Cipher

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public class Cipher
extends Object (/reference/java/lang/Object)
```

java.lang.Object (/reference/java/lang/Object)
↳ javax.crypto.Cipher

Known direct subclasses
NullCipher (/reference/javax/crypto/NullCipher)

NullCipher (/reference/javax/crypto/NullCipher)	The NullCipher class is a class that provides an "identity cipher" -- one that does not transform the plain text.

This class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework.

In order to create a Cipher object, the application calls the Cipher's `getInstance` method, and passes the name of the requested *transformation* to it. Optionally, the name of a provider may be specified.

A *transformation* is a string that describes the operation (or set of operations) to be performed on the given input, to produce some output. A transformation always includes the name of a cryptographic algorithm (e.g., *DES*), and may be followed by a feedback mode and padding scheme.

A transformation is of the form:

- "*algorithm/mode/padding*" or
- "*algorithm*"

(in the latter case, provider-specific default values for the mode and padding scheme are used). For example, the following is a valid transformation:

```
Cipher c = Cipher.getInstance("DES/CBC/PKCS5Padding");
```

Using modes such as `CFB` and `OFB`, block ciphers can encrypt data in units smaller than the cipher's actual block size. When requesting such a mode, you may optionally specify the number of bits to be processed at a time by appending this number to the mode name as shown in the "`DES/CFB8/NoPadding`" and "`DES/OFB32/PKCS5Padding`" transformations. If no such number is specified, a provider-specific default is used. (For example, the SunJCE provider uses a default of 64 bits for DES.) Thus, block ciphers can be turned into byte-oriented stream ciphers by using an 8 bit mode such as CFB8 or OFB8.

Modes such as Authenticated Encryption with Associated Data (AEAD) provide authenticity assurances for both confidential data and Additional Associated Data (AAD) that is not encrypted. (Please see RFC 5116 (http://www.ietf.org/rfc/rfc5116.txt) for more information on AEAD and AEAD algorithms such as GCM/CCM.) Both confidential and AAD data can be used when calculating the authentication tag (similar to a `Mac` (/reference/javax/crypto/Mac)). This tag is appended to the ciphertext during encryption, and is verified on decryption.

AEAD modes such as GCM/CCM perform all AAD authenticity calculations before starting the ciphertext authenticity calculations. To avoid implementations having to internally buffer ciphertext, all AAD data must be supplied to GCM/CCM implementations (via the `updateAAD` methods) **before** the ciphertext is processed (via the `update` and `doFinal` methods).

Note that GCM mode has a uniqueness requirement on IVs used in encryption with a given key. When IVs are repeated for GCM encryption, such usages are subject to forgery attacks. Thus, after each encryption operation using GCM mode, callers should re-initialize the cipher objects with GCM parameters which has a different IV value.

```
    GCMParameterSpec s = ...;
    cipher.init(..., s);

    // If the GCM parameters were generated by the provider, it can
    // be retrieved by:
    // cipher.getParameters().getParameterSpec(GCMParameterSpec.class);

    cipher.updateAAD(...);  // AAD
    cipher.update(...);     // Multi-part update
    cipher.doFinal(...);    // conclusion of operation

