FIPS140-2 Security Policy for CryptoLite

FIPS140-2 Security Policy for CryptoLite

October 2003

Revision: 1.8

NON CONFIDENTIAL

Status: Released

First Edition (January 2002)
This edition applies to the First Edition of the IBM BlueZ – FIPS140-2 Security Policy for CryptoLite and to all subsequent versions until otherwise indicated in new editions. IBM welcomes your comments on this publication. Please address them to: bluez@zurich.ibm.com. When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

1. Document Information

1.1. Document Scope

This document describes the services that the IBM CryptoLite in Java module provides to a population of security officers, and the security policy governing access to those services. Included is a description of the basic security requirements for the CryptoLite library and a qualitative description of how each of the security requirements is achieved.

1.2. Table of Contents

2. Applicable documents

Cryptography

- RSA Laboratories PKCS #15 v1.0: Cryptographic Token Information Format Standard – April 23, 1999
- RSA Laboratories PKCS#15 v1.0 Amendment 1 Draft #1 - October 20, 1999
- FIPS 140-2 standard, the *Derived Test Requirements*, and on-line implementation guidelines
- Digital Encryption Standard: FIPS PUB 46-3, FIPS PUB 74, and FIPS PUB 81
- SHA-1: FIPS PUB 180-1
- Digital Signature Standard : FIPS PUB 186-2 27 January 2000
- Pseudo-random Number Generation: Appendix 3 of FIPS PUB 186.
- *Digital Signature Scheme Giving Message Recovery*: ISO/IEC 9796
- The 3DES standard, ANSI X9.52, *Triple Data Encryption Algorithm Modes Of Operation*
- Advanced Encryption Standard (AES) FIPS Publication 197, November 26, 2001
- Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry. ANSI X9.31-1998

3. CryptoLite Library

3.1. Module Components

The following table lists the module components:

Type	Name	Release
Software	CryptoLite JAR file – cl140.jar	Version 3.0 (FIPS140/Prod)
Documentation	CryptoLite Java Docs	Version 3.0 (FIPS140/Prod)

Table 1: Module Component List for all platforms

3.2. Module Description

The IBM CryptoLite Module provides a library of cryptographic services written 100% in Java. For the purpose of FIPS 140-2 certification, the implementation is made available in the form of a signed Jar file. The CryptoLite module includes an interface for a high performance cryptographic booster module that written in native code. This booster module is outside the scope of the CryptoLite validation and dealt with in a separate validation.

4. Security Levels

The IBM CryptoLite module meets the overall requirements applicable to Level 1 security of FIPS 140-2. The individual security requirements specified for FIPS 140-2 meets the level specifications indicated in the following table.

Security Requirements Section	Level
Cryptographic Module	1
Ports and Interfaces	1
Roles and Services	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

Table 2: FIPS 140-2 validation levels

5. Cryptographic Module Specification

IBM CryptoLite is classified as a multi-chip standalone module for FIPS 140-2 purposes. As such, the CryptoLite Module must be validated upon a particular operating system and computer platform. The CryptoLite Module is packaged in a single Java Archive File, which contains all the classes for the module. IBM CryptoLite runs upon many other platforms including Windows '95, '98, and NT, Sun/Solaris, HP-UX, Linux, and AIX.

As outlined in G.5 of the Implementation Guidance for FIPS 140-2, the module maintains its compliance on other operating systems, provided:

- the GPC uses the specified single user operating system/mode specified on the validation certificate, or another compatible single user operating system, andion
- the source code of the software cryptographic module does not require modification prior to recompilation to allow porting to another compatible single user operating system.

Since the IBM CryptoLite module is a pure Java implementation it should be able to run unmodified on any system which supports a Java Runtime of at least Version 1.1. The above requirements will be demonstrated by testing and validating the IBM CryptoLite package on the following platforms.

Hardware	Operating System	Java VM version
IBM PC Compatible	Windows 2000, SP4	1.3.1_03
AIX Model 44P 170	AIX 5.2	1.3.1
Sun Blade 100	Solaris 5.8	1.2.2

Table 3: Platforms on which CryptoLite has been tested

The module provides no physical security features aside from the enclosure of the PC which the module runs on. Additionally, the computer that CryptoLite was tested on met the applicable FCC requirements. Finally, the module does not mitigate against any special attacks.

