



IBM Java JCE FIPS 140-2 Cryptographic Module

Security Policy

IBM JAVA JCE FIPS 140-2 Cryptographic Module February 2009
Revision: 1.6.1

Status: Final

1.6.1 Edition (February 2009)

This edition applies to the 1.6.1 Edition of the IBMJCEFIPS – Security Policy and to all subsequent versions until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2009.

All rights reserved. This document may be freely reproduced and distributed in its entirety and without modification.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems in the US and other countries



Table of Contents

Introduction.....	4
Operation of the Cryptographic Module.....	5
Changes from Version 1.2 to 1.3.1	6
Cryptographic Module Specification.....	6
Cryptographic Module Interfaces	10
Cryptographic Module Services	10
Self Test	11
Data Encryption/Decryption and Hashing (Digest).....	12
Key Generation	13
Key Security	13
Signature	14
Secret Key Factory.....	14
KeyFactory.....	15
Cryptographic Module Roles.....	15
Cryptographic Officer role.....	16
Cryptographic User role.....	16
Cryptographic Module Key Management	16
Key Generation	18
Key Storage.....	18
Key Protection	18
Key Zeroization	19
Cryptographic Module Self-Tests.....	19
User Guidance.....	20
Cryptographic Module Operating system environment.....	21
Framework	21
Single user access (operating system requirements).....	22
Java object model.....	23
Operating system restriction	23
Mitigation of other attacks.....	23
Appendix A: Function List	24
A.....	24
C.....	25
D.....	26
E.....	28
F	46
G.....	47
H.....	49
I	50
M.....	69
O.....	69
P	69



Q.....	69
R.....	70
S.....	71
T.....	73
U.....	73
Z.....	73
Notices	74



Introduction

The IBM® Java® JCE (Java Cryptographic Extension) FIPS 140-2 Cryptographic Module (Version 1.3.1) for Multi-platforms is a scalable, multi-purpose cryptographic module that supports FIPS approved cryptographic operations via the Java2 Application Programming Interfaces (APIs). The IBM Java JCE FIPS 140-2 Cryptographic Module (hereafter referred to as IBMJCEFIPS) comprises the following Federal Information Processing Standards (FIPS) 140-2 [Level 1] compliant components:

- IBMJCEFIPS.jar for Solaris®, Windows®, AIX®, z/OS®, AS/400®, Linux® (Red Hat and SuSE®)

In order to meet the requirements set forth in the FIPS publication 140-2, the encryption algorithms utilized by the IBMJCEFIPS provider are isolated into the IBMJCEFIPS provider cryptographic module (hereafter referred to as cryptographic module), which is accessed by the product code via the Java JCE framework APIs. As the IBMJCEFIPS provider utilizes the cryptographic module in an approved manner, the product complies with the FIPS 140-2 requirements when properly configured.

This document focuses on the features and security policy provided by the cryptographic module, and describes how the module is designed to meet FIPS 140-2 compliance.



Operation of the Cryptographic Module

The cryptographic module must be utilized in a secure manner, as described herein, to maintain FIPS 140-2 compliance. It is the application and application administrator's responsibility to understand and deploy the proper configuration for compliance.

The module is available as a software module on multiple platforms. The platforms tested are outlined in the *Cryptographic Module Specification* section of this document. The module must be used in one of the specified environments.

An application utilizes the module through the interfaces specified in the *Cryptographic Module Interfaces* section of this document. A list of the basic services provided through these interfaces may be found in the *Cryptographic Module Services* section of this document. A complete list of all services and details on their usage can be found in the *IBM Java JCE FIPS (IBMJCEFIPS) Cryptographic Module API Javadoc*.

The module provides for two operator roles:

- Crypto Officer
- User

There is no maintenance role in this cryptographic module.

An application must use the IBMJCEFIPS provider to enable the use of appropriate cryptographic functions in a FIPS approved manner. The application calling the IBMJCEFIPS provider must understand the roles of the APIs, Crypto Officer vs. User. The *Cryptographic Module Roles* section of this document details the APIs that apply to each role. In order to use the module in FIPS mode the User must ensure that only FIPS Approved cryptographic algorithms are being invoked and/or algorithms are used in an approved manner.

The module can provide for protection of sensitive data, such as keys or cryptographic contexts. Information on key protection is outlined in the *Cryptographic Module Key Management* section. When the module is initialized, it validates its own integrity, and verifies the algorithms are functioning correctly. The *Cryptographic Module Self-Tests* section details the internal tests performed by the module.



The module's physical security relies on the physical security of the computer. Steps to deploy and maintain this secure environment are outlined in the *User Guidance* section of this document.

Changes from Version 1.2 to 1.3.1

The following was added to the 1.3.1 version of IBMJCEFIPS:

- The non-approved CTS mode has been added for both AES and Triple-DES.
- HMAC SHA-256, HMAC SHA-384 and HMAC SHA-512 support has been added.
- RSA Support for key sizes up to 16384 has been added
- RSA with SHA-256, SHA-384, and SHA-512 signature support added
- DH support for AES derived key added
- DH support for 2048 bit prime modulus added
- DSA support for 2048 bit modulus added
- TLS implementation added
- Auth Hmac for SHA256 truncated to 128 and SHA512 truncated to 256 added

Cryptographic Module Specification

The cryptographic module is a software module, implemented as a Java archive (JAR). The software module is accessible from Java language programs through an application program interface (API). Some of the available API functions are listed below in the *Cryptographic Module Services* section. Usage guidelines and details of the full API function set are available in the *IBM Java JCE FIPS (IBMJCEFIPS) Cryptographic Module API Javadoc*.

The module is validated to the following FIPS 140-2 defined levels:

Overall	Security Level 1
Cryptographic Module Specification	Security Level 1
Cryptographic Module Ports and Interfaces	Security Level 1
Roles, Services, and	Security Level 1



Authentication	
Finite State Model	Security Level 1
Physical Security	n/a
Operational Environment	Security Level 1
Cryptographic Key Management	Security Level 1
EMI/EMC	Security Level 1
Self-Tests	Security Level 1
Design Assurance	Security Level 1
Mitigation of Other Attacks	Security Level 1



As outlined in section 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, and
- 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.

The IBMJCEFIPS provider was tested on a machine running Microsoft Windows XP Professional SP2 operating system in single-user mode with JVM 1.6. The software module maintains compliance when running on the Microsoft Windows 95,[®] Microsoft Windows 98[®], Microsoft Windows Me[®], Microsoft Windows NT[®], Microsoft Windows 2000[®], Microsoft Windows XP[®], and Microsoft Windows Vista[®] operating systems, as well as, JVMs at the 6.x level, 5.x level, and 1.4.x level on those operating systems.

IBM performs testing on AIX, Solaris, HP, Red Hat Linux, SuSE[®] Linux, z/OS[®] and IBM Operating System/400 platforms and affirms IBMJCEFIPS operates correctly on these platforms.

The module supports the following approved algorithms:

Type	Algorithm	Specification
Symmetric Cipher	AES (ECB, CBC, OFB, CFB and PCBC modes)	FIPS 197
	Triple-DES (ECB, CBC, OFB and CFB modes)	FIPS 46-3
Message Digest	SHA1 SHA-256 SHA-384 SHA-512	FIPS 180-2
Message Authentication	HMAC-SHA-1	FIPS 198a



	HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	
Random Number Generation	FIPS 186-2 Appendix 3.1	FIPS 186-2
Digital Signature	DSA (1024 ¹)	FIPS 186-2
Digital Signature	RSA (1024 ² – 16384 ³)	PKCS#1 v1.5

In addition, the module supports the following non-approved algorithms:

Type	Algorithm	Specification
Random Number Generation	Universal Software Based Random Number Generator	Available upon request from IBM. Patented by IBM, EC Pat. No. EP1081591A2, U.S. pat. Pend.
Message Digest	MD5	RCF 1321 (Allowed for use within the TLS protocol).
Asymmetric Cipher	RSA	PKCS #1 with and without blinding (RSASSL) (Allowed in the Approved mode for key transport ⁴)
Key Agreement Primitives	Diffie-Hellman Shared Secret (256 ⁵ –2048)	PKCS #3 (Allowed in Approved mode)
Digital Signature	DSAforSSL	Allowed for use within the TLS protocol
Digital Signature	RSAforSSL	Allowed for use within the TLS protocol

¹ The DSA implementation supports all key sizes from 512-2048, but only 1024 is allowed in approved mode.

² The RSA implementation also supports 512 bit key sizes, but this size may not be used in approved mode.

³ Though IBM JCE FIPS supports key sizes up to 16384 bits, generating or working with keys of this magnitude is not recommended for all uses. Key sizes above 4096 bits are allowed for future-proofing and for specialized use-cases. Application developers using the IBM JCE FIPS jar should regulate the key sizes their applications are permitted to operate upon in order to ensure that the machine cannot be tied up fulfilling requests for large keys and thus ignoring other requests. 16384 bits is the recommended maximum strength with 8192 and 4096 still meeting the majority of current needs.

⁴ Note: Per IG 7.5, only RSA keys of size 1024-15360 bit are acceptable for key transport. RSA keys of size below 1024 bit and above 15360 bit are non-compliant for key transport.

⁵ Note: Diffie-Hellman shared secret primitives provide between 40 and 112 bits of encryption strength.



Symmetric Cipher	AES	PCBC and CTS mode
Symmetric Cipher	Triple-DES	PCBC and CTS mode
MAC	HMAC	Auth HMAC for SHA-256 truncated to 128 and for SHA-512 truncated to 256

Cryptographic Module Interfaces

The cryptographic physical boundary is defined at the perimeter of the computer system enclosure on which the cryptographic module is to be executed, and includes all the hardware components within the enclosure. The cryptographic module interfaces with the Central Processing Unit (CPU) of the respective platform. The RAM and hard disk found on the computer are memory devices that store and execute the cryptographic module and its data.

The cryptographic module is classified as a “multi-chip standalone module” for FIPS 140-2 purposes. Thus, the module’s physical interfaces consist of those found as part of the computer’s hardware, such as the keyboard, mouse, disk drive, CD drive, network adapters, serial and USB ports, monitor, speakers, etc. The module’s logical interface is provided through the documented API.

Each of the FIPS 140-2 defined logical interfaces are implemented as follows:

- Data Input Interface – variables passed in with the API function calls
- Data Output Interface – variables passed back with the API function calls
- Control Input Interface – the API function calls exported from the module
- Status Output Interface – return values and error exceptions provided with the API method calls

Cryptographic Module Services

The module services are accessible from Java language programs through an Application Program Interface (API). The application will be required to call the IBMJCEFIPS provider (as opposed to another JCE provider) through the normal Java 2 mechanisms such as specifically adding the provider name to the



getInstance call as part of the instantiation of a cryptographic object or by placing the IBMJCEFIPS provider higher in the provider list and allowing the JVM to select the first provider that has the requested cryptographic capability. Usage guidelines and details of the API function are available in the *IBM Java JCE FIPS (IBMJCEFIPS) Cryptographic Module API Javadoc*.

The following is a high level description of the basic capabilities available in the cryptographic module (all services are for the user role unless otherwise noted). This is intended to outline the basic services available in the cryptographic module to allow a determination as to whether these services will adequately address the security needs of an application. Usage guidelines and details of all of the API functions are available in the *IBM Java JCE FIPS (IBMJCEFIPS) Cryptographic Module API Javadoc*.

Self Test

This section describes some of the capabilities that are available as they relate to the self test the cryptographic module performs to validate its own integrity and to verify the algorithms are functionally correct.

Services	Description
IsSelfTestInProgress	Identifies if a self test is currently in progress. Call is based on a SelfTest object returned from the getSelfTest call.
GetSelfTestFailure	Returns the exception associated with the self test failure or null if no failure was encountered. Call is based on a SelfTest object returned from the getSelfTest call.
RunSelfTest	Performs the known answer self tests. Call is based on a SelfTest object returned from the getSelfTest call. This is a Cryptographic Officer role call.
IsFipsRunnable	Identifies if the crypto module is runnable, has completed self test with no errors, and is in "Ready" state. Call is based on a SelfTest object returned from the getSelfTest call.
IsFipsCertified	Identifies if the cryptographic module is FIPS 140-2 validated. Call is based on a provider object.



GetFipsLevel	Returns the FIPS 140-2 validation level of the cryptographic module. Call is based on a provider object.
GetSelfTest	Returns a SelfTest object that can be used to execute any of the SelfTest class methods. Call is based on a provider object.
IsFipsApproved	Identifies if the cryptographic operation is FIPS 140-2 validated. Call is based on a cryptographic object.

Data Encryption/Decryption and Hashing (Digest)

This section describes some of the capabilities that are available as they relate to encryption/decryption (Cipher) of data and digesting or hashing (MessageDigest) of data.

Services	Description
getInstance Cipher.getInstance MessageDigest.getInstance	Creates a cryptographic object (Cipher/MessageDigest) for a selected algorithm. Also used to select the cryptographic provider to be used by that object. Cipher allows for Triple-DES, and AES algorithms with various cipher modes and paddings. MessageDigest allows for SHA-1, SHA-256, SHA-384, SHA-512, MD5 hashing.
init Cipher.init MessageDigest.init	Initializes the cryptographic object for use. This includes the mode (encryption or decryption) and the cryptographic key. This call is based on a cryptographic object.
update Cipher.update MessageDigest.update	Updates the cryptographic object with data to be encrypted/decrypted. This call is based on a cryptographic object.
doFinal Cipher.doFinal	Updates the cryptographic object with data to be encrypted/decrypted and returns the data in encrypted or decrypted form (based on the



MessageDigest.doFinal	init). This call is based on a cryptographic object
-----------------------	---

Key Generation

This section describes some of the capabilities that are available as they relate to keys.