    // Use a different IV value for every encryption
```

```
byte[] newIv = ...;
s = new GCMParameterSpec(s.getTLen(), newIv);
cipher.init(..., s);
...
```

Android provides the following `Cipher` transformations:

Algorithm	Modes	Paddings	Supported API Levels	Notes
AES	CBC CFB CTR CTS ECB OFB	ISO10126Padding NoPadding PKCS5Padding	1+	
	GCM	NoPadding	10+	
	GCM-SIV	NoPadding	30+	
AES_128	CBC ECB	NoPadding PKCS5Padding	26+	
	GCM	NoPadding	26+	
	GCM-SIV	NoPadding	30+	
AES_256	CBC ECB	NoPadding PKCS5Padding	26+	
	GCM	NoPadding	26+	
	GCM-SIV	NoPadding	30+	

Algorithm	Modes	Paddings	Supported API Levels	Notes
ARC4	ECB	NoPadding	10+	
	NONE	NoPadding	28+	
BLOWFISH	CBC CFB CTR CTS ECB OFB	ISO10126Padding NoPadding PKCS5Padding	10+	
ChaCha20	NONE Poly1305	NoPadding	28+	ChaCha with 20 rounds, 96-bit nonce, and 32-bit counter as described in RFC 7539.
DES	CBC CFB CTR CTS ECB OFB	ISO10126Padding NoPadding PKCS5Padding	1+	
DESede	CBC CFB CTR CTS ECB OFB	ISO10126Padding NoPadding PKCS5Padding	1+	
RSA	ECB NONE	NoPadding OAEPPadding PKCS1Padding	1+	
		OAEPwithSHA-1andMGF1Padding OAEPwithSHA-256andMGF1Padding	10+	

Algorithm	Modes	Paddings	Supported API Levels	Notes
		OAEPwithSHA-224andMGF1Padding OAEPwithSHA-384andMGF1Padding OAEPwithSHA-512andMGF1Padding	23+	

These transformations are described in the Cipher section (https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#Cipher) of the Java Cryptography Architecture Standard Algorithm Name Documentation.

See also:

`KeyGenerator` (/reference/javax/crypto/KeyGenerator)

`SecretKey` (/reference/javax/crypto/SecretKey)

Summary

Constants

`int`	`DECRYPT_MODE` (/reference/javax/crypto/Cipher#DECRYPT_MODE) Constant used to initialize cipher to decryption mode.
`int`	`ENCRYPT_MODE` (/reference/javax/crypto/Cipher#ENCRYPT_MODE) Constant used to initialize cipher to encryption mode.
`int`	`PRIVATE_KEY` (/reference/javax/crypto/Cipher#PRIVATE_KEY) Constant used to indicate the to-be-unwrapped key is a "private key".
`int`	`PUBLIC_KEY` (/reference/javax/crypto/Cipher#PUBLIC_KEY)

	Constant used to indicate the to-be-unwrapped key is a "public key".
`int`	SECRET_KEY (/reference/javax/crypto/Cipher#SECRET_KEY) Constant used to indicate the to-be-unwrapped key is a "secret key".
`int`	UNWRAP_MODE (/reference/javax/crypto/Cipher#UNWRAP_MODE) Constant used to initialize cipher to key-unwrapping mode.
`int`	WRAP_MODE (/reference/javax/crypto/Cipher#WRAP_MODE) Constant used to initialize cipher to key-wrapping mode.

Protected constructors

Cipher (/reference/javax/crypto/Cipher#Cipher(javax.crypto.CipherSpi,%20java.security.Provider,%20java.lang.St
(CipherSpi (/reference/javax/crypto/CipherSpi) `cipherSpi,` Provider (/reference/java/security/Provi
`provider,` String (/reference/java/lang/String) `transformation`)

Creates a Cipher object.

Public methods

`final int`	doFinal (/reference/javax/crypto/Cipher#doFina `byte[] output`) Encrypts or decrypts data in a single-part operat
`final int`	doFinal (/reference/javax/crypto/Cipher#doFina Finishes a multiple-part encryption or decryption
`final byte[]`	doFinal (/reference/javax/crypto/Cipher#doFina

	Finishes a multiple-part encryption or decryption
`final byte[]`	`doFinal` (/reference/javax/crypto/Cipher#doFina Encrypts or decrypts data in a single-part operat
`final int`	`doFinal` (/reference/javax/crypto/Cipher#doFina `inputLen, byte[] output, int outputOf` Encrypts or decrypts data in a single-part operat
`final int`	`doFinal` (/reference/javax/crypto/Cipher#doFina `input, ByteBuffer` (/reference/java/nio/ByteB Encrypts or decrypts data in a single-part operat
`final byte[]`	`doFinal` (/reference/javax/crypto/Cipher#doFina Encrypts or decrypts data in a single-part operat
`final String` (/reference/java/lang/String)	`getAlgorithm` (/reference/javax/crypto/Cipher# Returns the algorithm name of this `Cipher` objec
`final int`	`getBlockSize` (/reference/javax/crypto/Cipher# Returns the block size (in bytes).
`final ExemptionMechanism` (/reference/javax/crypto/ExemptionMechanism)	`getExemptionMechanism` (/reference/javax/cry Returns the exemption mechanism object used w
`final byte[]`	`getIV` (/reference/javax/crypto/Cipher#getIV()) (Returns the initialization vector (IV) in a new buffe
`static final Cipher` (/reference/javax/crypto/Cipher)	`getInstance` (/reference/javax/crypto/Cipher#g Returns a `Cipher` object that implements the spe

`static final` Cipher (/reference/javax/crypto/Cipher)	getInstance (/reference/javax/crypto/Cipher#g `transformation`, String (/reference/java/lan Returns a `Cipher` object that implements the spe
`static final` Cipher (/reference/javax/crypto/Cipher)	getInstance (/reference/javax/crypto/Cipher#g `transformation`, Provider (/reference/java/ Returns a `Cipher` object that implements the spe
`static final int`	getMaxAllowedKeyLength (/reference/javax/cr `transformation`) Returns the maximum key length for the specifiec
`static final` AlgorithmParameterSpec (/reference/java/security/spec/AlgorithmParameterSpec)	getMaxAllowedParameterSpec (/reference/jav (/reference/java/lang/String) `transformation` Returns an AlgorithmParameterSpec object whicl
`final int`	getOutputSize (/reference/javax/crypto/Cipher Returns the length in bytes that an output buffer the input length `inputLen` (in bytes).
`final` AlgorithmParameters (/reference/java/security/AlgorithmParameters)	getParameters (/reference/javax/crypto/Cipher Returns the parameters used with this cipher.
`final` Provider (/reference/java/security/Provider)	getProvider (/reference/javax/crypto/Cipher#g Returns the provider of this `Cipher` object.
`final void`	init (/reference/javax/crypto/Cipher#init(int,%2 (/reference/java/security/Key) `key`, Algorithı Initializes this cipher with a key and a set of algor
`final void`	init (/reference/javax/crypto/Cipher#init(int,%2 (/reference/java/security/cert/Certificate) `certi`

	Initializes this cipher with the public key from the
`final void`	`init` (/reference/javax/crypto/Cipher#init(int,%2((/reference/java/security/Key) `key, SecureRa` Initializes this cipher with a key and a source of ra
`final void`	`init` (/reference/javax/crypto/Cipher#init(int,%2((/reference/java/security/Key) `key, Algorithm` Initializes this cipher with a key and a set of algor
`final void`	`init` (/reference/javax/crypto/Cipher#init(int,%2(Initializes this cipher with a key.
`final void`	`init` (/reference/javax/crypto/Cipher#init(int,%20java `(int opmode, Key` (/reference/java/security/Ke (/reference/java/security/spec/AlgorithmParamet Initializes this cipher with a key, a set of algorithm
`final void`	`init` (/reference/javax/crypto/Cipher#init(int,%2((/reference/java/security/cert/Certificate) `certi` Initializes this cipher with the public key from the
`final void`	`init` (/reference/javax/crypto/Cipher#init(int,%2(`(int opmode, Key` (/reference/java/security/Ke `SecureRandom` (/reference/java/security/SecureI Initializes this cipher with a key, a set of algorithm
`final Key` (/reference/java/security/Key)	`unwrap` (/reference/javax/crypto/Cipher#unwrap (/reference/java/lang/String) `wrappedKeyAlgo` Unwrap a previously wrapped key.
`final byte[]`	`update` (/reference/javax/crypto/Cipher#update(

	Continues a multiple-part encryption or decrypti
`final int`	`update` (/reference/javax/crypto/Cipher#update(`byte[] output)` Continues a multiple-part encryption or decrypti
`final byte[]`	`update` (/reference/javax/crypto/Cipher#update(Continues a multiple-part encryption or decrypti
`final int`	`update` (/reference/javax/crypto/Cipher#update(`input, ByteBuffer` (/reference/java/nio/ByteB Continues a multiple-part encryption or decrypti
`final int`	`update` (/reference/javax/crypto/Cipher#update(`inputLen, byte[] output, int outputOf` Continues a multiple-part encryption or decrypti
`final void`	`updateAAD` (/reference/javax/crypto/Cipher#upd Continues a multi-part update of the Additional A
`final void`	`updateAAD` (/reference/javax/crypto/Cipher#upd Continues a multi-part update of the Additional A
`final void`	`updateAAD` (/reference/javax/crypto/Cipher#upd Continues a multi-part update of the Additional A
`final byte[]`	`wrap` (/reference/javax/crypto/Cipher#wrap(java. Wrap a key.

Inherited methods

From class `java.lang.Object` (/reference/java/lang/Object)

`Object` (/reference/java/lang/Object)	`clone` (/reference/java/lang/Object#clone())`()` Creates and returns a copy of this object.
`boolean`	`equals` (/reference/java/lang/Object#equals(java.lang.Object)) `(Object` (/reference/java/lang/Object) `obj)` Indicates whether some other object is "equal to" this one.
`void`	`finalize` (/reference/java/lang/Object#finalize())`()` Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
`final Class` (/reference/java/lang/Class)`<?>`	`getClass` (/reference/java/lang/Object#getClass())`()` Returns the runtime class of this `Object`.
`int`	`hashCode` (/reference/java/lang/Object#hashCode())`()` Returns a hash code value for the object.
`final void`	`notify` (/reference/java/lang/Object#notify())`()` Wakes up a single thread that is waiting on this object's monitor.
`final void`	`notifyAll` (/reference/java/lang/Object#notifyAll())`()` Wakes up all threads that are waiting on this object's monitor.
`String` (/reference/java/lang/String)	`toString` (/reference/java/lang/Object#toString())`()` Returns a string representation of the object.

`final void`	`wait` (/reference/java/lang/Object#wait(long,%20int))`(long timeoutMillis, int nanos)` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
`final void`	`wait` (/reference/java/lang/Object#wait(long))`(long timeoutMillis)` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
`final void`	`wait` (/reference/java/lang/Object#wait())`()` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*.

Constants

DECRYPT_MODE

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int DECRYPT_MODE
```