5.1. Cryptographic Interface

The CryptoLite package has a proprietary interface for interfacing to an external cryptographic acceleration module (CryptoLite in C). This module, also provided by IBM and FIPS140-2 validated, uses optimized native code to provide high performance speedup for CryptoLite functionality in a totally transparent manner. The module simply has to be present in the same directory as the CryptoLite module at module startup time. The validity and integrity of the booster module is checked using an approved HMAC method.

5.2. Cryptographic Standards

The IBM CryptoLite module supports the following approved and non-approved FIPS algorithms.

HASH Functions

Algorithm	Specification	FIPS Approved
MD2	**IETF RFC1319** Hash algorithm; hash size: 16 bytes; block size: 16 bytes. Used only for backward compatibility.	No
MD5	**IETF RFC 1321** Hash algorithm; hash size: 16 bytes; block size: 64 bytes. Used only for backward compatibility.	No
SHA-1	**FIPS180-1** Hash algorithm; hash size: 20 bytes; block size: 64 bytes.	Yes

SHA256	Hash algorithm; hash/block sizes: 32/64, bytes.	
SHA384	Hash algorithm; hash/block sizes: 48/128 bytes.	
SHA512	Hash algorithm; hash/block size 64/128 bytes.	

Table 4: CryptoLite Hash Functions

CIPHER Functions

Algorithm	Specification & Description	FIPS Approved
RC2	**IETF: RFC2268** Symmetric block cipher. Block size 8 bytes. Key size 0-1024 bits. RC2 allows adjustment of the effective key strength independent of the input key length.	No
RC4	Stream cipher; key sizes: 0-2048 bits.	No
RC6	Symmetric block cipher; block size: 16 bytes; key sizes: 0-? bits. RSA DSI Inc. candidate for AES	No
DES, DES-CBC	**FIPS 46-3** Symmetric block cipher; block size: 8 bytes; key size: 56 bits.	Yes
3DES, 3DES-CBC	**FIPS 46-3** Triple DES has 112/168 bits key length depending on type of key.	Yes
MDC1, MDC-2, MDC-4	Owned by IBM., Patent: US4908861 Modification detection codes (MDC) based on DES cipher. There are different schemes called MDC-1, MDC-2, and MDC-4. Mainly used for backward compatibility with main frame systems. Modern hash algorithms are much faster.	No
UNIX_CRYPT	Unix password encryption based on a modified DES.	No
AES **AES CBC** **AES 256**	FIPS197, Symmetric block cipher; block sizes: 16,24,32 bytes; key sizes: 16,24,32 bytes.	Yes
BLOWFISH	Blowfish encryption/decryption and macing; blocksize: 8 bytes.	No
HMAC SHA-1	Hashed Message Authentication Codes (HMAC) based on the SHA-1 hash algorithm.	Yes

Table 5: CryptoLite Cipher Functions

Public Key

Algorithm	Specification	FIPS Approved
RSA Sign/Verify	Public key encryption/signature scheme. Typical key/data sizes: 512, 768, 1024 (typical), 2048 bits.	Yes
RSA Encrypt/Decypt	RSA specification and padding scheme: PKCS#1 OAEP Padding scheme for RSA encryption: RFC2437	No
DSA Sign/Verify	Public key signature scheme. Cannot be used for encryption. Key sizes: 512-1024 bits in steps of 64 bits.	Yes
DH	Public key crypto system. Typical key/data sizes: 512, 768, 1024 (typical), 2048 bits. Used for key agreement.	No

Table 6: Public Key Functions

Random Number Generators

Algorithm	Specification	FIPS Approved
PSEUDO Random Number Generator	**FIPS 186-2** **ANSI X9.31 1998** .	Yes
Universal Software Based True Random Number Generator	Patented by IBM, EC Pat.No. EP1081591A2, True random number generator that works reliably on variety of platforms without exploiting platform specific features. Entropy evaluation through statistical analysis. Performance: 20-1000 bits/seconds.	No

Table 7: Random Number Generators

5.3. Module Interfaces

As a multi-chip standalone module, the CryptoLite Module's physical interfaces consist of the keyboard, mouse, monitor, serial ports, and network adapters. However, the underlying logical interface to the CryptoLite package is a Java language Application Program Interface (API) documented in the CryptoLite User Guide. The module provides for Control Input with the exported public method calls. Data Input and Output are provided in the variables passed with method calls, and Status Output is provided in the returns and error codes that are documented for each call. The CryptoLite Module is accessed from Java language programs via the inclusion of the package export files and the package class file packaged in JAR format.

