Home (http://www.java2s.com)
Java (/Code/Java/CatalogJava.htm)
2D Graphics GUI (/Code/Java/2D-Graphics-GUI/Catalog2D-Graphics-GUI.htm)
3D (/Code/Java/3D/Catalog3D.htm)
Advanced Graphics (/Code/Java/Advanced-Graphics/CatalogAdvanced-Graphics.htm)
Ant (/Code/Java/Ant/CatalogAnt.htm)
Apache Common (/Code/Java/Apache-Common/CatalogApache-Common.htm)
Chart (/Code/Java/Chart/CatalogChart.htm)
Class (/Code/Java/Class/CatalogClass.htm)
Collections Data Structure (/Code/Java/Collections-Data-Structure/CatalogCollections-Data-Structure.htm)
Data Type (/Code/Java/Data-Type/CatalogData-Type.htm)
Database SQL JDBC (/Code/Java/Database-SQL-JDBC/CatalogDatabase-SQL-JDBC.htm)
Design Pattern (/Code/Java/Design-Pattern/CatalogDesign-Pattern.htm)
Development Class (/Code/Java/Development-Class/CatalogDevelopment-Class.htm)
EJB3 (/Code/Java/EJB3/CatalogEJB3.htm)
Email (/Code/Java/Email/CatalogEmail.htm)
Event (/Code/Java/Event/CatalogEvent.htm)
File Input Output (/Code/Java/File-Input-Output/CatalogFile-Input-Output.htm)
Game (/Code/Java/Game/CatalogGame.htm)
Generics (/Code/Java/Generics/CatalogGenerics.htm)
GWT (/Code/Java/GWT/CatalogGWT.htm)
Hibernate (/Code/Java/Hibernate/CatalogHibernate.htm)
I18N (/Code/Java/I18N/CatalogI18N.htm)
J2EE (/Code/Java/J2EE/CatalogJ2EE.htm)
J2ME (/Code/Java/J2ME/CatalogJ2ME.htm)
JavaFX (/Code/Java/JavaFX/CatalogJavaFX.htm)
JDK 6 (/Code/Java/JDK-6/CatalogJDK-6.htm)
JDK 7 (/Code/Java/JDK-7/CatalogJDK-7.htm)
JNDI LDAP (/Code/Java/JNDI-LDAP/CatalogJNDI-LDAP.htm)
JPA (/Code/Java/JPA/CatalogJPA.htm)
JSP (/Code/Java/JSP/CatalogJSP.htm)
JSTL (/Code/Java/JSTL/CatalogJSTL.htm)
Language Basics (/Code/Java/Language-Basics/CatalogLanguage-Basics.htm)
Network Protocol (/Code/Java/Network-Protocol/CatalogNetwork-Protocol.htm)
PDF RTF (/Code/Java/PDF-RTF/CatalogPDF-RTF.htm)
Reflection (/Code/Java/Reflection/CatalogReflection.htm)
Regular Expressions (/Code/Java/Regular-Expressions/CatalogRegular-Expressions.htm)
Scripting (/Code/Java/Scripting/CatalogScripting.htm)
Security ()
Servlets (/Code/Java/Servlets/CatalogServlets.htm)
Spring (/Code/Java/Spring/CatalogSpring.htm)
Swing Components (/Code/Java/Swing-Components/CatalogSwing-Components.htm)
Swing JFC (/Code/Java/Swing-JFC/CatalogSwing-JFC.htm)
SWT JFace Eclipse (/Code/Java/SWT-JFace-Eclipse/CatalogSWT-JFace-Eclipse.htm)
Threads (/Code/Java/Threads/CatalogThreads.htm)
Tiny Application (/Code/Java/Tiny-Application/CatalogTiny-Application.htm)
Velocity (/Code/Java/Velocity/CatalogVelocity.htm)
Web Services SOA (/Code/Java/Web-Services-SOA/CatalogWeb-Services-SOA.htm)
XML (/Code/Java/XML/CatalogXML.htm)
Menu

