Module java.base
Package javax.crypto

Class KEM

java.lang.Object
    javax.crypto.KEM

```
public final class KEM
extends Object
```

This class provides the functionality of a Key Encapsulation Mechanism (KEM). A KEM can be used to secure symmetric keys using asymmetric or public key cryptography between two parties. The sender calls the encapsulate method to generate a secret key and a key encapsulation message, and the receiver calls the decapsulate method to recover the same secret key from the key encapsulation message.

The `getInstance` method creates a new `KEM` object that implements the specified algorithm.

A `KEM` object is immutable. It is safe to call multiple `newEncapsulator` and `newDecapsulator` methods on the same `KEM` object at the same time.

If a provider is not specified in the `getInstance` method when instantiating a `KEM` object, the `newEncapsulator` and `newDecapsulator` methods may return encapsulators or decapsulators from different providers. The provider selected is based on the parameters passed to the `newEncapsulator` or `newDecapsulator` methods: the private or public key and the optional `AlgorithmParameterSpec`. The `KEM.Encapsulator.providerName()` and `KEM.Decapsulator.providerName()` methods return the name of the selected provider.

`Encapsulator` and `Decapsulator` objects are also immutable. It is safe to invoke multiple `encapsulate` and `decapsulate` methods on the same `Encapsulator` or `Decapsulator` object at the same time. Each invocation of `encapsulate` will generate a new shared secret and key encapsulation message.

Example:

```
    // Receiver side
    var kpg = KeyPairGenerator.getInstance("X25519");
    var kp = kpg.generateKeyPair();

    // Sender side
    var kem1 = KEM.getInstance("DHKEM");
    var sender = kem1.newEncapsulator(kp.getPublic());
    var encapsulated = sender.encapsulate();
    var k1 = encapsulated.key();

    // Receiver side
    var kem2 = KEM.getInstance("DHKEM");
    var receiver = kem2.newDecapsulator(kp.getPrivate());
    var k2 = receiver.decapsulate(encapsulated.encapsulation());

    assert Arrays.equals(k1.getEncoded(), k2.getEncoded());
```

Since:

21

Nested Class Summary

Nested Classes

Modifier and Type	Class	Description
`static final class`	**`KEM.Decapsulator`**	A decapsulator, generated by `newDecapsulator(java.security.PrivateKey)` on the KEM receiver side.
`static final class`	**`KEM.Encapsulated`**	This class specifies the return value of the encapsulate method of a Key Encapsulation Mechanism (KEM), which includes the shared secret (as a `SecretKey`), the key encapsulation message, and optional parameters.
`static final class`	**`KEM.Encapsulator`**	An encapsulator, generated by `newEncapsulator(java.security.PublicKey)` on the KEM sender side.

Method Summary

All Methods **Static Methods** **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description
`String`	`getAlgorithm()`	Returns the name of the algorithm for this KEM object.
`static KEM`	`getInstance(String algorithm)`	Returns a KEM object that implements the specified algorithm.
`static KEM`	`getInstance(String algorithm, String provider)`	Returns a KEM object that implements the specified algorithm from the specified security provider.
`static KEM`	`getInstance(String algorithm, Provider provider)`	Returns a KEM object that implements the specified algorithm from the specified security provider.
`KEM.Decapsulator`	`newDecapsulator(PrivateKey privateKey)`	Creates a KEM decapsulator on the KEM receiver side.
`KEM.Decapsulator`	`newDecapsulator(PrivateKey privateKey, AlgorithmParameterSpec spec)`	Creates a KEM decapsulator on the KEM receiver side.
`KEM.Encapsulator`	`newEncapsulator(PublicKey publicKey)`	Creates a KEM encapsulator on the KEM sender side.
`KEM.Encapsulator`	`newEncapsulator(PublicKey publicKey, SecureRandom secureRandom)`	Creates a KEM encapsulator on the KEM sender side.
`KEM.Encapsulator`	`newEncapsulator(PublicKey publicKey, AlgorithmParameterSpec spec, SecureRandom secureRandom)`	Creates a KEM encapsulator on the KEM sender side.

Methods declared in class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Details

getInstance

```
public static KEM getInstance(String algorithm)
                       throws NoSuchAlgorithmException
```

Returns a `KEM` object that implements the specified algorithm.

Parameters:

`algorithm` - the name of the KEM algorithm. See the `KEM` section in the Java Security Standard Algorithm Names Specification for information about standard KEM algorithm names.

Returns:

the new `KEM` object

Throws:

`NoSuchAlgorithmException` - if no `Provider` supports a `KEM` implementation for the specified algorithm

`NullPointerException` - if `algorithm` is `null`

getInstance

```
public static KEM getInstance(String algorithm,
                              Provider provider)
                       throws NoSuchAlgorithmException
```

Returns a `KEM` object that implements the specified algorithm from the specified security provider.

Parameters:

`algorithm` - the name of the KEM algorithm. See the `KEM` section in the Java Security Standard Algorithm Names Specification for information about standard KEM algorithm names.

`provider` - the provider. If `null`, this method is equivalent to `getInstance(String)`.