Constant used to initialize cipher to decryption mode.

Constant Value: 2 (0x00000002)

ENCRYPT_MODE

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPT_MODE
```

Constant used to initialize cipher to encryption mode.

Constant Value: 1 (0x00000001)

PRIVATE_KEY

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_KEY
```

Constant used to indicate the to-be-unwrapped key is a "private key".

Constant Value: 2 (0x00000002)

PUBLIC_KEY

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PUBLIC_KEY
```

Constant used to indicate the to-be-unwrapped key is a "public key".

Constant Value: 1 (0x00000001)

SECRET_KEY

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int SECRET_KEY
```

Constant used to indicate the to-be-unwrapped key is a "secret key".

Constant Value: 3 (0x00000003)

UNWRAP_MODE

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int UNWRAP_MODE
```

Constant used to initialize cipher to key-unwrapping mode.

Constant Value: 4 (0x00000004)

WRAP_MODE

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WRAP_MODE
```

Constant used to initialize cipher to key-wrapping mode.

Constant Value: 3 (0x00000003)

Protected constructors

Cipher

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected Cipher (CipherSpi (/reference/javax/crypto/CipherSpi) cipherSpi,
                Provider (/reference/java/security/Provider) provider,
                String (/reference/java/lang/String) transformation)
```

Creates a Cipher object.

Parameters

`cipherSpi`	`CipherSpi`: the delegate
`provider`	`Provider`: the provider

`transformation`	`String`: the transformation

Public methods

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int doFinal (byte[] input,
                int inputOffset,
                int inputLen,
                byte[] output)
```

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation. The data is encrypted or decrypted, depending on how this cipher was initialized.

The first `inputLen` bytes in the `input` buffer, starting at `inputOffset` inclusive, and any input bytes that may have been buffered during a previous `update` operation, are processed, with padding (if requested) being applied. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in the `output` buffer.

If the `output` buffer is too small to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Note: this method should be copy-safe, which means the `input` and `output` buffers can reference the same byte array and no unprocessed input data is overwritten when the result is copied into the output buffer.

Parameters

`input`	`byte`: the input buffer
`inputOffset`	`int`: the offset in `input` where the input starts
`inputLen`	`int`: the input length
`output`	`byte`: the buffer for the result

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.
`ShortBufferException` (/reference/javax/crypto/ShortBufferException)	if the given output buffer is too small to hold the result

BadPaddingException (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
AEADBadTagException (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int doFinal (byte[] output,
                int outputOffset)
```

Finishes a multiple-part encryption or decryption operation, depending on how this cipher was initialized.

Input data that may have been buffered during a previous `update` operation is processed, with padding (if requested) being applied. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in the `output` buffer, starting at `outputOffset` inclusive.

If the `output` buffer is too small to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Parameters

`output`	`byte`: the buffer for the result
`outputOffset`	`int`: the offset in `output` where the result is stored

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.
`ShortBufferException` (/reference/javax/crypto/ShortBufferException)	if the given output buffer is too small to hold the result
`BadPaddingException` (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
`AEADBadTagException` (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] doFinal ()
```

Finishes a multiple-part encryption or decryption operation, depending on how this cipher was initialized.

Input data that may have been buffered during a previous `update` operation is processed, with padding (if requested) being applied. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in a new buffer.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Returns

`byte[]`	the new buffer with the result

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.