5.4. Cryptographic Module Self Tests

The CryptoLite module implements a number of self-tests to check the proper functioning of the Module. *Startup self-tests* are run automatically when the module is first activated. Conditional tests are performed when asymmetric keys are generated. These tests include a continuous random number generator test and pair-wise consistency tests of the generated RSA keys.

Startup Self-Tests

Power-up self-testing is initiated automatically when the CryptoLite module starts loading. (See the CryptoLite Finite State Machine for more details). These tests comprise of the software integrity test and the known answer tests of cryptographic algorithms. Should any of these tests fail; the CryptoLite module will terminate the loading process and generate an exception. The module cannot be used in this state.
The integrity of the module is verified by checking a HMAC of the all of the jar files classes. The Initialization will only succeed if this HMAC is valid.

The CryptoLite module executes the following cryptographic algorithms tests:

- DES KAT
- 3DES KAT
- AES KAT
- SHA-1 KAT
- SHA256 KAT
- SHA384 KAT
- SHA512 KAT
- RSA_SIGN/VERIFICATION
- DSA PARAMETER GENERATION
- DSA_SIGN/VERIFICATION
- DIFFIE-HELLMAN
- RNG

Startup Recovery
Should the startup self tests fail during module initialization the crypto officer should reinstall the complete application.

Conditional Self-Testing
This includes continuous PRNG testing. The very first output block generated by the PRNG is never used for any purpose other than initiating the continuous PRNG test which compares every newly generated block with the previously generated block. The test fails if newly generated PRNG output block matches the previously generated block. In such a case, the Module generates an exception to the calling application. It is the responsibility of the calling application to handle the exception in a FIPS appropriate manner, for example by retrying the PRNG service.

Pair-wise Consistency Self-Testing
The test is run whenever private key is generated by the CryptoLite Module. The private key structure of the Module always contains either the data of the corresponding public key or information sufficient for computing the corresponding public key. If the test fails the Module generates an exception to the calling application. It is the responsibility of the calling application to handle the exception in a FIPS appropriate manner, for example by retrying the key generation service.

Self Test Service
A service is provided which allows the modules self tests to be triggered at any time by a calling application. An error during the self test service will trigger an exception. It is the responsibility of the calling application to handle this error exception in a FIPS appropriate manner.

5.5. Operational Environment

The CryptoLite module is written entirely in the Java programming language that allows for extensive review to confirm security. Applications using CryptoLite functionality are secure from each other due to the fact that each runs in a “Java sandbox” where the firewall protects applet objects from illegal access by other applications. CryptoLite is developed and maintained according to IBM’s internal development standards and tools including CVS (Version 1.11.1p1) are used for configuration management.
The CryptoLite module implements both approved and non-approved services. The calling application controls the cryptographic material as well as the services that use them. It is the applications responsibility to ensure that when in a FIPS compliant mode, only those FIPS approved algorithms are used.

5.6. Module Status

The module communicates any error status asynchronously through the use of exceptions. It is the responsibility of the calling application to handle these exceptions.

6. Roles and Services

6.1. Roles

The IBM CryptoLite module supports two roles, a cryptographic officer role and a user role. There is no maintenance role.

- ROLE_CO: The Cryptographic Officer Role is purely an administrative role and does not involve the use of any of the modules cryptographic services, other than the startup self-tests. The role is not explicitly authenticated but assumed implicitly on implementation of the modules installation and usage sections defined in the security rules section.

- ROLE_USER: The User Role has access to all of the modules services. The role is not explicitly authenticated but assumed implicitly on access of any of the modules services.

Role	Type of Authentication	Authentication Data
Cryptographic Officer Role	None	None
User Role	None	None

Table 8: Roles and Required Identification and Authentication

6.2. Services

The modules services are accessed through API interfaces from the calling application.