Services	Description
getInstance KeyGenerator.getInstance	Creates a cryptographic object (KeyGenerator) for a selected algorithm. Also used to select the cryptographic provider to be used by that object.
init	Initializes the cryptographic object for use. This call is based on a cryptographic object.
GenerateKey	Generates a cryptographic key. This call is based on a cryptographic object.

Services	Description
getInstance KeyPairGenerator.getInstance	Creates a cryptographic object (KeyPairGenerator) for a selected algorithm. Also used to select the cryptographic provider to be used by that object.
initialize	Initializes the cryptographic object for use. This call is based on a cryptographic object.
generateKeyPair	Generates a cryptographic key pair. This call is based on a cryptographic object.

Key Security

In accordance with the FIPS 140-2 standards this cryptographic module provides the user of keys the ability to zero out the key information via a new API.

Service	Description
(crypto key object). zeroize	Zeros out the key(s) associated with a cryptographic object. This call is based on a



	cryptographic object.
--	-----------------------

Signature

This section describes some of the capabilities that are available as they relate to signature generation and verification.

Service	Description
getInstance Signature.getInstance	Creates a cryptographic object (Signature) for a selected algorithm. Also used to select the cryptographic provider to be used by that object.
InitSign	Initializes the cryptographic object for use. This includes the cryptographic private key. This call is based on a cryptographic object.
Update	Update a byte or byte array in the data to be signed or verified. This call is based on a cryptographic object.
Sign	Get message digest for all the data thus far updated, then sign the message digest. This call is based on a cryptographic object.
InitVerify	Initializes the cryptographic object for use. This includes the cryptographic public key. This call is based on a cryptographic object.
verify	Verify the signature (compare the result with the message digest). This call is based on a cryptographic object.

Secret Key Factory

This section describes some of the capabilities that are available as they relate to symmetric keys.

Service	Description
GetInstance	Creates a cryptographic object (SecretKeyFactory) for a selected algorithm.



	Also used to select the cryptographic provider to be used by that object.
GetKeySpec	Returns a specification (key material) of the given key in the requested format.
generateSecret	Generates a SecretKey object from the provided key specification (key material).

KeyFactory

This section describes some of the capabilities that are available as they relate to asymmetric keys.

Service	Description
GetInstance	Creates a cryptographic object (KeyFactory) for a selected algorithm. Also used to select the cryptographic provider to be used by that object
GeneratePublic	Generates a public key object from the provided key specification (key material).
GeneratePrivate	Generates a private key object from the provided key specification (key material).
getKeySpec	Returns a specification (key material) of the given key object in the requested format.

Cryptographic Module Roles

The cryptographic module implements both a Crypto Officer and a User role, meeting all FIPS 140-2 level 1 requirements for roles and services. A Maintenance Role is not implemented. The module does not provide authentication for any role.

All the services in the previous section are available to both the roles of the module. The role assumed by the operator is implicit based upon the service being invoked.



Cryptographic Officer role

The Crypto Officer role has responsibility for initiating on-demand self test diagnostics. This is accomplished through the runSelfTest API call described in the *IBMJCEFIPS provider Cryptographic Module API* document. This role is also responsible for installing and removing the module.

Cryptographic User role

The User role has the responsibility for operating cryptographic functions on data.

User guidance information is available in the *IBMJCEFIPS provider Cryptographic Module API* document.

There is no maintenance role.

Only one role is implicitly active in the module at a time.

Cryptographic Module Key Management

The module supports the use of the following cryptographic keys: Diffie-Hellman public/private keys, Triple-DES, AES, RSA public/private keys, DSA public/private keys, HMAC-SHA1, HMAC-SHA 256, HMAC-SHA 384, and HMAC-SHA 512.

CSP/Key Name	Key Type	Generation/Input	Output	Storage	Zeroization	Roles
AES	AES	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
DSA	DSA	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
HMAC-SHA1	HMAC-SHA1	Generated internally using approved	Plaintext	RAM	Zeroized when zeroize() method is	Crypto Officer, Crypto User



		RNG			called	User
HMAC-SHA256	HMAC-SHA256	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
HMAC-SHA384	HMAC-SHA384	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
HMAC-SHA512	HMAC-SHA512	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
RSA	RSA	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
Triple-DES	Triple-DES	Generated internally using approved RNG	Plaintext	RAM	Zeroized when zeroize() method is called	Crypto Officer, Crypto User
Integrity Key	HMAC-SHA1	Hardcoded	Never	Harddisk	When the module is deleted	Crypto Officer
PRNG Seed Key	FIPS 186-2 Seed Key	Generated internally using non-approved RNG	Never	RAM	When a new seed key is generated	Crypto Officer, Crypto User
PRNG Seed	FIPS 186-2 Seed	Generated internally using non-approved RNG	Never	RAM	When a new seed is generated	Crypto Officer, Crypto User



Operators of the module have full access to key material. These keys are accessed by calling the various cryptographic services specified in the *IBMJCEFIPS provider Cryptographic Module API* Javadoc.

Key Generation

Symmetric keys are generated using the FIPS Approved FIPS 186-2 (Appendix 3.1 and 3.3) pseudo random-number generation algorithm.

DSA parameters, along with public and private keys are generated using the random number algorithms as defined in FIPS 186-2. DSA and RSA key pairs are generated as defined in FIPS 186-2.

IBM has invented a scheme to generate randomness on a wide range of computer systems. The patented scheme, called the Universal Software Based True Random Number Generator, utilizes random events influenced by concurrent activities in the system (e.g. interrupts, process scheduling, etc). The run time of the algorithm will vary depending of the state of the system at the time of seed generation, and will be dependent on the type of system. The Universal Software Based True Random Number Generator is used to create a random seed value that is used in the PRNG algorithm, if a seed value is not supplied to the PRNG by the user.

Key Storage

We do not support key storage within the IBMJCEFIPS cryptographic module.

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. Only one instance of the module is available in each process space. All keys are associated with the User role.



Key Zeroization

All cryptographic keys and contexts are zeroized when an operator:

- Disposes of a key using the zeroize API call for that key object.
- When Java garbage collection is performed for an object no longer referenced, as part of the objects finalize method.
- Powers off the module by unloading it from memory

Cryptographic Module Self-Tests

When an application references the cryptographic module within the JVM in its process space, an initialization routine is called by the JVM before control is handed to the application. This initialization route automatically executes the power up tests to ensure correct operation of the cryptographic algorithms.

The integrity of the module is verified by performing a HMAC-SHA1 validation of the cryptographic module's classes contained in the module's jar file. The initialization route will only succeed if the HMAC is valid.

Power-up self-tests include known answer tests for the RSA, Diffie-Hellman, SHA1, SHA-256, SHA-384, SHA-512, Triple-DES, AES, DSA, HMAC SHA1, HMAC SHA256, HMAC SHA384, HMAC SHA512 cryptographic algorithms and pseudo random number generation. Should any self-test fail, the module transitions to the Error state.

These self tests can also be run on demand by the cryptographic officer via the runSelfTest method.

Additionally, conditional tests are performed when asymmetric keys are generated and random number generators are invoked. These tests include a continuous random number generator tests for both the FIPS-approved PRNG and the non FIPS-approved TRNG used to seed it as well as pair-wise consistency tests of the generated DSA and RSA keys.



User Guidance

Programming practices

This section contains guidance for application programmers to avoid practices that could potentially compromise the secure use of this cryptographic module.

- Zeroize - the zeroize method should be used when a cryptographic key object is no longer needed to remove the key from memory. While normal Java garbage collection will zeroize the key from memory as part of the object finalizer method it is a safer coding practice to explicitly call the zeroize method when an application is finished with a key object.
- Statics – To ensure that each cryptographic object is unique and accessible only by the individual user it is important not to use static objects, as all users of the JVM share these objects.
- As the Java architecture creates objects that are unique to the application and this allows for “single” user access to the cryptographic operations and data it is recommended that an application not create static objects. Static objects are shared in the Java architecture and the creation of a static object would be counter to the unique object method of controlling access and data.
- An application that wishes to use FIPS validated cryptography must use the IBM Secure Random algorithm associated with the IBMJCEFIPS provider for the source of random data needed by algorithms.
- RSA Cryptographic Cipher may only be used to Encrypt and Decrypt keys for transport to stay within the boundaries of the Approved Mode of FIPS 140-2 Level 1.
- One way to help alleviate performance problems is by creating a single source of randomness (IBMSecureRandom or FIPSPRNG) and using that object whenever possible.
- MD5, RSAforSSL and DSAforSSL can only be used if the user is implementing the TLS protocol for Secure Sockets. Any other use will cause the application to be in non-compliance.

Installation and Security rules for using IBMJCEFIPS

This section contains guidance for the installation and use of the FIPS 140-2 level 1 cryptographic module.



The IBMJCEFIPS provider jar file must be accessible via the Java CLASSPATH and should be installed in the directory lib/ext as this is a secure location and is also automatically available via the JVM without a CLASSPATH update.

The application will be required to call the IBMJCEFIPS provider (as opposed to another JCE provider) through the normal Java 2 mechanisms such as specifically adding the provider name to the getInstance call as part of the instantiation of a cryptographic object or by placing the IBMJCEFIPS provider higher in the provider list (in java.security) and allowing the JVM to select the first provider that has the requested cryptographic capability.

Cryptographic Module Operating system environment

Framework

The cryptographic module is dependent on the operating system environment being set up in accordance with FIPS 140-2 specifications. For this cryptographic provider a valid commercial grade installation of a Java SDK 1.3.1 or higher JVM must be available.

A valid commercial grade installation of a Java SDK 1.3.1 or higher JVM that includes the Java Cryptographic Extension framework (Version 1.2.1) is required. (Please note that a JVM at 1.4.0 or higher already contains the JCE framework). In addition to the SDK and the JCE framework the IBMJCEFIPS provider is required.

The following is a brief overview of the JCE framework (A more detailed explanation of this framework is available at <http://java.sun.com/products/jce/doc/guide/HowToImplAProvider.html#MutualAuth>)

In order to prevent unauthorized providers from plugging into the JCE 1.2.1 framework (herein referred to as "JCE 1.2.1"), and to assure authorized providers of the integrity and authenticity of the JCE 1.2.1 that they plug into, JCE 1.2.1 and its providers will engage in mutual authentication. Only providers that have authenticated JCE 1.2.1, and who in turn have been authenticated by JCE 1.2.1, will become usable in the JCE 1.2.1 environment. For more information about this, please see the above web page.

In addition, each provider does do self-integrity checking to ensure that the JAR file containing its code has not been tampered with. The JCE 1.2.1 framework is



digitally signed. Providers that provide implementations for JCE 1.2.1 services must also be digitally signed. Authentication includes verification of those signatures and ensuring the signatures were generated by trusted entities. Certain Certificate Authorities are deemed to be "trusted" and any code signed using a certificate that can be traced up a certificate chain to a certificate for one of the trusted Certificate Authorities are considered trusted. Both JCE 1.2.1 and provider packages do embed within themselves the bytes for the certificates for the relevant trusted Certificate Authorities. At runtime, the embedded certificates will be used in determining whether or not code is authentic. Currently, there are two trusted Certification Authorities: Sun Microsystems' JCE Code Signing CA, and IBM JCE Code Signing CA.

In order to insure that an application is using the FIPS validated cryptographic module, the application is required to call the IBMJCEFIPS provider (as opposed to another JCE provider) through the normal Java 2 mechanisms such as specifically adding the provider name to the getInstance call as part of the instantiation of a cryptographic object or by placing the IBMJCEFIPS provider higher in the provider list and allowing the JVM to select the first provider that has the requested cryptographic capability.

Single user access (operating system requirements)

This cryptographic module adheres to the FIPS 140-2 level 1 requirement that the operating system must be restricted to a single operator mode (concurrent operators are explicitly excluded). The following explains how to configure a Unix system for single user. The general idea is 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 a 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 can be done by disabling any service (within the Administrative tools) that provider remote access (e.g. – ftp, telnet, ssh, and server) and disallowing multiple operators to log in at once.



Java object model

The use of Java objects within the cryptographic module. In Java each cryptographic object is unique. Thus when an application generates a cryptographic object for use that object is unique to that instance of the application. In this regard other processes have no access to that object and can therefore not interrupt or gain access to the information or activities contained within that object. In this way the cryptographic module protects the single users control of the cryptographic activities and data.

Further as the Self Test class is a Java static object there can be only one instance of that class in the JVM and that instance controls the Self Test activities. In other words if the Self Test fails, then no cryptographic objects for the IBMJCEFIPS provider in the JVM will be operational as the cryptographic module would be in "Error" state.

As the Java architecture creates objects that are unique to the application and this allows for "single" user access to the cryptographic operations and data. It is recommended that an application not create static objects. Static objects are shared in the Java architecture and the creation of a static object would be counter to the unique object method of controlling access and data.

Operating system restriction

The operation of the cryptographic module is assumed to be in single user mode in that only one user is on the system at any point in time.

Mitigation of other attacks

The IBMJCEFIPS provider has been obfuscated. The commercial product KlassMaster provides code obfuscation. This level of optimized code makes it difficult to decompile and reuse the derived source code. IBM's tests with popular de-compilers (e.g. Jasmine) has shown that de-compiled IBMJCEFIPS code for Java code cannot be compiled and used without extensive alteration

RSA Blinding has been added to the RSA Signing and RSA encryption function to help mitigate timing attacks.