Returns:

the new `KEM` object

Throws:

`NoSuchAlgorithmException` - if a `provider` is specified and it does not support the specified KEM algorithm, or if `provider` is `null` and there is no provider that supports a KEM implementation of the specified algorithm

NullPointerException - if algorithm is null

getInstance

```
public static KEM getInstance(String algorithm,
                              String provider)
                       throws NoSuchAlgorithmException,
                              NoSuchProviderException
```

Returns a KEM object that implements the specified algorithm from the specified security provider.

Parameters:

algorithm - the name of the KEM algorithm. See the KEM section in the Java Security Standard Algorithm Names Specification for information about standard KEM algorithm names.

provider - the provider. If null, this method is equivalent to getInstance(String).

Returns:

the new KEM object

Throws:

NoSuchAlgorithmException - if a provider is specified and it does not support the specified KEM algorithm, or if provider is null and there is no provider that supports a KEM implementation of the specified algorithm

NoSuchProviderException - if the specified provider is not registered in the security provider list

NullPointerException - if algorithm is null

newEncapsulator

```
public KEM.Encapsulator newEncapsulator(PublicKey publicKey)
                                 throws InvalidKeyException
```

Creates a KEM encapsulator on the KEM sender side.

This method is equivalent to newEncapsulator(publicKey, null, null).

Parameters:

`publicKey` - the receiver's public key, must not be `null`

Returns:

the encapsulator for this key

Throws:

`InvalidKeyException` - if `publicKey` is `null` or invalid

`UnsupportedOperationException` - if this method is not supported because an `AlgorithmParameterSpec` must be provided

newEncapsulator

```
public KEM.Encapsulator newEncapsulator(PublicKey publicKey,
                                        SecureRandom secureRandom)
                                 throws InvalidKeyException
```

Creates a KEM encapsulator on the KEM sender side.

This method is equivalent to `newEncapsulator(publicKey, null, secureRandom)`.

Parameters:

`publicKey` - the receiver's public key, must not be `null`

`secureRandom` - the source of randomness for encapsulation. If null, a default one from the implementation will be used.

Returns:

the encapsulator for this key

Throws:

`InvalidKeyException` - if `publicKey` is `null` or invalid

`UnsupportedOperationException` - if this method is not supported because an `AlgorithmParameterSpec` must be provided

newEncapsulator

```
public KEM.Encapsulator newEncapsulator(PublicKey publicKey,
                                        AlgorithmParameterSpec spec,
                                        SecureRandom secureRandom)
                                 throws InvalidAlgorithmParameterException,
                                        InvalidKeyException
```

Creates a KEM encapsulator on the KEM sender side.

An algorithm can define an `AlgorithmParameterSpec` child class to provide extra information in this method. This is especially useful if the same key can be used to derive shared secrets in different ways. If any extra information inside this object needs to be transmitted along with the key encapsulation message so that the receiver is able to create a matching decapsulator, it will be included as a byte array in the `KEM.Encapsulated.params` field inside the encapsulation output. In this case, the security provider should provide an `AlgorithmParameters` implementation using the same algorithm name as the KEM. The receiver can initiate such an `AlgorithmParameters` instance with the `params` byte array received and recover an `AlgorithmParameterSpec` object to be used in its `newDecapsulator(PrivateKey, AlgorithmParameterSpec)` call.

Parameters:

`publicKey` - the receiver's public key, must not be `null`

`spec` - the optional parameter, can be `null`

`secureRandom` - the source of randomness for encapsulation. If null, a default one from the implementation will be used.

Returns:

the encapsulator for this key

Throws:

`InvalidAlgorithmParameterException` - if `spec` is invalid or one is required but `spec` is `null`

`InvalidKeyException` - if `publicKey` is `null` or invalid

newDecapsulator

```
public KEM.Decapsulator newDecapsulator(PrivateKey privateKey)
                                 throws InvalidKeyException
```

Creates a KEM decapsulator on the KEM receiver side.

This method is equivalent to `newDecapsulator(privateKey, null)`.

Parameters:

`privateKey` - the receiver's private key, must not be `null`

Returns:

the decapsulator for this key

Throws:

`InvalidKeyException` - if `privateKey` is `null` or invalid

`UnsupportedOperationException` - if this method is not supported because an `AlgorithmParameterSpec` must be provided

newDecapsulator

```
public KEM.Decapsulator newDecapsulator(PrivateKey privateKey,
                                        AlgorithmParameterSpec spec)
                                 throws InvalidAlgorithmParameterException,
                                        InvalidKeyException
```

Creates a KEM decapsulator on the KEM receiver side.

Parameters:

`privateKey` - the receiver's private key, must not be `null`

`spec` - the parameter, can be `null`

Returns:

the decapsulator for this key

Throws:

`InvalidAlgorithmParameterException` - if `spec` is invalid or one is required but `spec` is `null`

`InvalidKeyException` - if `privateKey` is `null` or invalid

getAlgorithm

```
public String getAlgorithm()
```

Returns the name of the algorithm for this `KEM` object.

Returns:

the name of the algorithm for this `KEM` object.