenance role or bypass capability. The module does not support authentication.

Role	Role Description	Authentication Type
CO	Crypto Officer – Powers on and off the module	N/A – Authentication is not a requirement for Level 1
User	User – The user of the complete API	N/A – Authentication is not a requirement for Level 1

Table 10 – Description of Roles

2.4.2 Services

All services implemented by the module are listed in Table 11 – Module Services, Roles, and

Note: The module services are the same in the approved and non-approved modes of operation. The only difference is the function(s) used (approved/allowed or non-approved/non-allowed).

Services in the module are accessed via the public APIs of the Jar file. The ability of a thread to invoke non-approved services depends on whether it has been registered with the module as approved mode only. In approved only mode no non-approved services are accessible. In the presence of a Java SecurityManager approved mode services specific to a context, such as DSA and ECDSA for use in TLS, require specific permissions to be configured in the JVM configuration by the Cryptographic Officer or User.

In the absence of a Java SecurityManager specific services related to protocols such as TLS are available, however must only be used in relation to those protocols.

Table 12 – CSP Access Rights within Services defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

G = Generate: The module generates the CSP.

R = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP.

E = Execute: The module executes using the CSP.

W = Write: The module writes the CSP. The write access is typically performed after a CSP is imported into the module, when the module generates a CSP, or when the module overwrites an existing CSP.

Z = Zeroize: The module zeroizes the CSP.

Service	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Keys	HMAC Keys	KDF Secret Values	RSA Keys	Triple-DES Keys
Initialize Module and Run Self-Tests on Demand										
Show Status										

Service	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Keys	HMAC Keys	KDF Secret Values	RSA Keys	Triple-DES Keys
Zeroize / Power-off	Z	Z	Z	Z	Z	Z	Z		Z	Z
Data Encryption	R									R
Data Decryption	R									R
MAC Calculation	R						R			R
Signature Authentication				R		R			R	
Signature Verification				R		R			R	
DRBG (SP800-90A) output	G	G	G,R	G	G	G	G		G	G
Message Hashing										
Keyed Message Hashing							R			
TLS Key Derivation Function								R		
SP 800-108 KDF								R		
SSH Derivation Function								R		
X9.63 Derivation Function								R		

Service	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Keys	HMAC Keys	KDF Secret Values	RSA Keys	Triple-DES Keys
SP 800-56A-rev2 Concatenation Derivation Function								R		
IKEv2 Derivation Function								R		
SRTP Derivation Function								R		
PBKDF							G,R			
Key Agreement Schemes	G	R			R		R		R	G
Key Wrapping/Transport (RSA, AES, Triple-DES)	R						R		R	R
Key Unwrapping (RSA, AES, Triple-DES)	R						R		R	R
NDRNG Callback			G							
Utility										

Table 12 – CSP Access Rights within Services

2.5 Physical Security

The module is a software-only module and does not have physical security mechanisms.

3 Security Rules and Guidance

3.1 Basic Enforcement

The module design corresponds to the Module security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-2 Level 1 module.

1. The module provides two distinct operator roles: User and Cryptographic Officer.
2. The module does not provide authentication.
3. The operator may command the module to perform the power up self-tests by cycling power or resetting the module.
4. Power-up self-tests do not require any operator action.
5. Data output is inhibited during key generation, self-tests, zeroization, and error states.
6. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
7. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
8. The module does not support concurrent operators.
9. The module does not have any external input/output devices used for entry/output of data.
10. The module does not enter or output plaintext CSPs from the module's physical boundary.
11. The module does not output intermediate key values.

3.1.1 Additional Enforcement with a Java SecurityManager

In the presence of a Java SecurityManager approved mode services specific to a context, such as DSA and ECDSA for use in TLS, require specific policy permissions to be configured in the JVM configuration by the Cryptographic Officer or User. The SecurityManager can also be used to restrict the ability of particular code bases to examine CSPs. See Section 2.1.3 Module Configuration for further advice on this.

In the absence of a Java SecurityManager specific services related to protocols such as TLS are available, however must only be used in relation to those protocols.

3.1.2 Basic Guidance

The jar file representing the module needs to be installed in a JVM's class path in a manner appropriate to its use in applications running on the JVM.

Functionality in the module is provided in two ways. At the lowest level there are distinct classes that provide access to the FIPS approved and non-FIPS approved services provided by the module. A more abstract level of access can also be gained through the use of strings providing operation names passed into the module's Java cryptography provider through the

APIs described in the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE).

When the module is being used in FIPS approved-only mode, classes providing implementations of algorithms which are not FIPS approved, or allowed, are explicitly disabled.

3.1.3 Enforcement and Guidance for GCM IVs

IVs for GCM can be generated randomly, where an IV is not generated randomly the module supports the importing of GCM IVs.