No other mitigation of other attacks is provided.



References

[1] National Institute of Standards and Technology. May 2001. *Security Requirements for Cryptographic Modules*. Federal Information Processing Standards Publication 140-2.

[2] National Institute of Standards and Technology. November 2001. *AES Key Wrap Specification*. Internet. 22 April 2002.
<http://csrc.nist.gov/encryption/kms/key-wrap.pdf>

Appendix A: Function List

The following is a list of the public functions found in this module. Please refer to the *IBM Java JCE FIPS (IBMJCEFIPS) Cryptographic Module API* document.

A

`addIdentity(Identity)` - Method in class `com.ibm.crypto.fips.provider.IdentityDatabase`
Adds an identity to the database.

`AESCipher` - Class in `com.ibm.crypto.fips.provider`

This class implements the AES algorithm in its various modes (ECB, CFB, OFB, CBC, PCBC) and padding schemes (PKCS5Padding, NoPadding).

`AESCipher()` - Constructor for class `com.ibm.crypto.fips.provider.AESCipher`

Creates an instance of AES cipher with default ECB mode and PKCS5Padding.

`AESCipher(String, String)` - Constructor for class `com.ibm.crypto.fips.provider.AESCipher`

Creates an instance of AES cipher with the requested mode and padding.

`AESKeyFactory` - Class in `com.ibm.crypto.fips.provider`

This class implements the AES key factory of the IBMJCEFIPS provider.

`AESKeyFactory()` - Constructor for class `com.ibm.crypto.fips.provider.AESKeyFactory`

Verify the JCE framework in the constructor.

`AESKeyGenerator` - Class in `com.ibm.crypto.fips.provider`

This class generates a secret key for use with the AES algorithm.

`AESKeyGenerator()` - Constructor for class

`com.ibm.crypto.fips.provider.AESKeyGenerator`

Verify the JCE framework in the constructor.

`AESKeySpec` - Class in `com.ibm.crypto.fips.provider`

This class specifies a AES key.

`AESKeySpec(byte[])` - Constructor for class `com.ibm.crypto.fips.provider.AESKeySpec`

Uses the bytes in `key` as the key material for the AES key.

`AESKeySpec(byte[], int, int)` - Constructor for class `com.ibm.crypto.fips.provider.AESKeySpec`



Uses the bytes in `key`, beginning at `offset` inclusive, as the key material for the AES key.

AESParameters - Class in `com.ibm.crypto.fips.provider`
This class implements the parameter (IV) used with the AES algorithm in feedback-mode.

AESParameters() - Constructor for class `com.ibm.crypto.fips.provider.AESParameters`

AESSecretKey - Class in `com.ibm.crypto.fips.provider`
This class represents a AES key.

AESSecretKey(byte[]) - Constructor for class `com.ibm.crypto.fips.provider.AESSecretKey`
Create a AES key from a given key

AESSecretKey(byte[], int) - Constructor for class `com.ibm.crypto.fips.provider.AESSecretKey`
Uses the first 16, 20, or 24 bytes (T) in `key`, beginning at `offset`, as the AES key.

AlgorithmStatus - Interface in `com.ibm.crypto.fips.provider`

C

CBC_MODE - Static variable in class `com.ibm.crypto.fips.provider.DESedeCipher`

CFB_MODE - Static variable in class `com.ibm.crypto.fips.provider.DESedeCipher`

cipherMode - Variable in class `com.ibm.crypto.fips.provider.DESedeCipher`

CipherWithWrappingSpi - Class in `com.ibm.crypto.fips.provider`
This class extends the `javax.crypto.CipherSpi` class with a concrete implementation of the methods for wrapping and unwrapping keys.

CipherWithWrappingSpi() - Constructor for class `com.ibm.crypto.fips.provider.CipherWithWrappingSpi`

clone() - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`

clone() - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`

clone() - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`

clone() - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`

clone() - Method in class `com.ibm.crypto.fips.provider.SHA`
Clones this object.

clone() - Method in class `com.ibm.crypto.fips.provider.SHA2`
Clones this object.

clone() - Method in class `com.ibm.crypto.fips.provider.SHA3`
Clones this object.



clone() - Method in class com.ibm.crypto.fips.provider.SHA5

Clones this object.

com.ibm.crypto.fips.provider - package com.ibm.crypto.fips.provider

D

DatawithDSA - Class in com.ibm.crypto.fips.provider

DatawithDSA() - Constructor for class com.ibm.crypto.fips.provider.DatawithDSA

Constructs a new instance of this class.

DatawithRSA - Class in com.ibm.crypto.fips.provider

This class implements signature without this algorithm doing the hashing with RSA

DatawithRSA() - Constructor for class com.ibm.crypto.fips.provider.DatawithRSA

Construct a blank RSA object.

DESedeCipher - Class in com.ibm.crypto.fips.provider

This class implements the Triple-DES algorithm (DES-EDE) in its various modes (ECB, CFB, OFB, CBC, PCBC) and padding schemes (PKCS5Padding, NoPadding).

DESedeCipher() - Constructor for class com.ibm.crypto.fips.provider.DESedeCipher

Creates an instance of DESede cipher with default ECB mode and PKCS5Padding.

DESedeCipher(String, String) - Constructor for class

com.ibm.crypto.fips.provider.DESedeCipher

Creates an instance of DESede cipher with the requested mode and padding.

DESedeKey - Class in com.ibm.crypto.fips.provider

This class represents a DES-EDE key.

DESedeKey(byte[]) - Constructor for class com.ibm.crypto.fips.provider.DESedeKey

Creates a DES-EDE key from a given key.

DESedeKey(byte[], int) - Constructor for class com.ibm.crypto.fips.provider.DESedeKey

Uses the first 24 bytes in *key*, beginning at *offset*, as the DES-EDE key

DESedeKeyFactory - Class in com.ibm.crypto.fips.provider

This class implements the DES-EDE key factory of the IBMJCEFIPS provider.

DESedeKeyFactory() - Constructor for class

com.ibm.crypto.fips.provider.DESedeKeyFactory

Verify the JCE framework in the constructor.

DESedeKeyGenerator - Class in com.ibm.crypto.fips.provider

This class generates a Triple-DES key.

DESedeKeyGenerator() - Constructor for class

com.ibm.crypto.fips.provider.DESedeKeyGenerator

Verify the JCE framework in the constructor.

DESedeParameters - Class in com.ibm.crypto.fips.provider

This class implements the parameter (IV) used with the Triple-DES algorithm in feedback-mode.

DESedeParameters() - Constructor for class

com.ibm.crypto.fips.provider.DESedeParameters



DHKeyAgreement - Class in com.ibm.crypto.fips.provider

This class implements the Diffie-Hellman key agreement protocol between any number of parties.

DHKeyAgreement() - Constructor for class
com.ibm.crypto.fips.provider.DHKeyAgreement

Verify the JCE framework in the constructor.

DHKeyFactory - Class in com.ibm.crypto.fips.provider

This class implements the Diffie-Hellman key factory of the IBMJCEFIPS provider.

DHKeyFactory() - Constructor for class com.ibm.crypto.fips.provider.DHKeyFactory

Verify the JCE framework in the constructor.

DHKeyPairGenerator - Class in com.ibm.crypto.fips.provider

This class represents the key pair generator for Diffie-Hellman key pairs.

DHKeyPairGenerator() - Constructor for class
com.ibm.crypto.fips.provider.DHKeyPairGenerator

DHParameterGenerator - Class in com.ibm.crypto.fips.provider

DHParameterGenerator() - Constructor for class
com.ibm.crypto.fips.provider.DHParameterGenerator

DHParameters - Class in com.ibm.crypto.fips.provider

This class implements the parameter set used by the Diffie-Hellman key agreement as defined in the PKCS #3 standard.

DHParameters() - Constructor for class com.ibm.crypto.fips.provider.DHParameters

DHPrivateKey - Class in com.ibm.crypto.fips.provider

A private key in PKCS#8 format for the Diffie-Hellman key agreement algorithm.

DHPrivateKey(BigInteger, BigInteger, BigInteger) - Constructor for class
com.ibm.crypto.fips.provider.DHPrivateKey

Make a DH private key out of a private value x , a prime modulus p , and a base generator g .

DHPrivateKey(BigInteger, BigInteger, BigInteger, int) - Constructor for class
com.ibm.crypto.fips.provider.DHPrivateKey

Make a DH private key out of a private value x , a prime modulus p , a base generator g , and a private-value length l .

DHPrivateKey(byte[]) - Constructor for class
com.ibm.crypto.fips.provider.DHPrivateKey

Make a DH private key from its DER encoding (PKCS #8).

DHPublicKey - Class in com.ibm.crypto.fips.provider

A public key in X.509 format for the Diffie-Hellman key agreement algorithm.

DHPublicKey(BigInteger, BigInteger, BigInteger) - Constructor for class
com.ibm.crypto.fips.provider.DHPublicKey

Make a DH public key out of a public value y , a prime modulus p , and a base generator g .



DHPublicKey(BigInteger, BigInteger, BigInteger, int) - Constructor for class com.ibm.crypto.fips.provider.DHPublicKey
Make a DH public key out of a public value y , a prime modulus p , a base generator g , and a private-value length l .

DHPublicKey(byte[]) - Constructor for class com.ibm.crypto.fips.provider.DHPublicKey
Make a DH public key from its DER encoding (X.509).

DSAKeyFactory - Class in com.ibm.crypto.fips.provider
This class is a concrete implementation of key factory for DSA.

DSAKeyFactory() - Constructor for class com.ibm.crypto.fips.provider.DSAKeyFactory
Constructs a new instance of this class.

DSAKeyPairGenerator - Class in com.ibm.crypto.fips.provider
This class is a concrete implementation for the generation of a pair of DSA keys

DSAKeyPairGenerator() - Constructor for class com.ibm.crypto.fips.provider.DSAKeyPairGenerator

DSAParameterGenerator - Class in com.ibm.crypto.fips.provider
This class generates parameters for the DSA signature.

DSAParameterGenerator() - Constructor for class com.ibm.crypto.fips.provider.DSAParameterGenerator
Constructs a new instance of this class.

DSAParameters - Class in com.ibm.crypto.fips.provider
This class implements Digital Signature Algorithm parameters specified by com.ibm.crypto.fips.provider 186 standard.

DSAParameters() - Constructor for class com.ibm.crypto.fips.provider.DSAParameters

DSAPrivateKey - Class in com.ibm.crypto.fips.provider
This class represents an X.509 private key for the DSA Algorithm.

DSAPrivateKey(BigInteger, BigInteger, BigInteger, BigInteger) - Constructor for class com.ibm.crypto.fips.provider.DSAPrivateKey
Create a DSA private key from x , p , q , and g .

DSAPrivateKey(byte[]) - Constructor for class com.ibm.crypto.fips.provider.DSAPrivateKey
Create a DSA private key from its DER encoding (PKCS#8)

DSAPublicKey - Class in com.ibm.crypto.fips.provider
This class represents an X.509 public key for the DSA Algorithm.

DSAPublicKey(BigInteger, BigInteger, BigInteger, BigInteger) - Constructor for class com.ibm.crypto.fips.provider.DSAPublicKey
Create a new DSA public key from y , p , q , and g .

DSAPublicKey(byte[]) - Constructor for class com.ibm.crypto.fips.provider.DSAPublicKey
Make a DSA public key from its DER encoding (X.509).

E

ECB_MODE - Static variable in class com.ibm.crypto.fips.provider.DESedeCipher



engineDigest() - Method in class com.ibm.crypto.fips.provider.SHA

engineDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA

engineDigest() - Method in class com.ibm.crypto.fips.provider.SHA2

Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

engineDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA2

Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

engineDigest() - Method in class com.ibm.crypto.fips.provider.SHA3

Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

engineDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA3

Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

engineDigest() - Method in class com.ibm.crypto.fips.provider.SHA5

Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

engineDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA5

Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

engineDoFinal(byte[], int, int) - Method in class

com.ibm.crypto.fips.provider.AESCipher

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

engineDoFinal(byte[], int, int, byte[], int) - Method in class

com.ibm.crypto.fips.provider.AESCipher

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

engineDoFinal(byte[], int, int) - Method in class

com.ibm.crypto.fips.provider.DESedeCipher

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

engineDoFinal(byte[], int, int, byte[], int) - Method in class

com.ibm.crypto.fips.provider.DESedeCipher

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

engineDoFinal() - Method in class com.ibm.crypto.fips.provider.HmacSHA1

Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

engineDoFinal() - Method in class com.ibm.crypto.fips.provider.HmacSHA256

Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

engineDoFinal() - Method in class com.ibm.crypto.fips.provider.HmacSHA384



Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`engineDoFinal()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`engineDoFinal(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`engineDoFinal(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`engineDoFinal(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`engineDoFinal(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`engineDoFinal(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.TDCNP`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`engineDoFinal(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.TDCNP`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`engineDoPhase(Key, boolean)` - Method in class `com.ibm.crypto.fips.provider.DHKeyAgreement`
Executes the next phase of this key agreement with the given key that was received from one of the other parties involved in this key agreement.

`engineGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.AESKeyGenerator`
Generates a AES key.

`engineGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.DESedeKeyGenerator`
Generates the Triple-DES key.

`engineGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`
Generates an HMAC-SHA1 key.

`engineGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`
Generates an HMAC-SHA256 key.

`engineGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Generates an HMAC-SHA384 key.

`engineGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`



Generates an HMAC-SHA512 key.
engineGenerateParameters() - Method in class
com.ibm.crypto.fips.provider.DHParameterGenerator
Generates the parameters.
engineGenerateParameters() - Method in class
com.ibm.crypto.fips.provider.DSAParameterGenerator
Answers the newly generated parameters.
engineGeneratePrivate(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DHKeyFactory
Generates a private key object from the provided key specification (key material).
engineGeneratePrivate(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DSAKeyFactory
Generates a private key for the given key specification.
engineGeneratePrivate(KeySpec) - Method in class
com.ibm.crypto.fips.provider.RSAKeyFactory
Generates a private key object from the provided key specification (key material).
engineGeneratePublic(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DHKeyFactory
Generates a public key object from the provided key specification (key material).
engineGeneratePublic(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DSAKeyFactory
Generates a public key for the given key specification.
engineGeneratePublic(KeySpec) - Method in class
com.ibm.crypto.fips.provider.RSAKeyFactory
Generates a public key object from the provided key specification (key material).
engineGenerateSecret(KeySpec) - Method in class
com.ibm.crypto.fips.provider.AESKeyFactory
Generates a `SecretKey` object from the provided key specification (key material).
engineGenerateSecret(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyFactory
Generates a `SecretKey` object from the provided key specification (key material).
engineGenerateSecret() - Method in class
com.ibm.crypto.fips.provider.DHKeyAgreement
Generates the shared secret and returns it in a new buffer.
engineGenerateSecret(byte[], int) - Method in class
com.ibm.crypto.fips.provider.DHKeyAgreement
Generates the shared secret, and places it into the buffer `sharedSecret`,
beginning at `offset`.
engineGenerateSecret(String) - Method in class
com.ibm.crypto.fips.provider.DHKeyAgreement
Creates the shared secret and returns it as a secret key object of the requested
algorithm type.
engineGenerateSeed(int) - Method in class com.ibm.crypto.fips.provider.SecureRandom

engineGetBlockSize() - Method in class com.ibm.crypto.fips.provider.AESCipher
Returns the block size (in bytes).



engineGetBlockSize() - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Returns the block size (in bytes).

engineGetBlockSize() - Method in class com.ibm.crypto.fips.provider.RSA

Returns the block size (in bytes).

engineGetBlockSize() - Method in class com.ibm.crypto.fips.provider.RSASSL

Returns the block size (in bytes).

engineGetBlockSize() - Method in class com.ibm.crypto.fips.provider.TDCNP

Returns the block size (in bytes).

engineGetDigestLength() - Method in class com.ibm.crypto.fips.provider.SHA

Return the digest length in bytes

engineGetDigestLength() - Method in class com.ibm.crypto.fips.provider.SHA2

Return the digest length in bytes

engineGetDigestLength() - Method in class com.ibm.crypto.fips.provider.SHA3

Return the digest length in bytes

engineGetDigestLength() - Method in class com.ibm.crypto.fips.provider.SHA5

Return the digest length in bytes

engineGetEncoded() - Method in class com.ibm.crypto.fips.provider.AESParameters

engineGetEncoded(String) - Method in class

com.ibm.crypto.fips.provider.AESParameters

engineGetEncoded() - Method in class com.ibm.crypto.fips.provider.DESedeParameters

engineGetEncoded(String) - Method in class

com.ibm.crypto.fips.provider.DESedeParameters

engineGetEncoded() - Method in class com.ibm.crypto.fips.provider.DHParameters

engineGetEncoded(String) - Method in class

com.ibm.crypto.fips.provider.DHParameters

engineGetEncoded() - Method in class com.ibm.crypto.fips.provider.DSAParameters

Returns the parameters in encoded bytes.

engineGetEncoded(String) - Method in class

com.ibm.crypto.fips.provider.DSAParameters

Returns the parameters in encoded bytes with encoding method specified.

engineGetIV() - Method in class com.ibm.crypto.fips.provider.AESCipher

Returns the initialization vector (IV) in a new buffer.

engineGetIV() - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Returns the initialization vector (IV) in a new buffer.

engineGetIV() - Method in class com.ibm.crypto.fips.provider.RSA

Returns the initialization vector (IV) in a new buffer.

engineGetIV() - Method in class com.ibm.crypto.fips.provider.RSASSL

Returns the initialization vector (IV) in a new buffer.

engineGetIV() - Method in class com.ibm.crypto.fips.provider.TDCNP

Returns the initialization vector (IV) in a new buffer.



`engineGetKeySize(Key)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Returns the key size of the given key object.

`engineGetKeySize(Key)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Returns the key size of the given key object.

`engineGetKeySize(Key)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Returns the key size of the given key object.

`engineGetKeySize(Key)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Returns the key size of the given key object.

`engineGetKeySize(Key)` - Method in class `com.ibm.crypto.fips.provider.TDCNP`
Returns the key size of the given key object.

`engineGetKeySpec(SecretKey, Class)` - Method in class `com.ibm.crypto.fips.provider.AESKeyFactory`
Returns a specification (key material) of the given key in the requested format.

`engineGetKeySpec(SecretKey, Class)` - Method in class `com.ibm.crypto.fips.provider.DESedeKeyFactory`
Returns a specification (key material) of the given key in the requested format.

`engineGetKeySpec(Key, Class)` - Method in class `com.ibm.crypto.fips.provider.DHKeyFactory`
Returns a specification (key material) of the given key object in the requested format.

`engineGetKeySpec(Key, Class)` - Method in class `com.ibm.crypto.fips.provider.DSAKeyFactory`
Answers a key specification for a given key.

`engineGetKeySpec(Key, Class)` - Method in class `com.ibm.crypto.fips.provider.RSAKeyFactory`
Returns a specification (key material) of the given key object in the requested format.

`engineGetMacLength()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Returns the length of the HMAC in bytes.

`engineGetMacLength()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`
Returns the length of the HMAC in bytes.

`engineGetMacLength()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Returns the length of the HMAC in bytes.

`engineGetMacLength()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Returns the length of the HMAC in bytes.

`engineGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).

`engineGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).

`engineGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.RSA`



- Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).
- `engineGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).
- `engineGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.TDCNP`
Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Deprecated.
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Deprecated.
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `engineGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA5withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `engineGetParameters()` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Returns the parameters used with this cipher.
- `engineGetParameters()` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Returns the parameters used with this cipher.
- `engineGetParameters()` - Method in class `com.ibm.crypto.fips.provider.RSA`
Returns the parameters used with this cipher.
- `engineGetParameters()` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Returns the parameters used with this cipher.
- `engineGetParameters()` - Method in class `com.ibm.crypto.fips.provider.TDCNP`



Returns the parameters used with this cipher.
engineGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.AESParameters

engineGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.DESedeParameters

engineGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.DHParameters

engineGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.DSAParameters

Return the parameter spec used by this parameter instance.
engineInit(int, Key, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESCipher

Initializes this cipher with a key and a source of randomness.
engineInit(int, Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESCipher

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

engineInit(int, Key, AlgorithmParameters, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESCipher
Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

engineInit(SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESKeyGenerator
Initializes this key generator.

engineInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESKeyGenerator
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

engineInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESKeyGenerator
Initializes this key generator for a certain keysize, using the given source of randomness.

engineInit(AlgorithmParameterSpec) - Method in class
com.ibm.crypto.fips.provider.AESParameters

engineInit(byte[]) - Method in class com.ibm.crypto.fips.provider.AESParameters

engineInit(byte[], String) - Method in class com.ibm.crypto.fips.provider.AESParameters

engineInit(int, Key, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeCipher
Initializes this cipher with a key and a source of randomness.



engineInit(int, Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeCipher

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

engineInit(int, Key, AlgorithmParameters, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeCipher

engineInit(SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator

Initializes this key generator.

engineInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator

Initializes this key generator with the specified parameter set and a user-provided source of randomness.

engineInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator

Initializes this key generator for a certain keysize, using the given source of randomness.

engineInit(AlgorithmParameterSpec) - Method in class
com.ibm.crypto.fips.provider.DESedeParameters

engineInit(byte[]) - Method in class com.ibm.crypto.fips.provider.DESedeParameters

engineInit(byte[], String) - Method in class
com.ibm.crypto.fips.provider.DESedeParameters

engineInit(Key, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHKeyAgreement

Initializes this key agreement with the given key and source of randomness.

engineInit(Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHKeyAgreement

Initializes this key agreement with the given key, set of algorithm parameters, and source of randomness.

engineInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHParameterGenerator

Initializes this parameter generator for a certain keysize and source of randomness.

engineInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHParameterGenerator

Initializes this parameter generator with a set of parameter generation values, which specify the size of the prime modulus and the size of the random exponent, both in bits.

engineInit(AlgorithmParameterSpec) - Method in class
com.ibm.crypto.fips.provider.DHParameters

engineInit(byte[]) - Method in class com.ibm.crypto.fips.provider.DHParameters



`engineInit(byte[], String)` - Method in class `com.ibm.crypto.fips.provider.DHParameters`

`engineInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.DSAParameterGenerator`

Initializes the receiver with the specified parameters and source of randomness.

`engineInit(int, SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.DSAParameterGenerator`

Initializes the receiver with the specified parameter size and source of randomness.

`engineInit(AlgorithmParameterSpec)` - Method in class

`com.ibm.crypto.fips.provider.DSAParameters`

Initialize the `DSAParameters` by a `DSAParameterSpec`

`engineInit(byte[])` - Method in class `com.ibm.crypto.fips.provider.DSAParameters`

Initialize the `DSAParameters` by encoded bytes

`engineInit(byte[], String)` - Method in class `com.ibm.crypto.fips.provider.DSAParameters`

Initialize the `DSAParameters` by encoded bytes with the specified decoding method.

`engineInit(Key, AlgorithmParameterSpec)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA1`

Initializes the HMAC with the given secret key and algorithm parameters.

`engineInit(SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`

Initializes this key generator.

`engineInit(AlgorithmParameterSpec, SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`

Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`engineInit(int, SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`

Initializes this key generator for a certain keysize, using the given source of randomness.

`engineInit(Key, AlgorithmParameterSpec)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA256`

Initializes the HMAC with the given secret key and algorithm parameters.

`engineInit(SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`

Initializes this key generator.

`engineInit(AlgorithmParameterSpec, SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`

Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`engineInit(int, SecureRandom)` - Method in class

`com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`

Initializes this key generator for a certain keysize, using the given source of randomness.



`engineInit(Key, AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Initializes the HMAC with the given secret key and algorithm parameters.

`engineInit(SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Initializes this key generator.

`engineInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`engineInit(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Initializes this key generator for a certain keysize, using the given source of randomness.

`engineInit(Key, AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Initializes the HMAC with the given secret key and algorithm parameters.

`engineInit(SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`
Initializes this key generator.

`engineInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`engineInit(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`
Initializes this key generator for a certain keysize, using the given source of randomness.

`engineInit(int, Key, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Initializes this cipher with a key and a source of randomness.

`engineInit(int, Key, AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

`engineInit(int, Key, AlgorithmParameters, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

`engineInit(int, Key, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Initializes this cipher with a key and a source of randomness.

`engineInit(int, Key, AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.



engineInit(int, Key, AlgorithmParameters, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.RSASSL

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

engineInit(int, Key, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.TDCNP

Initializes this cipher with a key and a source of randomness.

engineInit(int, Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.TDCNP

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

engineInit(int, Key, AlgorithmParameters, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.TDCNP

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.DatawithDSA

Initialize the receiver with the specified private key, to be used for signing purposes.

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.DatawithRSA

Initialize the RSA object with a RSA private key.

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.SHA1withDSA

Initialize the receiver with the specified private key, to be used for signing purposes.

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.SHA1withRSA

Initialize the RSA object with a RSA private key.

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.SHA2withRSA

Initialize the RSA object with a RSA private key.

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.SHA3withRSA

Initialize the RSA object with a RSA private key.

engineInitSign(PrivateKey) - Method in class
com.ibm.crypto.fips.provider.SHA5withRSA

Initialize the RSA object with a RSA private key.

engineInitVerify(PublicKey) - Method in class
com.ibm.crypto.fips.provider.DatawithDSA

Initialize the receiver with the specified public key, to be used for verification purposes.

engineInitVerify(PublicKey) - Method in class
com.ibm.crypto.fips.provider.DatawithRSA

Initialize the RSA object with a RSA public key.

engineInitVerify(PublicKey) - Method in class
com.ibm.crypto.fips.provider.SHA1withDSA



Initialize the receiver with the specified public key, to be used for verification purposes.

`engineInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Initialize the RSA object with a RSA public key.

`engineInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Initialize the RSA object with a RSA public key.

`engineInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Initialize the RSA object with a RSA public key.

`engineInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA5withRSA`
Initialize the RSA object with a RSA public key.

`engineNextBytes(byte[])` - Method in class `com.ibm.crypto.fips.provider.SecureRandom`

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.SHA`

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.SHA2`

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.SHA3`

`engineReset()` - Method in class `com.ibm.crypto.fips.provider.SHA5`

`engineSetMode(String)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Sets the mode of this cipher.

`engineSetMode(String)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Sets the mode of this cipher.

`engineSetMode(String)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Sets the mode of this cipher.

`engineSetMode(String)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Sets the mode of this cipher.

`engineSetMode(String)` - Method in class `com.ibm.crypto.fips.provider.TDCNP`
Sets the mode of this cipher.



`engineSetPadding(String)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Sets the padding mechanism of this cipher.

`engineSetPadding(String)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Sets the padding mechanism of this cipher.

`engineSetPadding(String)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Sets the padding mechanism of this cipher.

`engineSetPadding(String)` - Method in class `com.ibm.crypto.fips.provider.RSASSL`
Sets the padding mechanism of this cipher.