BadPaddingException (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
AEADBadTagException (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] doFinal (byte[] input)
```

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation. The data is encrypted or decrypted, depending on how this cipher was initialized.

The bytes in the `input` buffer, and any input bytes that may have been buffered during a previous `update` operation, are processed, with padding (if requested) being applied. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in a new buffer.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Parameters

`input`	`byte`: the input buffer

Returns

`byte[]`	the new buffer with the result

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.
`BadPaddingException` (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
`AEADBadTagException` (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int doFinal (byte[] input,
                int inputOffset,
                int inputLen,
                byte[] output,
                int outputOffset)
```

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation. The data is encrypted or decrypted, depending on how this cipher was initialized.

The first `inputLen` bytes in the `input` buffer, starting at `inputOffset` inclusive, and any input bytes that may have been buffered during a previous `update` operation, are processed, with padding (if requested) being applied. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in the `output` buffer, starting at `outputOffset` inclusive.

If the `output` buffer is too small to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Note: this method should be copy-safe, which means the `input` and `output` buffers can reference the same byte array and no unprocessed input data is overwritten when the result is copied into the output buffer.

Parameters

`input`	`byte`: the input buffer
`inputOffset`	`int`: the offset in `input` where the input starts
`inputLen`	`int`: the input length
`output`	`byte`: the buffer for the result
`outputOffset`	`int`: the offset in `output` where the result is stored

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.
`ShortBufferException` (/reference/javax/crypto/ShortBufferException)	if the given output buffer is too small to hold the result
`BadPaddingException` (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
`AEADBadTagException` (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int doFinal (ByteBuffer (/reference/java/nio/ByteBuffer) input,
                ByteBuffer (/reference/java/nio/ByteBuffer) output)
```

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation. The data is encrypted or decrypted, depending on how this cipher was initialized.

All `input.remaining()` bytes starting at `input.position()` are processed. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in the output buffer. Upon return, the input buffer's position will be equal to its limit; its limit will not have changed. The output buffer's position will have advanced by n, where n is the value returned by this method; the output buffer's limit will not have changed.

If `output.remaining()` bytes are insufficient to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Note: this method should be copy-safe, which means the `input` and `output` buffers can reference the same byte array and no unprocessed input data is overwritten when the result is copied into the output buffer.

Parameters

`input`	`ByteBuffer`: the input ByteBuffer
`output`	`ByteBuffer`: the output ByteBuffer

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if input and output are the same object
`ReadOnlyBufferException` (/reference/java/nio/ReadOnlyBufferException)	if the output buffer is read-only
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.
`ShortBufferException` (/reference/javax/crypto/ShortBufferException)	if there is insufficient space in the output buffer
`BadPaddingException` (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
`AEADBadTagException` (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

doFinal

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] doFinal (byte[] input,
                int inputOffset,
                int inputLen)
```

Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation. The data is encrypted or decrypted, depending on how this cipher was initialized.

The first `inputLen` bytes in the `input` buffer, starting at `inputOffset` inclusive, and any input bytes that may have been buffered during a previous `update` operation, are processed, with padding (if requested) being applied. If an AEAD mode such as GCM/CCM is being used, the authentication tag is appended in the case of encryption, or verified in the case of decryption. The result is stored in a new buffer.

Upon finishing, this method resets this cipher object to the state it was in when previously initialized via a call to `init`. That is, the object is reset and available to encrypt or decrypt (depending on the operation mode that was specified in the call to `init`) more data.

Note: if any exception is thrown, this cipher object may need to be reset before it can be used again.

Parameters

`input`	`byte`: the input buffer
`inputOffset`	`int`: the offset in `input` where the input starts
`inputLen`	`int`: the input length

Returns

`byte[]`	the new buffer with the result

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested (only in encryption mode), and the total input length of the data processed by this cipher is not a multiple of block size; or if this encryption algorithm is unable to process the input data provided.
`BadPaddingException` (/reference/javax/crypto/BadPaddingException)	if this cipher is in decryption mode, and (un)padding has been requested, but the decrypted data is not bounded by the appropriate padding bytes
`AEADBadTagException` (/reference/javax/crypto/AEADBadTagException)	if this cipher is decrypting in an AEAD mode (such as GCM/CCM), and the received authentication tag does not match the calculated value

getAlgorithm

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final String (/reference/java/lang/String) getAlgorithm ()
```

Returns the algorithm name of this `Cipher` object.

This is the same name that was specified in one of the `getInstance` calls that created this `Cipher` object..

Returns

`String` (/reference/java/lang/String)	the algorithm name of this `Cipher` object.

getBlockSize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int getBlockSize ()
```

Returns the block size (in bytes).

Returns

`int`	the block size (in bytes), or 0 if the underlying algorithm is not a block cipher

getExemptionMechanism

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final ExemptionMechanism (/reference/javax/crypto/ExemptionMechanism) getExemptior
```

Returns the exemption mechanism object used with this cipher.

Returns

`ExemptionMechanism` (/reference/javax/crypto/ExemptionMechanism)	the exemption mechanism object used with this cipher, or null if this cipher does not use any exemption mechanism.

getIV

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] getIV ()
```

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

This is useful in the case where a random IV was created, or in the context of password-based encryption or decryption, where the IV is derived from a user-supplied password.

Returns

`byte[]`	the initialization vector in a new buffer, or null if the underlying algorithm does not use an IV, or if the IV has not yet been set.

getInstance

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final Cipher (/reference/javax/crypto/Cipher) getInstance (String (/reference/
```

Returns a `Cipher` object that implements the specified transformation.

This method traverses the list of registered security Providers, starting with the most preferred Provider. A new Cipher object encapsulating the CipherSpi implementation from the first Provider that supports the specified algorithm is returned.

Note that the list of registered providers may be retrieved via the `Security.getProviders()` (/reference/java/security/Security#getProviders()) method.

Parameters

`transformation`	`String`: the name of the transformation, e.g., *DES/CBC/PKCS5Padding*. in the Java Cryptography Architecture Standard Algorithm Name Docum (https://docs.oracle.com/javase/8/docs/technotes/guides/security/Stanc for information about standard transformation names.

Returns

`Cipher` (/reference/javax/crypto/Cipher)	a cipher that implements the requested transformation.

Throws

`NoSuchAlgorithmException` (/reference/java/security/NoSuchAlgorithmException)	if `transformation` is null, empty, in an invalid format, or if no Provider supports a CipherSpi implementation for the specified algorithm.
`NoSuchPaddingException` (/reference/javax/crypto/NoSuchPaddingException)	if `transformation` contains a padding scheme that is not available.

See also:

`Provider` (/reference/java/security/Provider)

getInstance

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final Cipher (/reference/javax/crypto/Cipher) getInstance (String (/reference/
                String (/reference/java/lang/String) provider)
```

Returns a `Cipher` object that implements the specified transformation.