Search

Triple DES : DES « Security « Java

Triple DES

```java
/*
 * Copyright (c) 2000 David Flanagan.  All rights reserved.
 * This code is from the book Java Examples in a Nutshell, 2nd Edition.
 * It is provided AS-IS, WITHOUT ANY WARRANTY either expressed or implied.
 * You may study, use, and modify it for any non-commercial purpose.
 * You may distribute it non-commercially as long as you retain this notice.
 * For a commercial use license, or to purchase the book (recommended),
 * visit http://www.davidflanagan.com/javaexamples2.
 */

import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.Provider;
import java.security.Security;
import java.security.spec.InvalidKeySpecException;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESedeKeySpec;

/**
 * This class defines methods for encrypting and decrypting using the Triple DES
 * algorithm and for generating, reading and writing Triple DES keys. It also
 * defines a main() method that allows these methods to be used from the command
 * line.
 */
public class TripleDES {
  /**
   * The program. The first argument must be -e, -d, or -g to encrypt,
   * decrypt, or generate a key. The second argument is the name of a file
   * from which the key is read or to which it is written for -g. The -e and
   * -d arguments cause the program to read from standard input and encrypt or
   * decrypt to standard output.
   */
  public static void main(String[] args) {
    try {
      // Check to see whether there is a provider that can do TripleDES
      // encryption. If not, explicitly install the SunJCE provider.
      try {
        Cipher c = Cipher.getInstance("DESede");
      } catch (Exception e) {
        // An exception here probably means the JCE provider hasn't
        // been permanently installed on this system by listing it
        // in the $JAVA_HOME/jre/lib/security/java.security file.
        // Therefore, we have to install the JCE provider explicitly.
        System.err.println("Installing SunJCE provider.");
        Provider sunjce = new com.sun.crypto.provider.SunJCE();
        Security.addProvider(sunjce);
      }

      // This is where we'll read the key from or write it to
      File keyfile = new File(args[1]);
      // Now check the first arg to see what we're going to do
      if (args[0].equals("-g")) { // Generate a key
        System.out.print("Generating key. This may take some time...");
        System.out.flush();
        SecretKey key = generateKey();
        writeKey(key, keyfile);
        System.out.println("done.");
        System.out.println("Secret key written to " + args[1]
            + ". Protect that file carefully!");
      } else if (args[0].equals("-e")) { // Encrypt stdin to stdout
        SecretKey key = readKey(keyfile);
        encrypt(key, System.in, System.out);
      } else if (args[0].equals("-d")) { // Decrypt stdin to stdout
        SecretKey key = readKey(keyfile);
        decrypt(key, System.in, System.out);
      }
    } catch (Exception e) {
      System.err.println(e);
      System.err.println("Usage: java " + TripleDES.class.getName()
          + " -d|-e|-g <keyfile>");
    }
  }

  /** Generate a secret TripleDES encryption/decryption key */
  public static SecretKey generateKey() throws NoSuchAlgorithmException {
    // Get a key generator for Triple DES (a.k.a DESede)
    KeyGenerator keygen = KeyGenerator.getInstance("DESede");
    // Use it to generate a key
    return keygen.generateKey();
  }

  /** Save the specified TripleDES SecretKey to the specified file */
  public static void writeKey(SecretKey key, File f) throws IOException,
      NoSuchAlgorithmException, InvalidKeySpecException {
    // Convert the secret key to an array of bytes like this
    SecretKeyFactory keyfactory = SecretKeyFactory.getInstance("DESede");
    DESedeKeySpec keyspec = (DESedeKeySpec) keyfactory.getKeySpec(key,