`engineSetPadding(String)` - Method in class `com.ibm.crypto.fips.provider.TDCNP`
Sets the padding mechanism of this cipher.

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Deprecated. Replaced with `engineSetParameter(AlgorithmParameterSpec)`

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`
Have to implement Signature's abstract method `engineSetParameter` to be a concrete class.

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Deprecated. Replaced with `engineSetParameter(AlgorithmParameterSpec)`

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Have to implement Signature's abstract method `engineSetParameter` to be a concrete class.

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Have to implement Signature's abstract method `engineSetParameter` to be a concrete class.

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Have to implement Signature's abstract method `engineSetParameter` to be a concrete class.

`engineSetParameter(String, Object)` - Method in class `com.ibm.crypto.fips.provider.SHA5withRSA`
Have to implement Signature's abstract method `engineSetParameter` to be a concrete class.

`engineSetSeed(byte[])` - Method in class `com.ibm.crypto.fips.provider.SecureRandom`

`engineSign()` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Answers the signature bytes of the data updated so far.

`engineSign()` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`
Get message digest for all the data thus far updated, then sign the message digest.

`engineSign()` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Answers the signature bytes of the data updated so far.

`engineSign()` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Get message digest for all the data thus far updated, then sign the message digest.



engineSign() - Method in class com.ibm.crypto.fips.provider.SHA2withRSA
Get message digest for all the data thus far updated, then sign the message digest.

engineSign() - Method in class com.ibm.crypto.fips.provider.SHA3withRSA
Get message digest for all the data thus far updated, then sign the message digest.

engineSign() - Method in class com.ibm.crypto.fips.provider.SHA5withRSA
Get message digest for all the data thus far updated, then sign the message digest.

engineToString() - Method in class com.ibm.crypto.fips.provider.AESParameters

engineToString() - Method in class com.ibm.crypto.fips.provider.DESedeParameters

engineToString() - Method in class com.ibm.crypto.fips.provider.DHParameters

engineToString() - Method in class com.ibm.crypto.fips.provider.DSAParameters

engineTranslateKey(SecretKey) - Method in class com.ibm.crypto.fips.provider.AESKeyFactory
This action is not allowed in this provider.

engineTranslateKey(SecretKey) - Method in class com.ibm.crypto.fips.provider.DESedeKeyFactory
This action is not allowed in this provider.

engineTranslateKey(Key) - Method in class com.ibm.crypto.fips.provider.DHKeyFactory
This action is not allowed in this provider.

engineTranslateKey(Key) - Method in class com.ibm.crypto.fips.provider.DSAKeyFactory
This action is not allowed in this provider.

engineTranslateKey(Key) - Method in class com.ibm.crypto.fips.provider.RSAKeyFactory
This action is not allowed in this provider.

engineUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.AESCipher
Unwrap a previously wrapped key.

engineUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.CipherWithWrappingSpi
Unwrap a previously wrapped key.

engineUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.DESedeCipher
Unwrap a previously wrapped key.

engineUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.RSA
Unwrap a previously wrapped key.

engineUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.RSASSL
Unwrap a previously wrapped key.

engineUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.TDCNP
Unwrap a previously wrapped key.

engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.AESCipher



Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte[], int, int, byte[], int) - Method in class com.ibm.crypto.fips.provider.AESCipher

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.DatawithDSA
Update the bytes signed so far with the extra byte provided.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.DatawithDSA
Update the bytes signed so far with the extra bytes provided.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.DatawithRSA
Update a byte to be signed or verified.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.DatawithRSA
Update an array of bytes to be signed or verified.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte[], int, int, byte[], int) - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.HmacSHA1
Processes the given byte.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.HmacSHA1
Processes the first len bytes in input, starting at offset.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.HmacSHA256
Processes the given byte.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.HmacSHA256
Processes the first len bytes in input, starting at offset.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.HmacSHA384
Processes the given byte.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.HmacSHA384
Processes the first len bytes in input, starting at offset.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.HmacSHA512
Processes the given byte.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.HmacSHA512
Processes the first len bytes in input, starting at offset.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.RSA



Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte[], int, int, byte[], int) - Method in class com.ibm.crypto.fips.provider.RSA

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.RSASSL

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte[], int, int, byte[], int) - Method in class com.ibm.crypto.fips.provider.RSASSL

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA
Update adds the passed byte to the digested data.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA
Update adds the selected part of an array of bytes to the digest.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA1withDSA
Update the bytes signed so far with the extra byte provided.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA1withDSA
Update the bytes signed so far with the extra bytes provided.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA1withRSA
Update a byte to be signed or verified.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA1withRSA
Update an array of bytes to be signed or verified.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA2
Update adds the passed byte to the digested data.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA2
Update adds the selected part of an array of bytes to the digest.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA2withRSA
Update a byte to be signed or verified.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA2withRSA
Update an array of bytes to be signed or verified.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA3
Update adds the passed byte to the digested data.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA3
Update adds the selected part of an array of bytes to the digest.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA3withRSA
Update a byte to be signed or verified.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA3withRSA
Update an array of bytes to be signed or verified.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA5



Update adds the passed byte to the digested data.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA5
Update adds the selected part of an array of bytes to the digest.
engineUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA5withRSA
Update a byte to be signed or verified.
engineUpdate(byte[], int, int) - Method in class
com.ibm.crypto.fips.provider.SHA5withRSA
Update an array of bytes to be signed or verified.
engineUpdate(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.TDCNP
Continues a multiple-part encryption or decryption operation (depending on how
this cipher was initialized), processing another data part.
engineUpdate(byte[], int, int, byte[], int) - Method in class
com.ibm.crypto.fips.provider.TDCNP
Continues a multiple-part encryption or decryption operation (depending on how
this cipher was initialized), processing another data part.
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.DatawithDSA
Verifies the passed signature.
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.DatawithRSA
Verify the signature (compare the result with the message digest).
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA1withDSA
Verifies the passed signature.
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA1withRSA
Verify the signature (compare the result with the message digest).
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA2withRSA
Verify the signature (compare the result with the message digest).
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA3withRSA
Verify the signature (compare the result with the message digest).
engineVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA5withRSA
Verify the signature (compare the result with the message digest).
engineWrap(Key) - Method in class com.ibm.crypto.fips.provider.AESCipher
Wrap a key.
engineWrap(Key) - Method in class
com.ibm.crypto.fips.provider.CipherWithWrappingSpi
Wrap a key.
engineWrap(Key) - Method in class com.ibm.crypto.fips.provider.DESedeCipher
Wrap a key.
engineWrap(Key) - Method in class com.ibm.crypto.fips.provider.RSA
Wrap a key.
engineWrap(Key) - Method in class com.ibm.crypto.fips.provider.RSASSL
Wrap a key.
engineWrap(Key) - Method in class com.ibm.crypto.fips.provider.TDCNP
Wrap a key.
equals(Object) - Method in class com.ibm.crypto.fips.provider.DESedeKey
equals(Object) - Method in class com.ibm.crypto.fips.provider.DHPrivateKey



equals(Object) - Method in class com.ibm.crypto.fips.provider.DHPublicKey

F

FeedbackCipher - Interface in com.ibm.crypto.fips.provider

This interface represents the type of cipher that has a feedback mechanism built into it, such as CBC or CFB.

finalize() - Method in class com.ibm.crypto.fips.provider.AESSecretKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.DESedeKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.DHPublicKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.DSAPrivateKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.DSAPublicKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.HmacSHA1

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.HmacSHA256

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.HmacSHA384

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.HmacSHA512

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.RSAPrivateKey

This function zeroizes the key so that it isn't in memory when GC is done.

finalize() - Method in class com.ibm.crypto.fips.provider.RSAPublicKey

This function zeroizes the key so that it isn't in memory when GC is done.

FIPSRuntimeException - Exception in com.ibm.crypto.fips.provider

FIPSRuntimeException() - Constructor for exception

com.ibm.crypto.fips.provider.FIPSRuntimeException

Constructs a FIPSRuntimeException with no detail message.

FIPSRuntimeException(String) - Constructor for exception

com.ibm.crypto.fips.provider.FIPSRuntimeException

Constructs a FIPSRuntimeException with the specified detail message.

fromFile(File) - Static method in class com.ibm.crypto.fips.provider.IdentityDatabase

Initialize an IdentityDatabase from file.

fromStream(InputStream) - Static method in class

com.ibm.crypto.fips.provider.IdentityDatabase



Initialize an identity database from a stream.

G

g - Variable in class com.ibm.crypto.fips.provider.DHParameters

g - Variable in class com.ibm.crypto.fips.provider.DSAParameters

generateKeyPair() - Method in class com.ibm.crypto.fips.provider.DHKeyPairGenerator
Generates a key pair.

generateKeyPair() - Method in class
com.ibm.crypto.fips.provider.DSAKeyPairGenerator
Answers a newly generated key pair.

generateKeyPair() - Method in class com.ibm.crypto.fips.provider.RSAKeyPairGenerator

getAlgorithm() - Method in class com.ibm.crypto.fips.provider.AESSecretKey

getAlgorithm() - Method in class com.ibm.crypto.fips.provider.DESedeKey

getAlgorithm() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
Returns the name of the algorithm associated with this key: "DH"

getAlgorithm() - Method in class com.ibm.crypto.fips.provider.DHPublicKey
Returns the name of the algorithm associated with this key: "DH"

getCrtCoefficient() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Returns the crtCoefficient.

getEncoded() - Method in class com.ibm.crypto.fips.provider.AESSecretKey

getEncoded() - Method in class com.ibm.crypto.fips.provider.DESedeKey

getEncoded() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
Get the encoding of the key.

getEncoded() - Method in class com.ibm.crypto.fips.provider.DHPublicKey
Get the encoding of the key.

getFeedback() - Method in interface com.ibm.crypto.fips.provider.FeedbackCipher
Gets the name of the feedback mechanism

getFipsLevel() - Method in class com.ibm.crypto.fips.provider.IBMJCEFIPS
Method returns the cryptographic modules FIPS 140-2 certification level

getFipsLevel() - Method in interface com.ibm.crypto.fips.provider.ModuleStatus
Method returns the cryptographic modules FIPS 140-2 certification level

getFormat() - Method in class com.ibm.crypto.fips.provider.AESSecretKey

getFormat() - Method in class com.ibm.crypto.fips.provider.DESedeKey

getFormat() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
Returns the encoding format of this key: "PKCS#8"

getFormat() - Method in class com.ibm.crypto.fips.provider.DHPublicKey



Returns the encoding format of this key: "X.509"
getIdentity(String) - Method in class com.ibm.crypto.fips.provider.IdentityDatabase

getIdentity(PublicKey) - Method in class com.ibm.crypto.fips.provider.IdentityDatabase
Get an identity by key.

getIV() - Method in interface com.ibm.crypto.fips.provider.FeedbackCipher
Gets the initialization vector.

getKey() - Method in class com.ibm.crypto.fips.provider.AESKeySpec
Returns the AES key material.

getModulus() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Return the modulus.

getModulus() - Method in class com.ibm.crypto.fips.provider.RSAPrivateKey
Return the modulus.

getModulus() - Method in class com.ibm.crypto.fips.provider.RSAPublicKey
Return the modulus.

getParams() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
Returns the key parameters.

getParams() - Method in class com.ibm.crypto.fips.provider.DHPublicKey
Returns the key parameters.

getParams() - Method in class com.ibm.crypto.fips.provider.DSAPrivateKey
Returns the DSA parameters associated with this key, or null if the parameters could not be parsed.

getParams() - Method in class com.ibm.crypto.fips.provider.DSAPublicKey
Return the DSA parameters for the receiver.

getPrimeExponentP() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Returns the primeExponentP.

getPrimeExponentQ() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Returns the primeExponentQ.

getPrimeP() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Returns the primeP.

getPrimeQ() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Returns the primeQ.

getPrivateExponent() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Return the private exponent.

getPrivateExponent() - Method in class com.ibm.crypto.fips.provider.RSAPrivateKey
Return the private exponent.

getPublicExponent() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
Returns the public exponent.

getPublicExponent() - Method in class com.ibm.crypto.fips.provider.RSAPublicKey
Return the public exponent.

getSelfTest() - Method in class com.ibm.crypto.fips.provider.IBMJCEFIPS
Method returns a SelfTest object that can be used to

getSelfTest() - Method in interface com.ibm.crypto.fips.provider.ModuleStatus
Method returns a SelfTest object that can be used to

getSelfTestFailure() - Method in class com.ibm.crypto.fips.provider.SelfTest
Method identifies any failures associated with the last self test



getX() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
Returns the private value, x.

getX() - Method in class com.ibm.crypto.fips.provider.DSAPrivateKey
Return the value of the private key.

getY() - Method in class com.ibm.crypto.fips.provider.DHPublicKey
Returns the public value, y.

getY() - Method in class com.ibm.crypto.fips.provider.DSAPublicKey
Return the value of the public key.

H

hashCode() - Method in class com.ibm.crypto.fips.provider.DESedeKey
Calculates a hash code value for the object.

hashCode() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
Calculates a hash code value for the object.

hashCode() - Method in class com.ibm.crypto.fips.provider.DHPublicKey
Calculates a hash code value for the object.

HmacSHA1 - Class in com.ibm.crypto.fips.provider
This is an implementation of the HMAC-SHA1 algorithm.

HmacSHA1() - Constructor for class com.ibm.crypto.fips.provider.HmacSHA1
Standard constructor, creates a new HmacSHA1 instance.

HmacSHA1KeyGenerator - Class in com.ibm.crypto.fips.provider
This class generates a secret key for use with the HMAC-SHA1 algorithm.