A new Cipher object encapsulating the CipherSpi implementation from the specified provider is returned. The specified provider must be registered in the security provider list.

Note that the list of registered providers may be retrieved via the `Security.getProviders()` (/reference/java/security/Security#getProviders()) method.

Parameters

`transformation`	`String`: the name of the transformation, e.g., *DES/CBC/PKCS5Padding*. in the Java Cryptography Architecture Standard Algorithm Name Docum (https://docs.oracle.com/javase/8/docs/technotes/guides/security/Stanc for information about standard transformation names.
`provider`	`String`: the name of the provider.

Returns

`Cipher` (/reference/javax/crypto/Cipher)	a cipher that implements the requested transformation.

Throws

`NoSuchAlgorithmException` (/reference/java/security/NoSuchAlgorithmException)	if `transformation` is null, empty, in an invalid format, or if a CipherSpi implementation for the specified algorithm is not available from the specified provider.
`NoSuchProviderException` (/reference/java/security/NoSuchProviderException)	if the specified provider is not registered in the security provider list.
`NoSuchPaddingException` (/reference/javax/crypto/NoSuchPaddingException)	if `transformation` contains a padding scheme that is not available.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the `provider` is null or empty.

See also:

`Provider` (/reference/java/security/Provider)

getInstance

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final Cipher (/reference/javax/crypto/Cipher) getInstance (String (/reference/
                Provider (/reference/java/security/Provider) provider)
```

Returns a `Cipher` object that implements the specified transformation.

A new Cipher object encapsulating the CipherSpi implementation from the specified Provider object is returned. Note that the specified Provider object does not have to be registered in the provider list.

Parameters

`transformation`	`String`: the name of the transformation, e.g., *DES/CBC/PKCS5Padding*. in the Java Cryptography Architecture Standard Algorithm Name Docum (https://docs.oracle.com/javase/8/docs/technotes/guides/security/Stanc for information about standard transformation names.
`provider`	`Provider`: the provider.

Returns

Cipher (/reference/javax/crypto/Cipher)	a cipher that implements the requested transformation.

Throws

NoSuchAlgorithmException (/reference/java/security/NoSuchAlgorithmException)	if `transformation` is null, empty, in an invalid format, or if a CipherSpi implementation for the

	specified algorithm is not available from the specified Provider object.
`NoSuchPaddingException` (/reference/javax/crypto/NoSuchPaddingException)	if `transformation` contains a padding scheme that is not available.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the `provider` is null.

See also:

`Provider` (/reference/java/security/Provider)

getMaxAllowedKeyLength

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int getMaxAllowedKeyLength (String (/reference/java/lang/String) tra
```

Returns the maximum key length for the specified transformation according to the installed JCE jurisdiction policy files. If JCE unlimited strength jurisdiction policy files are installed, Integer.MAX_VALUE will be returned. For more information on default key size in JCE jurisdiction policy files, please see Appendix E in the Java Cryptography Architecture Reference Guide (https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#AppC).

Parameters

`transformation`	`String`: the cipher transformation.

Returns

`int`	the maximum key length in bits or Integer.MAX_VALUE.

Throws

NullPointerException (/reference/java/lang/NullPointerException)	if `transformation` is null.
NoSuchAlgorithmException (/reference/java/security/NoSuchAlgorithmException)	if `transformation` is not a valid transformation, i.e. in the form of "algorithm" or "algorithm/mode/padding".

getMaxAllowedParameterSpec

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final AlgorithmParameterSpec (/reference/java/security/spec/AlgorithmParamete
```

Returns an AlgorithmParameterSpec object which contains the maximum cipher parameter value according to the jurisdiction policy file. If JCE unlimited strength jurisdiction policy files are installed or there is no maximum limit on the parameters for the specified transformation in the policy file, null will be returned.

Parameters

`transformation`	`String`: the cipher transformation.

Returns

AlgorithmParameterSpec (/reference/java/security/spec/AlgorithmParameterSpec)	an AlgorithmParameterSpec which holds the maximum value or null.

Throws

NullPointerException (/reference/java/lang/NullPointerException)	if `transformation` is null.
NoSuchAlgorithmException (/reference/java/security/NoSuchAlgorithmException)	if `transformation` is not a valid transformation, i.e. in the form of "algorithm" or "algorithm/mode/padding".

getOutputSize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int getOutputSize (int inputLen)
```

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).

This call takes into account any unprocessed (buffered) data from a previous `update` call, padding, and AEAD tagging.

The actual output length of the next `update` or `doFinal` call may be smaller than the length returned by this method.

Parameters

`inputLen`	`int`: the input length (in bytes)

Returns

`int`	the required output buffer size (in bytes)

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not yet been initialized)

getParameters

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final AlgorithmParameters (/reference/java/security/AlgorithmParameters) getParamete
```

Returns the parameters used with this cipher.

The returned parameters may be the same that were used to initialize this cipher, or may contain a combination of default and random parameter values used by the underlying cipher implementation if this cipher requires algorithm parameters but was not initialized with any.

Returns

`AlgorithmParameters` (/reference/java/security/AlgorithmParameters)	the parameters used with this cipher, or null if this cipher does not use any parameters.

getProvider

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final Provider (/reference/java/security/Provider) getProvider ()
```

Returns the provider of this `Cipher` object.

