

Java Security

Storing Key

- A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system.
- The various components of a basic cryptosystem are Plaintext, Encryption Algorithm, Ciphertext, Decryption Algorithm, Encryption Key and, Decryption Key.
- Encryption Key is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the cipher text.
- Decryption Key is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the cipher text in order to compute the plaintext.
- Symmetric Key Encryption: The encryption process where same keys are used for encrypting and decrypting the information. Symmetric cryptosystems are also sometimes referred to as secret key cryptosystems. Examples are Digital Encryption Standard (DES), Triple-DES (3DES), IDEA, BLOWFISH.
- Asymmetric Key Encryption: The encryption process where different keys are used for encrypting and decrypting the information is known as Asymmetric Key Encryption.
- KeyStore: The Keys and certificates used/generated are stored in a data base called as keystore.
- We can access the contents of this database using the KeyStore class of the java.security package. This manages three different entries namely, PrivateKeyEntry, SecretKeyEntry, TrustedCertificateEntry..

```
import java.io.FileInputStream;
import java.security.KeyStore;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
public class Test
{
```

```
    public static void main(String args[]) throws Exception
    {
```

```
        //Creating the KeyStore object
```

```
        KeyStore keyStore = KeyStore.getInstance("JCEKS");
```

```
        //Loading the KeyStore object
```

```
        char[] password = "changeit".toCharArray();
```

```
        String path = "C:/Program Files/Java/jdk-11.0.12/lib/security/cacerts";
```

```
        java.io.FileInputStream fis = new FileInputStream(path);
```

```
        keyStore.load(fis, password);
```

```
        //Creating the KeyStore.ProtectionParameter object
```

```
        KeyStore.ProtectionParameter protectionParam = new KeyStore.PasswordProtection(password);
```

```
        //Creating SecretKey object
```

```
        SecretKey mySecretKey = new SecretKeySpec("Manish@1".getBytes(), "DSA");
```

```
        //Creating SecretKeyEntry object
```

```
        KeyStore.SecretKeyEntry secretKeyEntry = new KeyStore.SecretKeyEntry(mySecretKey);
```

```
        keyStore.setEntry("R-CAT", secretKeyEntry, protectionParam);
```

```
        //Storing the KeyStore object
```

```
        java.io.FileOutputStream fos = null;
```

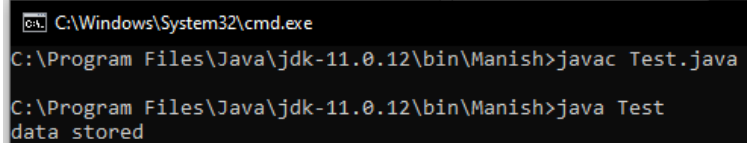
```
        fos = new java.io.FileOutputStream("newKeyStoreName");
```

```
        keyStore.store(fos, password);
```

```
        System.out.println("Key data stored");
```

```
    }
```

```
}
```



```
C:\Windows\System32\cmd.exe
C:\Program Files\Java\jdk-11.0.12\bin\Manish>javac Test.java
C:\Program Files\Java\jdk-11.0.12\bin\Manish>java Test
data stored
```

Retrieving keys

```
import java.io.FileInputStream;
import java.security.KeyStore;
import java.security.KeyStore.ProtectionParameter;
import java.security.KeyStore.SecretKeyEntry;

import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class Test
{
    public static void main(String args[]) throws Exception
    {
        //Creating the KeyStore object
        KeyStore keyStore = KeyStore.getInstance("JCEKS");

        //Loading the the KeyStore object
        char[] password = "changeit".toCharArray();
        java.io.FileInputStream fis = new FileInputStream("C:/Program Files/Java/jdk-
11.0.12/lib/security/cacerts");
        keyStore.load(fis, password);

        //Creating the KeyStore.ProtectionParameter object
        ProtectionParameter protectionParam = new KeyStore.PasswordProtection(password);

        //Creating SecretKey object
        SecretKey mySecretKey = new SecretKeySpec("Manish@1".getBytes(), "DSA");

        //Creating SecretKeyEntry object
        SecretKeyEntry secretKeyEntry = new SecretKeyEntry(mySecretKey);
        keyStore.setEntry("R-CAT", secretKeyEntry, protectionParam);

        //Storing the KeyStore object
        java.io.FileOutputStream fos = null;
        fos = new java.io.FileOutputStream("newKeyStoreName");
        keyStore.store(fos, password);

        //Creating the KeyStore.SecretKeyEntry object
        SecretKeyEntry secretKeyEnt = (SecretKeyEntry)keyStore.getEntry("R-CAT", protectionParam);

        //Creating SecretKey object
        SecretKey mysecretKey = secretKeyEnt.getSecretKey();
        System.out.println("Algorithm used to generate key : "+myscretKey.getAlgorithm());
        System.out.println("Format used for the key: "+myscretKey.getFormat());
        System.out.println("Format used for the key: "+new String(myscretKey.getEncoded()));
    }
}
```

```
C:\Program Files\Java\jdk-11.0.12\bin\Manish>java Test
Algorithm used to generate key : DSA
Format used for the key: RAW
Format used for the key: myPassword
```