HmacSHA1KeyGenerator() - Constructor for class
com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator
Verify the JCE framework in the constructor.

HmacSHA256 - Class in com.ibm.crypto.fips.provider
This is an implementation of the HMAC-SHA256 algorithm.

HmacSHA256() - Constructor for class com.ibm.crypto.fips.provider.HmacSHA256
Standard constructor, creates a new HmacSHA256 instance.

HmacSHA256KeyGenerator - Class in com.ibm.crypto.fips.provider
This class generates a secret key for use with the HMAC-SHA256 algorithm.

HmacSHA256KeyGenerator() - Constructor for class
com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator
Verify the JCE framework in the constructor.

HmacSHA384 - Class in com.ibm.crypto.fips.provider
This is an implementation of the HMAC-SHA384 algorithm.

HmacSHA384() - Constructor for class com.ibm.crypto.fips.provider.HmacSHA384
Standard constructor, creates a new HmacSHA384 instance.

HmacSHA384KeyGenerator - Class in com.ibm.crypto.fips.provider
This class generates a secret key for use with the HMAC-SHA384 algorithm.

HmacSHA384KeyGenerator() - Constructor for class
com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator
Verify the JCE framework in the constructor.

HmacSHA512 - Class in com.ibm.crypto.fips.provider
This is an implementation of the HMAC-SHA512 algorithm.



HmacSHA512() - Constructor for class com.ibm.crypto.fips.provider.HmacSHA512
Standard constructor, creates a new HmacSHA512 instance.

HmacSHA512KeyGenerator - Class in com.ibm.crypto.fips.provider
This class generates a secret key for use with the HMAC-SHA512 algorithm.

HmacSHA512KeyGenerator() - Constructor for class
com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator
Verify the JCE framework in the constructor.

I

IBMJCEFIPS - Class in com.ibm.crypto.fips.provider
Defines the "IBMJCEFIPS" provider.

IBMJCEFIPS() - Constructor for class com.ibm.crypto.fips.provider.IBMJCEFIPS

identities() - Method in class com.ibm.crypto.fips.provider.IdentityDatabase

IdentityDatabase - Class in com.ibm.crypto.fips.provider
An implementation of IdentityScope as a persistent identity database.

IdentityDatabase(File) - Constructor for class
com.ibm.crypto.fips.provider.IdentityDatabase
Construct a new, empty database with a specified source file.

IdentityDatabase(String) - Constructor for class
com.ibm.crypto.fips.provider.IdentityDatabase
Construct a new, empty database.

init() - Method in class com.ibm.crypto.fips.provider.SHA
Initialize the SHA information

init() - Method in class com.ibm.crypto.fips.provider.SHA2
Initialize the SHA2 information

init() - Method in class com.ibm.crypto.fips.provider.SHA3
Initialize the SHA3 information

init() - Method in class com.ibm.crypto.fips.provider.SHA5
Initialize the SHA5 information

initialize(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHKeyPairGenerator
Initializes this key pair generator for a certain keysize and source of randomness.

initialize(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHKeyPairGenerator
Initializes this key pair generator for the specified parameter set and source of
randomness.

initialize(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DSAKeyPairGenerator
Initialize the receiver to use a given secure random generator, and generate keys
from the provided set of parameters.

initialize(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DSAKeyPairGenerator



Initialize the receiver to use a given secure random generator, and generate keys of a certain size.

initialize(AlgorithmParameterSpec, SecureRandom) - Method in class com.ibm.crypto.fips.provider.RSAKeyPairGenerator

initialize(int, SecureRandom) - Method in class com.ibm.crypto.fips.provider.RSAKeyPairGenerator

Initializes this KeyPairGenerator for given modulus and random source
initialize(int) - Method in class com.ibm.crypto.fips.provider.RSAKeyPairGenerator

internalClone() - Method in class com.ibm.crypto.fips.provider.HmacSHA1

internalClone() - Method in class com.ibm.crypto.fips.provider.HmacSHA256

internalClone() - Method in class com.ibm.crypto.fips.provider.HmacSHA384

internalClone() - Method in class com.ibm.crypto.fips.provider.HmacSHA512

internalClone() - Method in class com.ibm.crypto.fips.provider.SHA
Clones this object.

internalClone() - Method in class com.ibm.crypto.fips.provider.SHA2
Clones this object.

internalClone() - Method in class com.ibm.crypto.fips.provider.SHA3
Clones this object.

internalClone() - Method in class com.ibm.crypto.fips.provider.SHA5
Clones this object.

internalDigest() - Method in class com.ibm.crypto.fips.provider.SHA

internalDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA

internalDigest(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA

internalDigest() - Method in class com.ibm.crypto.fips.provider.SHA2
Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

internalDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA2
Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

internalDigest() - Method in class com.ibm.crypto.fips.provider.SHA3
Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

internalDigest(byte[], int, int) - Method in class com.ibm.crypto.fips.provider.SHA3
Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

internalDigest() - Method in class com.ibm.crypto.fips.provider.SHA5



Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

`internalDigest(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA5`
Compute the final hash and reset the engine so that it is ready for re-use, as specified by MessageDigest.

`internalDoFinal(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`internalDoFinal(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`internalDoFinal(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`internalDoFinal(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`internalDoFinal()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`internalDoFinal()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`
Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`internalDoFinal()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`internalDoFinal()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Completes the HMAC computation and resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

`internalDoFinal(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`internalDoFinal(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.

`internalDoPhase(Key, boolean)` - Method in class `com.ibm.crypto.fips.provider.DHKeyAgreement`
Executes the next phase of this key agreement with the given key that was received from one of the other parties involved in this key agreement.

`internalGenerateKey()` - Method in class `com.ibm.crypto.fips.provider.AESKeyGenerator`
Generates a AES key.



internalGenerateKey() - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator

Generates the Triple-DES key.

internalGenerateKey() - Method in class
com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator

Generates an HMAC-SHA1 key.

internalGenerateKey() - Method in class
com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator

Generates an HMAC-SHA256 key.

internalGenerateKey() - Method in class
com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator

Generates an HMAC-SHA384 key.

internalGenerateKey() - Method in class
com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator

Generates an HMAC-SHA512 key.

internalGenerateKeyPair() - Method in class
com.ibm.crypto.fips.provider.DHKeyPairGenerator

Generates a key pair.

internalGenerateKeyPair() - Method in class
com.ibm.crypto.fips.provider.DSAKeyPairGenerator

internalGenerateKeyPair() - Method in class
com.ibm.crypto.fips.provider.RSAKeyPairGenerator

internalGenerateParameters() - Method in class
com.ibm.crypto.fips.provider.DHParameterGenerator

Generates the parameters.

internalGenerateParameters() - Method in class
com.ibm.crypto.fips.provider.DSAParameterGenerator

Answers the newly generated parameters.

internalGeneratePrivate(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DHKeyFactory

Generates a private key object from the provided key specification (key material).

internalGeneratePrivate(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DSAKeyFactory

Generates a private key for the given key specification.

internalGeneratePrivate(KeySpec) - Method in class
com.ibm.crypto.fips.provider.RSAKeyFactory

Generates a private key object from the provided key specification (key material).

internalGeneratePublic(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DHKeyFactory

Generates a public key object from the provided key specification (key material).

internalGeneratePublic(KeySpec) - Method in class
com.ibm.crypto.fips.provider.DSAKeyFactory

Generates a public key for the given key specification.



`internalGeneratePublic(KeySpec)` - Method in class `com.ibm.crypto.fips.provider.RSAKeyFactory`
Generates a public key object from the provided key specification (key material).

`internalGenerateSecret(KeySpec)` - Method in class `com.ibm.crypto.fips.provider.AESKeyFactory`
Generates a `SecretKey` object from the provided key specification (key material).

`internalGenerateSecret(KeySpec)` - Method in class `com.ibm.crypto.fips.provider.DESedeKeyFactory`
Generates a `SecretKey` object from the provided key specification (key material).

`internalGenerateSecret()` - Method in class `com.ibm.crypto.fips.provider.DHKeyAgreement`
Generates the shared secret and returns it in a new buffer.

`internalGenerateSecret(byte[], int)` - Method in class `com.ibm.crypto.fips.provider.DHKeyAgreement`
Generates the shared secret, and places it into the buffer `sharedSecret`, beginning at `offset`.

`internalGenerateSecret(String)` - Method in class `com.ibm.crypto.fips.provider.DHKeyAgreement`
Creates the shared secret and returns it as a secret key object of the requested algorithm type.

`internalGenerateSeed(int)` - Method in class `com.ibm.crypto.fips.provider.SecureRandom`

`internalGetBlockSize()` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Returns the block size (in bytes).

`internalGetBlockSize()` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Returns the block size (in bytes).

`internalGetBlockSize()` - Method in class `com.ibm.crypto.fips.provider.RSA`
Returns the block size (in bytes).

`internalGetDigestLength()` - Method in class `com.ibm.crypto.fips.provider.SHA`
Return the digest length in bytes

`internalGetDigestLength()` - Method in class `com.ibm.crypto.fips.provider.SHA2`
Return the digest length in bytes

`internalGetDigestLength()` - Method in class `com.ibm.crypto.fips.provider.SHA3`
Return the digest length in bytes

`internalGetDigestLength()` - Method in class `com.ibm.crypto.fips.provider.SHA5`
Return the digest length in bytes

`internalGetEncoded()` - Method in class `com.ibm.crypto.fips.provider.AESParameters`

`internalGetEncoded(String)` - Method in class `com.ibm.crypto.fips.provider.AESParameters`

`internalGetEncoded()` - Method in class `com.ibm.crypto.fips.provider.DESedeParameters`

`internalGetEncoded(String)` - Method in class `com.ibm.crypto.fips.provider.DESedeParameters`



internalGetEncoded() - Method in class com.ibm.crypto.fips.provider.DHParameters

internalGetEncoded() - Method in class com.ibm.crypto.fips.provider.DSAParameters

Returns the parameters in encoded bytes.

internalGetEncoded(String) - Method in class

com.ibm.crypto.fips.provider.DSAParameters

Returns the parameters in encoded bytes with encoding method specified.

internalGetIV() - Method in class com.ibm.crypto.fips.provider.AESCipher

Returns the initialization vector (IV) in a new buffer.

internalGetIV() - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Returns the initialization vector (IV) in a new buffer.

internalGetKey() - Method in class com.ibm.crypto.fips.provider.AESKeySpec

internalGetKeySize(Key) - Method in class com.ibm.crypto.fips.provider.AESCipher

Returns the key size of the given key object.

internalGetKeySize(Key) - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Returns the key size of the given key object.

internalGetKeySize(Key) - Method in class com.ibm.crypto.fips.provider.RSA

Returns the key size of the given key object.

internalGetKeySpec(SecretKey, Class) - Method in class

com.ibm.crypto.fips.provider.AESKeyFactory

Returns a specification (key material) of the given key in the requested format.

internalGetKeySpec(SecretKey, Class) - Method in class

com.ibm.crypto.fips.provider.DESedeKeyFactory

Returns a specification (key material) of the given key in the requested format.

internalGetKeySpec(Key, Class) - Method in class

com.ibm.crypto.fips.provider.DHKeyFactory

Returns a specification (key material) of the given key object in the requested format.

internalGetKeySpec(Key, Class) - Method in class

com.ibm.crypto.fips.provider.DSAKeyFactory

Answers a key specification for a given key.

internalGetKeySpec(Key, Class) - Method in class

com.ibm.crypto.fips.provider.RSAKeyFactory

Returns a specification (key material) of the given key object in the requested format.

internalGetMacLength() - Method in class com.ibm.crypto.fips.provider.HmacSHA1

Returns the length of the HMAC in bytes.

internalGetMacLength() - Method in class com.ibm.crypto.fips.provider.HmacSHA256

Returns the length of the HMAC in bytes.

internalGetMacLength() - Method in class com.ibm.crypto.fips.provider.HmacSHA384

Returns the length of the HMAC in bytes.

internalGetMacLength() - Method in class com.ibm.crypto.fips.provider.HmacSHA512

Returns the length of the HMAC in bytes.

internalGetOutputSize(int) - Method in class com.ibm.crypto.fips.provider.AESCipher



- Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).
- `internalGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).
- `internalGetOutputSize(int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Returns the length in bytes that an output buffer would need to be in order to hold the result of the next `update` or `doFinal` operation, given the input length `inputLen` (in bytes).
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Deprecated.
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Deprecated.
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `internalGetParameter(String)` - Method in class `com.ibm.crypto.fips.provider.SHA5withRSA`
Have to implement Signature's abstract method `engineGetParameter` to be a concrete class.
- `internalGetParameters()` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Returns the parameters used with this cipher.
- `internalGetParameters()` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Returns the parameters used with this cipher.
- `internalGetParameterSpec(Class)` - Method in class `com.ibm.crypto.fips.provider.AESParameters`



internalGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.DESedeParameters

internalGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.DHParameters

internalGetParameterSpec(Class) - Method in class
com.ibm.crypto.fips.provider.DSAParameters

Return the parameter spec used by this parameter instance.

internalInit(int, Key, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESCipher

Initializes this cipher with a key and a source of randomness.

internalInit(int, Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESCipher

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

internalInit(int, Key, AlgorithmParameters, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESCipher

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.

internalInit(SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESKeyGenerator

Initializes this key generator.

internalInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESKeyGenerator

Initializes this key generator with the specified parameter set and a user-provided source of randomness.

internalInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.AESKeyGenerator

Initializes this key generator for a certain keysize, using the given source of randomness.