Returns

`Provider` (/reference/java/security/Provider)	the provider of this `Cipher` object

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Key (/reference/java/security/Key) key,
                AlgorithmParameters (/reference/java/security/AlgorithmParameters) params)
```

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

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If this cipher requires any algorithm parameters and `params` is null, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidAlgorithmParameterException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them using the `SecureRandom` (/reference/java/security/SecureRandom) implementation of the highest-priority installed provider as the source of randomness. (If none of the installed providers supply an implementation of SecureRandom, a system-provided source of randomness will be used.)

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)
`key`	`Key`: the encryption key
`params`	`AlgorithmParameters`: the algorithm parameters

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the given key is inappropriate for initializing this cipher, or its keysize exceeds the maximum allowable keysize (as determined from the configured jurisdiction policy files).
`InvalidAlgorithmParameterException` (/reference/java/security/InvalidAlgorithmParameterException)	if the given algorithm parameters are inappropriate for this cipher, or this cipher requires algorithm parameters and `params` is null, or the given algorithm parameters imply a cryptographic strength that would exceed the legal limits (as determined from the configured jurisdiction policy files).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is `WRAP_MODE` or `UNWRAP_MODE` but the mode is not implemented by the underlying `CipherSpi`.

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Certificate (/reference/java/security/cert/Certificate) certificate,
                SecureRandom (/reference/java/security/SecureRandom) random)
```

Initializes this cipher with the public key from the given certificate and a source of randomness.

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If the certificate is of type X.509 and has a *key usage* extension field marked as critical, and the value of the *key usage* extension field implies that the public key in the certificate and its corresponding private key are not supposed to be used for the operation represented by the value of `opmode`, an `InvalidKeyException` is thrown.

If this cipher requires any algorithm parameters that cannot be derived from the public key in the given `certificate`, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidKeyException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them from `random`.

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)

`certificate`	`Certificate`: the certificate
`random`	`SecureRandom`: the source of randomness

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the public key in the given certificate is inappropriate for initializing this cipher, or this cipher requires algorithm parameters that cannot be determined from the public key in the given certificate, or the keysize of the public key in the given certificate has a keysize that exceeds the maximum allowable keysize (as determined by the configured jurisdiction policy files).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is `WRAP_MODE` or `UNWRAP_MODE` but the mode is not implemented by the underlying `CipherSpi`.

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Key (/reference/java/security/Key) key,
                SecureRandom (/reference/java/security/SecureRandom) random)
```

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

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If this cipher requires any algorithm parameters that cannot be derived from the given `key`, the underlying cipher implementation is supposed to generate the required parameters itself

(using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidKeyException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them from `random`.

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)
`key`	`Key`: the encryption key
`random`	`SecureRandom`: the source of randomness

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the given key is inappropriate for initializing this cipher, or requires algorithm parameters that cannot be determined from the given key, or if the given key has a keysize that exceeds the

	maximum allowable keysize (as determined from the configured jurisdiction policy files).
UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is **WRAP_MODE** or **UNWRAP_MODE** but the mode is not implemented by the underlying **CipherSpi**.

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Key (/reference/java/security/Key) key,
                AlgorithmParameterSpec (/reference/java/security/spec/AlgorithmParameterSpe
```

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

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If this cipher requires any algorithm parameters and `params` is null, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidAlgorithmParameterException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them using the `SecureRandom` (/reference/java/security/SecureRandom) implementation of the highest-priority installed provider as the source of randomness. (If none of the installed providers supply an implementation of SecureRandom, a system-provided source of randomness will be used.)

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)
`key`	`Key`: the encryption key
`params`	`AlgorithmParameterSpec`: the algorithm parameters

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the given key is inappropriate for initializing this cipher, or its keysize exceeds the maximum allowable keysize (as determined from the configured jurisdiction policy files).
`InvalidAlgorithmParameterException` (/reference/java/security/InvalidAlgorithmParameterException)	if the given algorithm parameters are inappropriate for this cipher, or this cipher requires algorithm parameters and `params` is null, or the given algorithm parameters imply a cryptographic strength that would exceed the legal limits (as determined from the configured jurisdiction policy files).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is `WRAP_MODE` or `UNWRAP_MODE` but the mode is not implemented by the underlying `CipherSpi`.

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Key (/reference/java/security/Key) key)
```

Initializes this cipher with a key.

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If this cipher requires any algorithm parameters that cannot be derived from the given `key`, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidKeyException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them using the `SecureRandom` (/reference/java/security/SecureRandom) implementation of the highest-priority installed provider as the source of randomness. (If none of the installed providers supply an implementation of SecureRandom, a system-provided source of randomness will be used.)

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)

`key`	**`Key`**: the key

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the given key is inappropriate for initializing this cipher, or requires algorithm parameters that cannot be determined from the given key, or if the given key has a keysize that exceeds the maximum allowable keysize (as determined from the configured jurisdiction policy files).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is **`WRAP_MODE`** or **`UNWRAP_MODE`** but the mode is not implemented by the underlying **`CipherSpi`**.

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                **Key** (/reference/java/security/Key) key,
                **AlgorithmParameterSpec** (/reference/java/security/spec/AlgorithmParameterSpe
                **SecureRandom** (/reference/java/security/SecureRandom) random)
```

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

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If this cipher requires any algorithm parameters and `params` is null, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidAlgorithmParameterException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters`

(/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them from `random`.

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)
`key`	`Key`: the encryption key
`params`	`AlgorithmParameterSpec`: the algorithm parameters
`random`	`SecureRandom`: the source of randomness

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the given key is inappropriate for initializing this cipher, or its keysize exceeds the maximum allowable keysize (as determined from the configured jurisdiction policy files).