Services	User Role
Self Tests	Yes
AES encryption/decryption, MACing and internal key generation services	Yes
Blowfish encryption/decryption, MACing. And internal key generation services	Yes
DES/3DES encryption/decryption and internal key generation services	Yes
Diffie-Hellman key exchange and parameter generation	Yes
DSA signature generation, verification and parameter generation services.	Yes
Key Import and Export services	Yes
HMAC services	Yes
ISO 9796 message padding services	Yes
MD2 secure hashing services	Yes
MD5 secure hashing services	Yes
Modification detection codes based on DES	Yes
RC2 encryption/decryption, MACing and internal parameter generation services	Yes
RC4 encryption and decryption and internal parameter generation services	Yes
Random Number Services	Yes
RSA decryption, encryption and key generation services	Yes
RSA signature and verification services	Yes
SHA-1 secure hashing services	Yes
SHA-256 secure hashing services	Yes
SHA-384 secure hashing services	Yes
SHA-256 secure hashing services	Yes
Unix password encryption (based on a modified DES) services	Yes

Table 10: Services

Self Test Service
A self test service is initiated automatically on module start-up. A module user can access this service at any time using the CL3 class runSelfTests() method.

7. Cryptographically Sensitive Material

7.1. Cryptographic Keys

Key Storage
The CryptoLite module does not provide long-term cryptographic key storage. If an application program makes use of CryptoLite service to implement cryptographic key storage functionality, it is a responsibility of the application program developers to ensure FIPS140-2 compliance of key storing techniques they implement.

Key Protection
The management and allocation of memory is the responsibility of the operating system. It is assumed that a unique process space is allocated for each request, and that the operating system and the underlying central processing unit (CPU) hardware control access to that space. Each instance of the cryptographic module is self-contained within a process space. All keys are associated with the User role.
It is the responsibility of application program developers to protect keys exported from the CryptoLite Module.

Key zeroization
Key objects are normally zeroed and any associated data discarded when the key object is garbage collected through the finalizer method. CryptoLite provides an additional mechanism which helps to ensure key zeroisation through a dispose method. An application can explicitly call this method in order to clear and release key material associated with a key object without waiting for a possible pending invocation of the finalizer method.

Key Import/Export
CryptoLite provides applications key import and export routines such that key material can be used in conjunction with cryptographic services. It is the responsibility of the applications to ensure that these services are used in a FIPS compliant manner.

Key Generation
Key generation uses the FIPS approved RNG algorithm which is based on SHA-1. The RNG has a maximum number of internal states of 2^160, this being limited by the compression function in SHA-1. The RSA and DH key generation algorithms use the RNG engine seeded with 20 bytes of true random data. This true random generator is based on IBM patented technology where statistical analysis used to estimate the entropy of the clock jitter. The internal RNG engine is enhanced using an automatic reseeding policy that insert a true random byte every 128 bytes of output if more than 30 seconds passed since last being reseeded. Applications can additionally provide their own seeding data and also increase the automatic reseeding policy of the internal RNG engine for example to add true random data every 8th byte without time constraint.

8. Security Rules

Operating System
The cryptographic module is dependant on the operating system environment being set up in accordance with FIPS 140-2 specifications. This includes that the host operating system be restricted to a single operator mode. An additional requirement for this cryptographic provider is the availability of a valid commercial grade installation of a Java SDK 1.3.1 or greater JVM.

Application Usage
The application shall ensure that keys are exchange in a FIPS compliant manner
The application shall ensure that only FIPS approved algorithms are used.
The application shall insert a reference to the CryptoLite class in the static area of each of its classes. This will force the recursive initialization of CryptoLite and thus trigger the self tests, during the initialization of the application.
The application shall be ensure that cryptographically sensitive material is not inadvertently output over physical ports

Single User Guidelines
The following explains how to configure a Unix system for single user. The general idea is the same across all Unix variants:

- Remove all login accounts except "root" (the superuser).
- Disable NIS and other name services for users and groups.
- Turn off all remote login, remote command execution, and file transfer daemons.

The Windows Operating Systems can be configured in single user mode by disabling all user accounts except the administrator. This can be done through the Computer Management window of the operating system. Additionally, the operating system must be configured to operate securely and to prevent remote login. This is accomplished by disabling all services (within the Administrative tools) that provide remote access (e.g. – ftp, telnet, ssh, and server) and disallowing multiple operators to log in at once.

9. Notices

AIX, Everyplace, and IBM are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.

Pentium and X-Scale are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.