iaik.pkcs.pkcs5

Class PBKDF2

java.lang.Object
    javax.crypto.KeyGeneratorSpi
        iaik.pkcs.pkcs5.PBKDF2

Direct Known Subclasses:

PBKDF2.PBKDF2WithHmacSHA1, PBKDF2.PBKDF2WithHmacSHA224, PBKDF2.PBKDF2WithHmacSHA256, PBKDF2.PBKDF2WithHmacSHA384, PBKDF2.PBKDF2WithHmacSHA512

```
public class PBKDF2
extends javax.crypto.KeyGeneratorSpi
```

This class implements a KeyGenerator for the PBKDF2 (password-based-key-derivation-function-2) specified by the PKCS#5 v2.1 Password-Based Cryptography Standard to derive a key from a password.

The PBKDF2 key derivation function PBKDF2 needs the following parameters: salt value, iteration count, length of the to-be-derived key, and (MAC based) pseudo random function (default: HMCA/SHA1). After creating a PBKDF2 KeyGenerator you have to specify salt value, iteration count and length of the to-be-derived key as `PBEKeyAndParameterSpec` object. If you want to use another pseudorandom function than HMAC/SHA1 you may use a `PBKDF2KeyAndParameterSpec` object allowing to specify an alternative mac function by its AlgorithmID. Both parameter classes also need the (encoded) password from which to derive the secret key.

The following example uses the PBKDF2 KeyGenerator to derive an AES key from a password:

```
char[] password = { 't', 'o', 'p', 'S', 'e', 'c', 'r', 'e', 't' };
// create a KeySpec from our password
PBEKeySpec keySpec = new PBEKeySpec(password);
// use the "PKCS#5" or "PBE" SecretKeyFactory to convert the password
SecretKeyFactory kf = SecretKeyFactory.getInstance("PKCS#5", "IAIK");
// create an appropriate PbeKey
PBEKey pbeKey = (PBEKey)kf.generateSecret(keySpec);
// create PBKDF2 KeyGenerator
KeyGenerator pbkdf2 = KeyGenerator.getInstance("PBKDF2", "IAIK");
int iterationCount = 2000;
byte[] salt = new byte[32];
SecureRandom random = ...;
random.nextBytes(salt);
int derivedKeyLength = 16;
PBEKeyAndParameterSpec parameterSpec =
  new PBEKeyAndParameterSpec(pbeKey.getEncoded(),
                             salt,
                             iterationCount,
                             derivedKeyLength);

pbkdf2.init(parameterSpec, random);
SecretKey derivedKey = pbkdf2.generateKey();
String keyName = "AES";
// use SecretKeyFactory to set the right key format
SecretKeySpec spec = new SecretKeySpec(derivedKey.getEncoded(), keyName);
SecretKeyFactory skf = SecretKeyFactory.getInstance(keyName, "IAIK");
SecretKey cipherKey = skf.generateSecret(spec);
```

As mentioned above you may use a `PBKDF2KeyAndParameterSpec` object to specify another (mac based) pseudo random function than the default HMAC/SHA1, e.g.:

```
PBKDF2KeyAndParameterSpec parameterSpec =
  new PBKDF2KeyAndParameterSpec(pbeKey.getEncoded(),
                                salt,
                                iterationCount,
                                derivedKeyLength);
parameterSpec.setPrf((AlgorithmID)AlgorithmID.hMAC_SHA256.clone());
```

Alternatively you may use one of the following pre-defined PPKDF2 KeyGenerators with fixed pseudorandom function:

- PBKDF2WithHmacSHA1:
  PBKDF2 with HMAC/SHA1: `KeyGenerator.getInstance("PBKDF2WithHmacSHA1", "IAIK");`
- PBKDF2WithHmacSHA224:
  PBKDF2 with HMAC/SHA224: `KeyGenerator.getInstance("PBKDF2WithHmacSHA224", "IAIK");`
- PBKDF2WithHmacSHA256:
  PBKDF2 with HMAC/SHA256: `KeyGenerator.getInstance("PBKDF2WithHmacSHA256", "IAIK");`
- PBKDF2WithHmacSHA384:
  PBKDF2 with HMAC/SHA384: `KeyGenerator.getInstance("PBKDF2WithHmacSHA384", "IAIK");`
- PBKDF2WithHmacSHA512:
  PBKDF2 with HMAC/SHA512: `KeyGenerator.getInstance("PBKDF2WithHmacSHA512", "IAIK");`

Version:

File Revision 19

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
static class	PBKDF2.PBKDF2WithHmacSHA1 PBKDF2 key derivation function using HmacSHA1 as pseudo random function.
static class	PBKDF2.PBKDF2WithHmacSHA224 PBKDF2 key derivation function using HmacSHA224 as pseudo random function.
static class	PBKDF2.PBKDF2WithHmacSHA256 PBKDF2 key derivation function using HmacSHA256 as pseudo random function.
static class	PBKDF2.PBKDF2WithHmacSHA384 PBKDF2 key derivation function using HmacSHA384 as pseudo random function.
static class	PBKDF2.PBKDF2WithHmacSHA512 PBKDF2 key derivation function using HmacSHA512 as pseudo random function.

Constructor Summary

Constructors

Constructor and Description
PBKDF2() The default constructor

Method Summary

Methods

Modifier and Type	Method and Description
javax.crypto.SecretKey	engineGenerateKey() Derives symmetric key.
void	engineInit(java.security.spec.AlgorithmParameterSpec algorithmParameterSp, java.security.SecureRandom secureRandom) Initializes the password-based-key-derivation-function
void	engineInit(int int1, java.security.SecureRandom secureRandom) Don't use this method.
void	engineInit(java.security.SecureRandom secureRandom) Don't use this method.

Methods inherited from class java.lang.Object

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

Constructor Detail

PBKDF2

```
public  PBKDF2()
```

The default constructor

Method Detail

engineGenerateKey

```
public  javax.crypto.SecretKey  engineGenerateKey()
```

Derives symmetric key. The algorithm name is set to "RAW" and may be later changed by the calling application, if required:

```
 String algorithm = ...;
 KeyGenerator pbkdf2 = KeyGenerator.getInstance("PBKDF2", "IAIK");
 ...
 iaik.security.cipher.SecretKey secretKey = (iaik.security.cipher.SecretKey)pbkdf2.generateKey();
 secretKey.setAlgorithm(algorithm);
```

Specified by:

`engineGenerateKey` in class `javax.crypto.KeyGeneratorSpi`

Returns:

the derived key

engineInit

```
public  void  engineInit(int  int1,
                         java.security.SecureRandom  secureRandom)
```

Don't use this method. It is not implemented.

Specified by:

`engineInit` in class `javax.crypto.KeyGeneratorSpi`

engineInit

```
public  void  engineInit(java.security.SecureRandom  secureRandom)
```

Don't use this method. It is not implemented.

Specified by:

`engineInit` in class `javax.crypto.KeyGeneratorSpi`

engineInit

```
public  void  engineInit(java.security.spec.AlgorithmParameterSpec  algorithmParameterSp,
                java.security.SecureRandom  secureRandom)
                  throws java.security.InvalidAlgorithmParameterException
```

Initializes the password-based-key-derivation-function

Specified by:

`engineInit` in class `javax.crypto.KeyGeneratorSpi`

Parameters:

`algorithmParameterSp` - must be an instance of `PBEKeyAndParameterSpec`

`secureRandom` - not needed, should be `null`

Throws:

`java.security.InvalidAlgorithmParameterException`

Overview Package Class Tree Deprecated Index Help

This Javadoc may contain text parts from IETF Internet Standard specifications (see copyright note) and RSA Data Security Public-Key Cryptography Standards (PKCS, see copyright note).

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method