InvalidAlgorithmParameterException (/reference/java/security/InvalidAlgorithmParameterException)	if the given algorithm parameters are inappropriate for this cipher, or this cipher requires algorithm parameters and **params** is null, or the given algorithm parameters imply a cryptographic strength that would exceed the legal limits (as determined from the configured jurisdiction policy files).
UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is **WRAP_MODE** or **UNWRAP_MODE** but the mode is not implemented by the underlying **CipherSpi**.

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Certificate (/reference/java/security/cert/Certificate) certificate)
```

Initializes this cipher with the public key from the given certificate.

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If the certificate is of type X.509 and has a *key usage* extension field marked as critical, and the value of the *key usage* extension field implies that the public key in the certificate and its corresponding private key are not supposed to be used for the operation represented by the value of `opmode`, an `InvalidKeyException` is thrown.

If this cipher requires any algorithm parameters that cannot be derived from the public key in the given certificate, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidKeyException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them using the `SecureRandom` implementation of the highest-priority installed provider as the source of randomness. (If none of the installed providers supply an implementation of SecureRandom, a system-provided source of randomness will be used.)

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)
`certificate`	`Certificate`: the certificate

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the public key in the given certificate is inappropriate for initializing this cipher, or this cipher requires algorithm parameters that cannot be determined from the public key in the given certificate, or the keysize of the public key in the given certificate has a keysize that exceeds the maximum allowable keysize (as determined by the configured jurisdiction policy files).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is `WRAP_MODE` or `UNWRAP_MODE` but the mode is not implemented

by the underlying `CipherSpi`.

---

init

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void init (int opmode,
                Key (/reference/java/security/Key) key,
                AlgorithmParameters (/reference/java/security/AlgorithmParameters) params,
                SecureRandom (/reference/java/security/SecureRandom) random)
```

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

The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of `opmode`.

If this cipher requires any algorithm parameters and `params` is null, the underlying cipher implementation is supposed to generate the required parameters itself (using provider-specific default or random values) if it is being initialized for encryption or key wrapping, and raise an `InvalidAlgorithmParameterException` if it is being initialized for decryption or key unwrapping. The generated parameters can be retrieved using `getParameters` (/reference/javax/crypto/Cipher#getParameters()) or `getIV` (/reference/javax/crypto/Cipher#getIV()) (if the parameter is an IV).

If this cipher requires algorithm parameters that cannot be derived from the input parameters, and there are no reasonable provider-specific default values, initialization will necessarily fail.

If this cipher (including its underlying feedback or padding scheme) requires any random bytes (e.g., for parameter generation), it will get them from `random`.

Note that when a Cipher object is initialized, it loses all previously-acquired state. In other words, initializing a Cipher is equivalent to creating a new instance of that Cipher and initializing it.

Parameters

`opmode`	`int`: the operation mode of this cipher (this is one of the following: `ENCRYPT_MODE`, `DECRYPT_MODE`, `WRAP_MODE` or `UNWRAP_MODE`)
`key`	`Key`: the encryption key
`params`	`AlgorithmParameters`: the algorithm parameters
`random`	`SecureRandom`: the source of randomness

Throws

`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if the given key is inappropriate for initializing this cipher, or its keysize exceeds the maximum allowable keysize (as determined from the configured jurisdiction policy files).
`InvalidAlgorithmParameterException` (/reference/java/security/InvalidAlgorithmParameterException)	if the given algorithm parameters are inappropriate for this cipher, or this cipher requires algorithm parameters and `params` is null, or the given algorithm parameters imply a cryptographic strength that would exceed the legal limits (as determined from the configured jurisdiction policy files).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if (@code opmode} is `WRAP_MODE` or `UNWRAP_MODE` but the mode is not implemented by the underlying `CipherSpi`.

unwrap

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final Key (/reference/java/security/Key) unwrap (byte[] wrappedKey,
                String (/reference/java/lang/String) wrappedKeyAlgorithm,
                int wrappedKeyType)
```

Unwrap a previously wrapped key.

Parameters

`wrappedKey`	`byte`: the key to be unwrapped.
`wrappedKeyAlgorithm`	`String`: the algorithm associated with the wrapped key.
`wrappedKeyType`	`int`: the type of the wrapped key. This must be one of `SECRET_KEY`, `PRIVATE_KEY`, or `PUBLIC_KEY`.

Returns

Key (/reference/java/security/Key)	the unwrapped key.

Throws

IllegalStateException (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized).
NoSuchAlgorithmException (/reference/java/security/NoSuchAlgorithmException)	if no installed providers can create keys of type `wrappedKeyType` for the `wrappedKeyAlgorithm`.

InvalidKeyException (/reference/java/security/InvalidKeyException)	if **wrappedKey** does not represent a wrapped key of type **wrappedKeyType** for the **wrappedKeyAlgorithm**.
UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the corresponding method in the **CipherSpi** is not supported.

update

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] update (byte[] input)
```

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

The bytes in the **input** buffer are processed, and the result is stored in a new buffer.

If **input** has a length of zero, this method returns **null**.

Parameters

input	**byte**: the input buffer

Returns

byte[]	the new buffer with the result, or null if the underlying cipher is a block cipher and the input data is too short to result in a new block.

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)

update

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int update (byte[] input,
                int inputOffset,
                int inputLen,
                byte[] output)
```

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

The first `inputLen` bytes in the `input` buffer, starting at `inputOffset` inclusive, are processed, and the result is stored in the `output` buffer.

If the `output` buffer is too small to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

If `inputLen` is zero, this method returns a length of zero.

Note: this method should be copy-safe, which means the `input` and `output` buffers can reference the same byte array and no unprocessed input data is overwritten when the result is copied into the output buffer.

Parameters

`input`	`byte`: the input buffer
`inputOffset`	`int`: the offset in `input` where the input starts

`inputLen`	`int`: the input length
`output`	`byte`: the buffer for the result

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`ShortBufferException` (/reference/javax/crypto/ShortBufferException)	if the given output buffer is too small to hold the result

update

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] update (byte[] input,
                int inputOffset,
                int inputLen)
```

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

The first `inputLen` bytes in the `input` buffer, starting at `inputOffset` inclusive, are processed, and the result is stored in a new buffer.