internalInit(AlgorithmParameterSpec) - Method in class
com.ibm.crypto.fips.provider.AESParameters

internalInit(byte[]) - Method in class com.ibm.crypto.fips.provider.AESParameters

internalInit(byte[], String) - Method in class
com.ibm.crypto.fips.provider.AESParameters

internalInit(int, Key, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeCipher

Initializes this cipher with a key and a source of randomness.

internalInit(int, Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeCipher

Initializes this cipher with a key, a set of algorithm parameters, and a source of randomness.



internalInit(int, Key, AlgorithmParameters, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeCipher

internalInit(SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator
Initializes this key generator.

internalInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator
Initializes this key generator with the specified parameter set and a user-provided
source of randomness.

internalInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DESedeKeyGenerator
Initializes this key generator for a certain keysize, using the given source of
randomness.

internalInit(AlgorithmParameterSpec) - Method in class
com.ibm.crypto.fips.provider.DESedeParameters

internalInit(byte[]) - Method in class com.ibm.crypto.fips.provider.DESedeParameters

internalInit(byte[], String) - Method in class
com.ibm.crypto.fips.provider.DESedeParameters

internalInit(Key, AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHKeyAgreement
Initializes this key agreement with the given key, set of algorithm parameters, and
source of randomness.

internalInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHParameterGenerator
Initializes this parameter generator for a certain keysize and source of
randomness.

internalInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DHParameterGenerator
Initializes this parameter generator with a set of parameter generation values,
which specify the size of the prime modulus and the size of the random exponent,
both in bits.

internalInit(AlgorithmParameterSpec) - Method in class
com.ibm.crypto.fips.provider.DHParameters

internalInit(byte[]) - Method in class com.ibm.crypto.fips.provider.DHParameters

internalInit(AlgorithmParameterSpec, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DSAParameterGenerator
Initializes the receiver with the specified parameters and source of randomness.

internalInit(int, SecureRandom) - Method in class
com.ibm.crypto.fips.provider.DSAParameterGenerator



Initializes the receiver with the specified parameter size and source of randomness.

`internalInit(AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.DSAParameters`
Initialize the `DSAParameters` by a `DSAParameterSpec`

`internalInit(byte[])` - Method in class `com.ibm.crypto.fips.provider.DSAParameters`
Initialize the `DSAParameters` by encoded bytes

`internalInit(byte[], String)` - Method in class `com.ibm.crypto.fips.provider.DSAParameters`
Initialize the `DSAParameters` by encoded bytes with the specified decoding method.

`internalInit(Key, AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Initializes the HMAC with the given secret key and algorithm parameters.

`internalInit(SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`
Initializes this key generator.

`internalInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`internalInit(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator`
Initializes this key generator for a certain keysize, using the given source of randomness.

`internalInit(Key, AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`
Initializes the HMAC with the given secret key and algorithm parameters.

`internalInit(SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`
Initializes this key generator.

`internalInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`internalInit(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator`
Initializes this key generator for a certain keysize, using the given source of randomness.

`internalInit(Key, AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Initializes the HMAC with the given secret key and algorithm parameters.

`internalInit(SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Initializes this key generator.



`internalInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`internalInit(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator`
Initializes this key generator for a certain keysize, using the given source of randomness.

`internalInit(Key, AlgorithmParameterSpec)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Initializes the HMAC with the given secret key and algorithm parameters.

`internalInit(SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`
Initializes this key generator.

`internalInit(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`
Initializes this key generator with the specified parameter set and a user-provided source of randomness.

`internalInit(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator`
Initializes this key generator for a certain keysize, using the given source of randomness.

`internalInit(int, Key, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Initializes this cipher with a key and a source of randomness.

`internalInit()` - Method in class `com.ibm.crypto.fips.provider.SHA`
Initialize the SHA information

`internalInit()` - Method in class `com.ibm.crypto.fips.provider.SHA2`
Initialize the SHA2 information

`internalInit()` - Method in class `com.ibm.crypto.fips.provider.SHA3`
Initialize the SHA3 information

`internalInit()` - Method in class `com.ibm.crypto.fips.provider.SHA5`
Initialize the SHA5 information

`internalInitialize(int, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.DHKeyPairGenerator`
Initializes this key pair generator for a certain keysize and source of randomness.

`internalInitialize(AlgorithmParameterSpec, SecureRandom)` - Method in class `com.ibm.crypto.fips.provider.DHKeyPairGenerator`
Initializes this key pair generator for the specified parameter set and source of randomness.

`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Initialize the receiver with the specified private key, to be used for signing purposes.

`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`



Initialize the RSA object with a RSA private key.
`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Initialize the receiver with the specified private key, to be used for signing purposes.
`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Initialize the RSA object with a RSA private key.
`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Initialize the RSA object with a RSA private key.
`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Initialize the RSA object with a RSA private key.
`internalInitSign(PrivateKey)` - Method in class `com.ibm.crypto.fips.provider.SHA5withRSA`
Initialize the RSA object with a RSA private key.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Initialize the receiver with the specified public key, to be used for verification purposes.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`
Initialize the RSA object with a RSA public key.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Initialize the receiver with the specified public key, to be used for verification purposes.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Initialize the RSA object with a RSA public key.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Initialize the RSA object with a RSA public key.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Initialize the RSA object with a RSA public key.
`internalInitVerify(PublicKey)` - Method in class `com.ibm.crypto.fips.provider.SHA5withRSA`
Initialize the RSA object with a RSA public key.
`internalNextBytes(byte[])` - Method in class `com.ibm.crypto.fips.provider.SecureRandom`

`internalReset()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.
`internalReset()` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`



Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

internalReset() - Method in class com.ibm.crypto.fips.provider.HmacSHA384
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

internalReset() - Method in class com.ibm.crypto.fips.provider.HmacSHA512
Resets the HMAC for further use, maintaining the secret key that the HMAC was initialized with.

internalReset() - Method in class com.ibm.crypto.fips.provider.SHA

internalReset() - Method in class com.ibm.crypto.fips.provider.SHA2

internalReset() - Method in class com.ibm.crypto.fips.provider.SHA3

internalReset() - Method in class com.ibm.crypto.fips.provider.SHA5

internalSetMode(String) - Method in class com.ibm.crypto.fips.provider.AESCipher
Sets the mode of this cipher.

internalSetMode(String) - Method in class com.ibm.crypto.fips.provider.DESedeCipher
Sets the mode of this cipher.

internalSetMode(String) - Method in class com.ibm.crypto.fips.provider.RSA
Sets the mode of this cipher.

internalSetPadding(String) - Method in class com.ibm.crypto.fips.provider.AESCipher
Sets the padding mechanism of this cipher.

internalSetPadding(String) - Method in class com.ibm.crypto.fips.provider.DESedeCipher
Sets the padding mechanism of this cipher.

internalSetPadding(String) - Method in class com.ibm.crypto.fips.provider.RSA
Sets the padding mechanism of this cipher.

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.DatawithDSA
Deprecated. Replaced with *engineSetParameter(AlgorithmParameterSpec)*

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.DatawithRSA
Have to implement Signature's abstract method *engineSetParameter* to be a concrete class.

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.SHA1withDSA
Deprecated. Replaced with *engineSetParameter(AlgorithmParameterSpec)*

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.SHA1withRSA
Have to implement Signature's abstract method *engineSetParameter* to be a concrete class.

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.SHA2withRSA



Have to implement Signature's abstract method engineSetParameter to be a concrete class.

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.SHA3withRSA
Have to implement Signature's abstract method engineSetParameter to be a concrete class.

internalSetParameter(String, Object) - Method in class com.ibm.crypto.fips.provider.SHA5withRSA
Have to implement Signature's abstract method engineSetParameter to be a concrete class.

internalSetSeed(byte[]) - Method in class com.ibm.crypto.fips.provider.SecureRandom

internalSign() - Method in class com.ibm.crypto.fips.provider.DatawithDSA
Answers the signature bytes of the data updated so far.

internalSign() - Method in class com.ibm.crypto.fips.provider.DatawithRSA
Get message digest for all the data thus far updated, then sign the message digest.

internalSign() - Method in class com.ibm.crypto.fips.provider.SHA1withDSA
Answers the signature bytes of the data updated so far.

internalSign() - Method in class com.ibm.crypto.fips.provider.SHA1withRSA
Get message digest for all the data thus far updated, then sign the message digest.

internalSign() - Method in class com.ibm.crypto.fips.provider.SHA2withRSA
Get message digest for all the data thus far updated, then sign the message digest.

internalSign() - Method in class com.ibm.crypto.fips.provider.SHA3withRSA
Get message digest for all the data thus far updated, then sign the message digest.

internalSign() - Method in class com.ibm.crypto.fips.provider.SHA5withRSA
Get message digest for all the data thus far updated, then sign the message digest.

internalToString() - Method in class com.ibm.crypto.fips.provider.AESParameters

internalToString() - Method in class com.ibm.crypto.fips.provider.DatawithDSA
Answers a string containing a concise, human-readable description of the receiver.

internalToString() - Method in class com.ibm.crypto.fips.provider.DESedeParameters

internalToString() - Method in class com.ibm.crypto.fips.provider.DHParameters

internalToString() - Method in class com.ibm.crypto.fips.provider.DSAParameters

internalToString() - Method in class com.ibm.crypto.fips.provider.SHA1withDSA
Answers a string containing a concise, human-readable description of the receiver.

internalUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.AESCipher
Unwrap a previously wrapped key.

internalUnwrap(byte[], String, int) - Method in class com.ibm.crypto.fips.provider.CipherWithWrappingSpi
Unwrap a previously wrapped key.



`internalUnwrap(byte[], String, int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Unwrap a previously wrapped key.

`internalUnwrap(byte[], String, int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Unwrap a previously wrapped key.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

`internalUpdate(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.AESCipher`
Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Update the bytes signed so far with the extra byte provided.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.DatawithDSA`
Update the bytes signed so far with the extra bytes provided.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.DatawithRSA`
Update a byte to be signed or verified.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

`internalUpdate(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`
Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Processes the given byte.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA1`
Processes the first `len` bytes in `input`, starting at `offset`.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`
Processes the given byte.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA256`
Processes the first `len` bytes in `input`, starting at `offset`.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Processes the given byte.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA384`
Processes the first `len` bytes in `input`, starting at `offset`.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Processes the given byte.



`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.HmacSHA512`
Processes the first `len` bytes in `input`, starting at `offset`.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

`internalUpdate(byte[], int, int, byte[], int)` - Method in class `com.ibm.crypto.fips.provider.RSA`
Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA`
Update adds the passed byte to the digested data.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA`
Update adds the selected part of an array of bytes to the digest.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Update the bytes signed so far with the extra byte provided.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA1withDSA`
Update the bytes signed so far with the extra bytes provided.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Update a byte to be signed or verified.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA1withRSA`
Update an array of bytes to be signed or verified.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA2`
Update adds the passed byte to the digested data.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA2`
Update adds the selected part of an array of bytes to the digest.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Update a byte to be signed or verified.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA2withRSA`
Update an array of bytes to be signed or verified.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA3`
Update adds the passed byte to the digested data.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA3`
Update adds the selected part of an array of bytes to the digest.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Update a byte to be signed or verified.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA3withRSA`
Update an array of bytes to be signed or verified.

`internalUpdate(byte)` - Method in class `com.ibm.crypto.fips.provider.SHA5`
Update adds the passed byte to the digested data.

`internalUpdate(byte[], int, int)` - Method in class `com.ibm.crypto.fips.provider.SHA5`
Update adds the selected part of an array of bytes to the digest.



internalUpdate(byte) - Method in class com.ibm.crypto.fips.provider.SHA5withRSA

Update a byte to be signed or verified.

internalUpdate(byte[], int, int) - Method in class

com.ibm.crypto.fips.provider.SHA5withRSA

Update an array of bytes to be signed or verified.

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.DatawithDSA

Verifies the passed signature.

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.DatawithRSA

Verify the signature (compare the result with the message digest).

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA1withDSA

Verifies the passed signature.

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA1withRSA

Verify the signature (compare the result with the message digest).

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA2withRSA

Verify the signature (compare the result with the message digest).

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA3withRSA

Verify the signature (compare the result with the message digest).

internalVerify(byte[]) - Method in class com.ibm.crypto.fips.provider.SHA5withRSA

Verify the signature (compare the result with the message digest).

internalWrap(Key) - Method in class com.ibm.crypto.fips.provider.AESCipher

Wrap a key.

internalWrap(Key) - Method in class

com.ibm.crypto.fips.provider.CipherWithWrappingSpi

Wrap a key.

internalWrap(Key) - Method in class com.ibm.crypto.fips.provider.DESedeCipher

Wrap a key.

internalWrap(Key) - Method in class com.ibm.crypto.fips.provider.RSA

Wrap a key.

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.AESCipher

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.AESKeyFactory

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.AESKeyGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.AESKeySpec

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.AESParameters

isFipsApproved() - Method in interface com.ibm.crypto.fips.provider.AlgorithmStatus

Module identifies if the cryptographic operation (algorithm) is FIPS certified

isFipsApproved() - Method in class

com.ibm.crypto.fips.provider.CipherWithWrappingSpi

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DatawithDSA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DatawithRSA



isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DESedeCipher

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DESedeKeyFactory

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DESedeKeyGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DESedeParameters

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DHKeyAgreement
This function allows an application to verify the the algorithm is FIPS approved.

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DHKeyFactory
This function allows an application to verify the the algorithm is FIPS approved.