In approved mode, when a GCM IV is generated randomly, the module enforces the use of an approved DRGB in line with Section 8.2.2 of SP 800-38D.

In approved mode, importing a GCM IV is non-conformant unless the source of the IV is also FIPS approved for GCM IV generation.

Per IG A.5, Section 2.2.1 of this Security Policy also states that in the event module power is lost and restored the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

3.1.4 Enforcement and Guidance for use of the Approved PBKDF

In line with the requirements for SP 800-132, keys generated using the approved PBKDF must only be used for storage applications. Any other use of the approved PBKDF is non-conformant.

In approved mode the module enforces that any password used must encode to at least 14 bytes (112 bits) and that the salt is at least 16 bytes (128 bits) long. The iteration count associated with the PBKDF should be as large as practical.

As the module is a general purpose software module, it is not possible to anticipate all the levels of use for the PBKDF, however a user of the module should also note that a password should at least contain enough entropy to be unguessable and also contain enough entropy to reflect the security strength required for the key being generated. In the event a password encoding is simply based on ASCII a 14 byte password is unlikely to contain sufficient entropy for most purposes. Users are referred to Appendix A, “Security Considerations” of SP 800-132 for further information on password, salt, and iteration count selection.

3.1.5 Software Installation

The module is provided directly to solution developers and is not available for direct download to the general public. The module and its host application are to be installed on an operating system specified in Section 2.6 or one where portability is maintained.

4 References and Acronyms

4.1 References

Abbreviation	Full Specification Name
ANSI X9.31	*X9.31-1998, Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), September 9, 1998*
FIPS 140-2	*Security Requirements for Cryptographic modules, May 25, 2001*
FIPS 180-4	*Secure Hash Standard (SHS)*
FIPS 186-3	*Digital Signature Standard (DSS)*
FIPS 186-4	*Digital Signature Standard (DSS)*
FIPS 197	*Advanced Encryption Standard*
FIPS 198-1	*The Keyed-Hash Message Authentication Code (HMAC)*
FIPS 202	*SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*
IG	*Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*
PKCS#1 v2.1	*RSA Cryptography Standard*
PKCS#5	*Password-Based Cryptography Standard*
PKCS#12	*Personal Information Exchange Syntax Standard*
SP 800-20	*Modes of Operation Validation System for Triple Data Encryption Algorithm (TMOVS)*
SP 800-38A	*Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*
SP 800-38B	*Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
SP 800-38C	*Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*
SP 800-38D	*Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
SP 800-38F	*Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*
SP 800-56A	*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*
SP 800-56B	*Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*
SP 800-56C	*Recommendation for Key Derivation through Extraction-then- Expansion*
SP 800-67	*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*
SP 800-89	*Recommendation for Obtaining Assurances for Digital Signature Applications*

SP 800-90A	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
SP 800-108	*Recommendation for Key Derivation Using Pseudorandom Functions*
SP 800-132	*Recommendation for Password-Based Key Derivation*
SP 800-135	*Recommendation for Existing Application–Specific Key Derivation Functions*

Table 15 – References

4.2 Acronyms

The following table defines acronyms found in this document:

Acronym	Term
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher-Block Chaining
CCM	Counter with CBC-MAC
CDH	Computational Diffie-Hellman
CFB	Cipher Feedback Mode
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CPU	Central Processing Unit
CS	Ciphertext Stealing
CSP	Critical Security Parameter
CTR	Counter-mode
CVL	Component Validation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DRAM	Dynamic Random Access Memory
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DSTU4145	Ukrainian DSTU-4145-2002 Elliptic Curve Scheme
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code

Acronym	Term
GOST	Gosudarstvennyi Standard Soyuza SSR/Government Standard of the Union of Soviet Socialist Republics
GPC	General Purpose Computer
HMAC	(Keyed-) Hash Message Authentication Code
IG	Implementation Guidance
IV	Initialization Vector
JAR	Java ARchive
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
MD5	Message Digest algorithm MD5
N/A	Non Applicable
NDRNG	Non Deterministic Random Number Generator
OCB	Offset Codebook Mode
OFB	Output Feedback
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PKCS	Public-Key Cryptography Standards
PQG	Diffie-Hellman Parameters P, Q and G
RC	Rivest Cipher, Ron's Code
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
TCBC	TDEA Cipher-Block Chaining
TCFB	TDEA Cipher Feedback Mode
TDEA	Triple Data Encryption Algorithm
TDES	Triple Data Encryption Standard
TECB	TDEA Electronic Codebook
TOFB	TDEA Output Feedback
TLS	Transport Layer Security
USB	Universal Serial Bus
XOF	Extendable-Output Function

Table 16 – Acronyms and Terms