If `inputLen` is zero, this method returns `null`.

Parameters

`input`	`byte`: the input buffer
`inputOffset`	`int`: the offset in `input` where the input starts
`inputLen`	`int`: the input length

Returns

`byte[]`	the new buffer with the result, or null if the underlying cipher is a block cipher and the input data is too short to result in a new block.

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)

update

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int update (ByteBuffer (/reference/java/nio/ByteBuffer) input,
                ByteBuffer (/reference/java/nio/ByteBuffer) output)
```

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

All `input.remaining()` bytes starting at `input.position()` are processed. The result is stored in the output buffer. Upon return, the input buffer's position will be equal to its limit;

its limit will not have changed. The output buffer's position will have advanced by n, where n is the value returned by this method; the output buffer's limit will not have changed.

If `output.remaining()` bytes are insufficient to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

Note: this method should be copy-safe, which means the `input` and `output` buffers can reference the same block of memory and no unprocessed input data is overwritten when the result is copied into the output buffer.

Parameters

`input`	`ByteBuffer`: the input ByteBuffer
`output`	`ByteBuffer`: the output ByteByffer

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if input and output are the same object

ReadOnlyBufferException (/reference/java/nio/ReadOnlyBufferException)	if the output buffer is read-only
ShortBufferException (/reference/javax/crypto/ShortBufferException)	if there is insufficient space in the output buffer

update

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int update (byte[] input,
                int inputOffset,
                int inputLen,
                byte[] output,
                int outputOffset)
```

Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

The first `inputLen` bytes in the `input` buffer, starting at `inputOffset` inclusive, are processed, and the result is stored in the `output` buffer, starting at `outputOffset` inclusive.

If the `output` buffer is too small to hold the result, a `ShortBufferException` is thrown. In this case, repeat this call with a larger output buffer. Use `getOutputSize` (/reference/javax/crypto/Cipher#getOutputSize(int)) to determine how big the output buffer should be.

If `inputLen` is zero, this method returns a length of zero.

Note: this method should be copy-safe, which means the `input` and `output` buffers can reference the same byte array and no unprocessed input data is overwritten when the result is copied into the output buffer.

Parameters

`input`	`byte`: the input buffer

`inputOffset`	`int`: the offset in `input` where the input starts
`inputLen`	`int`: the input length
`output`	`byte`: the buffer for the result
`outputOffset`	`int`: the offset in `output` where the result is stored

Returns

`int`	the number of bytes stored in `output`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized)
`ShortBufferException` (/reference/javax/crypto/ShortBufferException)	if the given output buffer is too small to hold the result

updateAAD

Added in API level 19 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void updateAAD (byte[] src,
                int offset,
                int len)
```

Continues a multi-part update of the Additional Authentication Data (AAD), using a subset of the provided buffer.

Calls to this method provide AAD to the cipher when operating in modes such as AEAD (GCM/CCM). If this cipher is operating in either GCM or CCM mode, all AAD must be supplied before beginning operations on the ciphertext (via the `update` and `doFinal` methods).

Parameters

`src`	`byte`: the buffer containing the AAD
`offset`	`int`: the offset in `src` where the AAD input starts
`len`	`int`: the number of AAD bytes

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the `src` byte array is null, or the `offset` or `length` is less than 0, or the sum of the `offset` and `len` is greater than the length of the `src` byte array
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized), does not accept AAD, or if operating in either GCM or CCM mode and one of the `update` methods has already been called for the active encryption/decryption operation
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if the corresponding method in the `CipherSpi` has not been overridden by an implementation

updateAAD

Added in API level 19 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void updateAAD (ByteBuffer (/reference/java/nio/ByteBuffer) src)
```

Continues a multi-part update of the Additional Authentication Data (AAD).

Calls to this method provide AAD to the cipher when operating in modes such as AEAD (GCM/CCM). If this cipher is operating in either GCM or CCM mode, all AAD must be supplied before beginning operations on the ciphertext (via the `update` and `doFinal` methods).

All `src.remaining()` bytes starting at `src.position()` are processed. Upon return, the input buffer's position will be equal to its limit; its limit will not have changed.

Parameters

`src`	`ByteBuffer`: the buffer containing the AAD

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the `src ByteBuffer` is null
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized), does not accept AAD, or if operating in either GCM or CCM mode and one of the `update` methods has already been called for the active encryption/decryption operation
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if the corresponding method in the `CipherSpi` has not been overridden by an implementation

updateAAD

Added in API level 19 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void updateAAD (byte[] src)
```

Continues a multi-part update of the Additional Authentication Data (AAD).

Calls to this method provide AAD to the cipher when operating in modes such as AEAD (GCM/CCM). If this cipher is operating in either GCM or CCM mode, all AAD must be supplied before beginning operations on the ciphertext (via the `update` and `doFinal` methods).

Parameters

`src`	`byte`: the buffer containing the Additional Authentication Data

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the `src` byte array is null
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized), does not accept AAD, or if operating in either GCM or CCM mode and one of the `update` methods has already been called for the active encryption/decryption operation
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if the corresponding method in the `CipherSpi` has not been overridden by an implementation

wrap

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final byte[] wrap (Key (/reference/java/security/Key) key)
```

Wrap a key.

Parameters

`key`	`Key`: the key to be wrapped.

Returns

`byte[]`	the wrapped key.

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if this cipher is in a wrong state (e.g., has not been initialized).
`IllegalBlockSizeException` (/reference/javax/crypto/IllegalBlockSizeException)	if this cipher is a block cipher, no padding has been requested, and the length of the encoding of the key to be wrapped is not a multiple of the block size.
`InvalidKeyException` (/reference/java/security/InvalidKeyException)	if it is impossible or unsafe to wrap the key with this cipher (e.g., a hardware protected key is being passed to a software-only cipher).
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if the corresponding method in the `CipherSpi` is not supported.