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DHKeyPairGenerator
This function allows an application to verify the the algorithm is FIPS approved.

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DHParameterGenerator
This function allows an application to verify the the algorithm is FIPS approved.

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DHParameters

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DSAKeyFactory

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DSAKeyPairGenerator

isFipsApproved() - Method in class
com.ibm.crypto.fips.provider.DSAParameterGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.DSAParameters

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.HmacSHA1

isFipsApproved() - Method in class
com.ibm.crypto.fips.provider.HmacSHA1KeyGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.HmacSHA256

isFipsApproved() - Method in class
com.ibm.crypto.fips.provider.HmacSHA256KeyGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.HmacSHA384

isFipsApproved() - Method in class
com.ibm.crypto.fips.provider.HmacSHA384KeyGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.HmacSHA512



isFipsApproved() - Method in class
com.ibm.crypto.fips.provider.HmacSHA512KeyGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.RSA
This function allows an application to verify the the algorithm is FIPS approved.
isFipsApproved() - Method in class com.ibm.crypto.fips.provider.RSAKeyFactory

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.RSAKeyPairGenerator

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.RSASSL
This function allows an application to verify the the algorithm is FIPS approved.
isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SecureRandom

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA1withDSA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA1withRSA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA2

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA2withRSA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA3

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA3withRSA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA5

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SHA5withRSA

isFipsApproved() - Method in class com.ibm.crypto.fips.provider.SystemSigner

isFipsCertified() - Method in class com.ibm.crypto.fips.provider.IBMJCEFIPS

Method identifies if the cryptographic module is FIPS 140-2 certified

isFipsCertified() - Method in interface com.ibm.crypto.fips.provider.ModuleStatus

Method identifies if the cryptographic module is FIPS 140-2 certified

isFipsRunnable() - Static method in class com.ibm.crypto.fips.provider.SelfTest

Method identifies if the cryptographic module is FIPS 140-2 runnable, in that the self test has completed with no failures.

isSelfTestInProgress() - Method in class com.ibm.crypto.fips.provider.SelfTest

Method identifies if a self test is currently in progress

isTrusted() - Method in class com.ibm.crypto.fips.provider.SystemIdentity

Is this identity trusted by sun.* facilities?

isTrusted() - Method in class com.ibm.crypto.fips.provider.SystemSigner

Returns true if this signer is trusted.



M

ModuleStatus - Interface in com.ibm.crypto.fips.provider

O

OFB_MODE - Static variable in class com.ibm.crypto.fips.provider.DESedeCipher

P

p - Variable in class com.ibm.crypto.fips.provider.DHParameters

p - Variable in class com.ibm.crypto.fips.provider.DSAParameters

pad(byte[], int, int) - Method in interface com.ibm.crypto.fips.provider.Padding
Performs padding for the given data input.

Padding - Interface in com.ibm.crypto.fips.provider
Padding interface.

padLength(int) - Method in interface com.ibm.crypto.fips.provider.Padding
Determines how long the padding will be for a given input length.

padWithLen(byte[], int, int) - Method in interface com.ibm.crypto.fips.provider.Padding
Adds the given number of padding bytes to the data input.

parseKeyBits() - Method in class com.ibm.crypto.fips.provider.DSAPrivateKey

parseKeyBits() - Method in class com.ibm.crypto.fips.provider.DSAPublicKey

parseKeyBits() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey

parseKeyBits() - Method in class com.ibm.crypto.fips.provider.RSAPrivateKey

parseKeyBits() - Method in class com.ibm.crypto.fips.provider.RSAPublicKey

PCBC_MODE - Static variable in class com.ibm.crypto.fips.provider.DESedeCipher

propertyNames() - Method in class com.ibm.crypto.fips.provider.IBMJCEFIPS

Q

q - Variable in class com.ibm.crypto.fips.provider.DSAParameters



R

rawAlg - Variable in class com.ibm.crypto.fips.provider.DESedeCipher

removeIdentity(Identity) - Method in class
com.ibm.crypto.fips.provider.IdentityDatabase

Removes an identity to the database.

reset() - Method in interface com.ibm.crypto.fips.provider.FeedbackCipher
Resets the iv to its original value.

RSA - Class in com.ibm.crypto.fips.provider
This class implements the RSA algorithm.

RSA() - Constructor for class com.ibm.crypto.fips.provider.RSA
Creates an instance of RSA

RSA(boolean) - Constructor for class com.ibm.crypto.fips.provider.RSA
Creates an instance of RSA

RSAKeyFactory - Class in com.ibm.crypto.fips.provider
This class implements the RSA key factory of the IBMJCE/IBMJCA provider.

RSAKeyFactory() - Constructor for class com.ibm.crypto.fips.provider.RSAKeyFactory

RSAKeyPairGenerator - Class in com.ibm.crypto.fips.provider
This class generates RSA public/private key pairs.

RSAKeyPairGenerator() - Constructor for class
com.ibm.crypto.fips.provider.RSAKeyPairGenerator

RSAPrivateCrtKey - Class in com.ibm.crypto.fips.provider
An X.509 private crt key for the RSA Algorithm.

RSAPrivateCrtKey(BigInteger, BigInteger, BigInteger, BigInteger, BigInteger,
BigInteger, BigInteger, BigInteger) - Constructor for class
com.ibm.crypto.fips.provider.RSAPrivateCrtKey

This constructor computes missing key values and formats key values.

RSAPrivateCrtKey(byte[]) - Constructor for class
com.ibm.crypto.fips.provider.RSAPrivateCrtKey

Make a RSA private key from its DER encoding (PKCS #8).

RSAPrivateKey - Class in com.ibm.crypto.fips.provider
An X.509 private key for the RSA Algorithm.

RSAPrivateKey(BigInteger, BigInteger) - Constructor for class
com.ibm.crypto.fips.provider.RSAPrivateKey

Make a RSA private key.

RSAPrivateKey(byte[]) - Constructor for class
com.ibm.crypto.fips.provider.RSAPrivateKey

Make a RSA private key from its DER encoding (PKCS #8).

RSAPublicKey - Class in com.ibm.crypto.fips.provider
An X.509 public key for the RSA Algorithm.

RSAPublicKey(BigInteger, BigInteger) - Constructor for class
com.ibm.crypto.fips.provider.RSAPublicKey

Make a RSA public key.



`RSAPublicKey(byte[])` - Constructor for class `com.ibm.crypto.fips.provider.RSAPublicKey`
Make a RSA public key from its DER encoding (X.509).

`RSASSL` - Class in `com.ibm.crypto.fips.provider`
This class uses the RSA class with blinding turned on.

`RSASSL()` - Constructor for class `com.ibm.crypto.fips.provider.RSASSL`
Creates an instance of `RSASSL`

`runSelfTest()` - Method in class `com.ibm.crypto.fips.provider.SelfTest`
Method initiates a new self test

S

`save(OutputStream)` - Method in class `com.ibm.crypto.fips.provider.IdentityDatabase`
Save the database in its current state to an output stream.

`save()` - Method in class `com.ibm.crypto.fips.provider.IdentityDatabase`
Saves the database to the default source file.

`SecureRandom` - Class in `com.ibm.crypto.fips.provider`
This class provides a cryptographically strong pseudo-random number generator based on the SHA1 message digest algorithm.

`SecureRandom()` - Constructor for class `com.ibm.crypto.fips.provider.SecureRandom`

`SecureRandom(byte[])` - Constructor for class `com.ibm.crypto.fips.provider.SecureRandom`

`SelfTest` - Class in `com.ibm.crypto.fips.provider`

`SelfTest()` - Constructor for class `com.ibm.crypto.fips.provider.SelfTest`

`setRawAlg()` - Method in class `com.ibm.crypto.fips.provider.DESedeCipher`

`setTrusted(boolean)` - Method in class `com.ibm.crypto.fips.provider.SystemIdentity`
Set the trust status of this identity.

`SHA` - Class in `com.ibm.crypto.fips.provider`
This class implements the Secure Hash Algorithm (SHA) developed by the National Institute of Standards and Technology along with the National Security Agency.

`SHA()` - Constructor for class `com.ibm.crypto.fips.provider.SHA`
Standard constructor, creates a new SHA instance, allocates its buffers from the heap.

`SHA1withDSA` - Class in `com.ibm.crypto.fips.provider`

`SHA1withDSA()` - Constructor for class `com.ibm.crypto.fips.provider.SHA1withDSA`
Constructs a new instance of this class.

`SHA1withRSA` - Class in `com.ibm.crypto.fips.provider`
This class implements the SHA1withRSA

`SHA1withRSA()` - Constructor for class `com.ibm.crypto.fips.provider.SHA1withRSA`



- Construct a blank RSA object.
- SHA2** - Class in `com.ibm.crypto.fips.provider`
This class implements the Secure Hash Algorithm 2 (SHA2) developed by the National Institute of Standards and Technology along with the National Security Agency.
- SHA2()** - Constructor for class `com.ibm.crypto.fips.provider.SHA2`
Standard constructor, creates a new SHA2 instance, allocates its buffers from the heap.
- SHA2withRSA** - Class in `com.ibm.crypto.fips.provider`
This class implements the SHA1withRSA
- SHA2withRSA()** - Constructor for class `com.ibm.crypto.fips.provider.SHA2withRSA`
Construct a blank RSA object.
- SHA3** - Class in `com.ibm.crypto.fips.provider`
This class implements the Secure Hash Algorithm 3 (SHA-3) developed by the National Institute of Standards and Technology along with the National Security Agency.
- SHA3()** - Constructor for class `com.ibm.crypto.fips.provider.SHA3`
Standard constructor, creates a new SHA3 instance, allocates its buffers from the heap.
- SHA3withRSA** - Class in `com.ibm.crypto.fips.provider`
This class implements the SHA1withRSA
- SHA3withRSA()** - Constructor for class `com.ibm.crypto.fips.provider.SHA3withRSA`
Construct a blank RSA object.
- SHA5** - Class in `com.ibm.crypto.fips.provider`
This class implements the Secure Hash Algorithm 5 (SHA-5) developed by the National Institute of Standards and Technology along with the National Security Agency.
- SHA5()** - Constructor for class `com.ibm.crypto.fips.provider.SHA5`
Standard constructor, creates a new SHA5 instance, allocates its buffers from the heap.
- SHA5withRSA** - Class in `com.ibm.crypto.fips.provider`
This class implements the SHA1withRSA
- SHA5withRSA()** - Constructor for class `com.ibm.crypto.fips.provider.SHA5withRSA`
Construct a blank RSA object.
- size()** - Method in class `com.ibm.crypto.fips.provider.IdentityDatabase`
- SystemIdentity** - Class in `com.ibm.crypto.fips.provider`
An identity with a very simple trust mechanism.
- SystemIdentity(String, IdentityScope)** - Constructor for class `com.ibm.crypto.fips.provider.SystemIdentity`
- SystemSigner** - Class in `com.ibm.crypto.fips.provider`
SunSecurity signer.
- SystemSigner(String)** - Constructor for class `com.ibm.crypto.fips.provider.SystemSigner`
Construct a signer with a given name.



SystemSigner(String, IdentityScope) - Constructor for class
com.ibm.crypto.fips.provider.SystemSigner
Construct a signer with a name and a scope.

T

TDCNP - Class in com.ibm.crypto.fips.provider

This class creates a DESede cipher with default mode CBC with no Padding.

TDCNP() - Constructor for class com.ibm.crypto.fips.provider.TDCNP

Creates an instance of DESede cipher with CBC mode and no Padding.

toString() - Method in class com.ibm.crypto.fips.provider.DatawithDSA

Answers a string containing a concise, human-readable description of the receiver.

toString() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey

toString() - Method in class com.ibm.crypto.fips.provider.DHPublicKey

toString() - Method in class com.ibm.crypto.fips.provider.DSAPrivateKey

Returns a string containing a concise, human-readable description of the receiver.

toString() - Method in class com.ibm.crypto.fips.provider.DSAPublicKey

toString() - Method in class com.ibm.crypto.fips.provider.IdentityDatabase

toString() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey

toString() - Method in class com.ibm.crypto.fips.provider.RSAPrivateKey

toString() - Method in class com.ibm.crypto.fips.provider.RSAPublicKey

toString() - Method in class com.ibm.crypto.fips.provider.SHA1withDSA

Answers a string containing a concise, human-readable description of the receiver.

toString() - Method in class com.ibm.crypto.fips.provider.SystemIdentity

toString() - Method in class com.ibm.crypto.fips.provider.SystemSigner

U

unpad(byte[], int, int) - Method in interface com.ibm.crypto.fips.provider.Padding

Returns the index where padding starts.

Z

zeroize() - Method in class com.ibm.crypto.fips.provider.AESSecretKey



This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.DESedeKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.DHPrivateKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.DHPublicKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.DSAPrivateKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.DSAPublicKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.HmacSHA1
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.HmacSHA256
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.HmacSHA384
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.HmacSHA512
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.RSAPrivateCrtKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.RSAPrivateKey
This function zeroizes the key so that it isn't in memory
zeroize() - Method in class com.ibm.crypto.fips.provider.RSAPublicKey
This function zeroizes the key so that it isn't in memory.

Notices

Java is a registered trademark of SUN. Inc.

AIX, z/OS, AS/400 and IBM are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.

HP-UX is a registered trademark Hewlet Packard, Inc

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.



Linux is a registered trademark of Linus Torvalds.

Red Hat is a trademark of Red Hat, Inc.

SuSE is a registered trademark of SuSE AG

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

© 2008 International Business Machines Corporation. All rights reserved. This document may be freely reproduced and distributed in its entirety and without modification.