        DESedeKeySpec.class);
    byte[] rawkey = keyspec.getKey();

    // Write the raw key to the file
    FileOutputStream out = new FileOutputStream(f);
    out.write(rawkey);
    out.close();
  }

  /** Read a TripleDES secret key from the specified file */
  public static SecretKey readKey(File f) throws IOException,
      NoSuchAlgorithmException, InvalidKeyException,
      InvalidKeySpecException {
    // Read the raw bytes from the keyfile
    DataInputStream in = new DataInputStream(new FileInputStream(f));
    byte[] rawkey = new byte[(int) f.length()];
    in.readFully(rawkey);
    in.close();

    // Convert the raw bytes to a secret key like this
    DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
    SecretKeyFactory keyfactory = SecretKeyFactory.getInstance("DESede");
    SecretKey key = keyfactory.generateSecret(keyspec);
    return key;
  }

  /**
   * Use the specified TripleDES key to encrypt bytes from the input stream
   * and write them to the output stream. This method uses CipherOutputStream
   * to perform the encryption and write bytes at the same time.
   */
  public static void encrypt(SecretKey key, InputStream in, OutputStream out)
      throws NoSuchAlgorithmException, InvalidKeyException,
      NoSuchPaddingException, IOException {
    // Create and initialize the encryption engine
    Cipher cipher = Cipher.getInstance("DESede");
    cipher.init(Cipher.ENCRYPT_MODE, key);

    // Create a special output stream to do the work for us
    CipherOutputStream cos = new CipherOutputStream(out, cipher);

    // Read from the input and write to the encrypting output stream
    byte[] buffer = new byte[2048];
    int bytesRead;
    while ((bytesRead = in.read(buffer)) != -1) {
      cos.write(buffer, 0, bytesRead);
    }
    cos.close();

    // For extra security, don't leave any plaintext hanging around memory.
    java.util.Arrays.fill(buffer, (byte) 0);
  }

  /**
   * Use the specified TripleDES key to decrypt bytes ready from the input
   * stream and write them to the output stream. This method uses uses Cipher
   * directly to show how it can be done without CipherInputStream and
   * CipherOutputStream.
   */
  public static void decrypt(SecretKey key, InputStream in, OutputStream out)
      throws NoSuchAlgorithmException, InvalidKeyException, IOException,
      IllegalBlockSizeException, NoSuchPaddingException,
      BadPaddingException {
    // Create and initialize the decryption engine
    Cipher cipher = Cipher.getInstance("DESede");
    cipher.init(Cipher.DECRYPT_MODE, key);

    // Read bytes, decrypt, and write them out.
    byte[] buffer = new byte[2048];
    int bytesRead;
    while ((bytesRead = in.read(buffer)) != -1) {
      out.write(cipher.update(buffer, 0, bytesRead));
    }

    // Write out the final bunch of decrypted bytes
    out.write(cipher.doFinal());
    out.flush();
  }
}
```

Related examples in the same category

1. DES Crypter and Decrypter (/Code/Java/Security/DESCrypterandDecrypter.htm) (/Code/Java/Security/DESCrypterandDecrypter.htm)
2. Decrypt an object with DES (/Code/Java/Security/DecryptanobjectwithDES.htm)
3. Encrypt an object with DES (/Code/Java/Security/EncryptanobjectwithDES.htm)
4. Encrypting a String with DES (/Code/Java/Security/EncryptingaStringwithDES.htm)
5. Encrypting an Object with DES (/Code/Java/Security/EncryptinganObjectwithDES.htm)
6. Encrypting a File or Stream with DES (/Code/Java/Security/EncryptingaFileorStreamwithDES.htm)
7. Encrypting with DES Using a Pass Phrase (/Code/Java/Security/EncryptingwithDESUsingaPassPhrase.htm)
8. DES Engine (/Code/Java/Security/DESEngine.htm)
9. DES algorithm (/Code/Java/Security/DESalgorithm.htm)
10. Des Encrypter (/Code/Java/Security/DesEncrypter.htm)
11. DES Decrypt (/Code/Java/Security/DESDecrypt.htm)
12. DES Encrypt (/Code/Java/Security/DESEncrypt.htm)