le cnam

S. Bouzefrane

Exercise on Java Card 2.x

The objective of this Lab is to encrypt and decrypt the exchanged data between the terminal and the cardlet using Java Card 2 technology.
The ciphering algorithm that is use is triple DES because it is the only algorithm implemented on our cards. The algorithm is based on a key of 192 bits (=24 bytes) and blocs of 64 bits (8 bytes) with an initialization vector that has the same size as the blocs.

Let's consider the Eclipse environment starting from the examples of Echo Client and Echo applet. You have to complete the Echo programs with the appropriate instructions.

Question 1: Complete the client to perform the following actions:
- Send a clear data of 8 bytes to the applet. This data will be encrypted by the applet and sent to the terminal that decrypts it and displays it.
- Send an encrypted data to the applet that decrypts it and returns it as a clear text. This data has to be displayed by the terminal.

Follow the different steps to design the client program:

a. Define the APDU commands,
b. Import the following packages:
```
import javax.crypto.*;
import javax.crypto.spec.DESedeKeySpec;
import javax.crypto.spec.IvParameterSpec;
```
c. Declare an array of 8 bytes initialized to an hexadecimal value that corresponds to the data value that will be sent by the client to the applet.
d. Declare a byte array that has the size of the secret key and that contains the key value used by the triple DES algorithm.
e. Declare the initialization vector as an array of 8 bytes initialized to 0 values because the 3DES algorithm implemented on our card is based on a vector equal to 0 due to the memory limitation.
f. Each encryption or decryption operation requires some steps as in the following. This link will help you find the syntax of the involved methods:
http://docs.oracle.com/javase/1.4.2/docs/api/index.html (choose javax.crypto)

  - Create the object *DESedeKeySpec* that requires the secret key that has been already defined,
  - Create a factory of secret keys with the static class *SecretKeyFactory* for the 3DES algorithm called here "DESede" for DES Encryption-Decryption-Encryption,
  - Generate the secret key with the method *generateSecret*(),
  - Create an object of type *IvParameterSpec* corresponding to the initialization vector,

- Create an object Cipher while indicating a variant of 3DES, called "DESede/CBC/NoPadding", that means DESede algorithm using blocks of (Cipher Block Chaining) of 64 bits with no padding.
- Initialize the Cipher object by indicating the operation type to perform. The initialization requires three parameters:
  1. Type of operation: is a constant equal to Cipher.ENCRYPT_MODE in case of encryption or Cipher.ENCRYPT_MODE if decryption.
  2. The secret key that has been generated
  3. And the initialization vector.

g. The encryption/decryption operation is acheived thanks to the *doFinal*() method.

Question 2: Complete the Java Card applet so that it can deal with the APDU commands sent by the terminal.

For this purpose, you have to follow the different steps:

h. Import the following packages:
```
import javacard.security.DESKey;
import javacard.security.Key;
import javacard.security.KeyBuilder;
import javacardx.crypto.Cipher;
```

i. Declare the same secret key as for the client.

j. Declare a byte array that will be used to store the data received from the client. Let's notice that you will need a method to copy a byte array to another, as in the following: Util.arrayCopyNonAtomic(source, index of the source, destination, index of destination, size of data to copy) ;

k. According to the APDU command, the received data will be encrypted or decrypted before returning it to the terminal.

l. Whatever the operation is encryption or decryption, the following steps have to be performed. The following link to the Java Card API is useful: http://www.win.tue.nl/pinpasjc/docs/apis/jc222/index.html (choose javacardx.crypto)

- Create an object from static class Cipher while indicating the variant of the algorithm. Here the algorithm is a 3DES of type CBC No padding. We do it with the constant Cipher.ALG_DES_CBC_NOPAD as a first parameter, and false as a second parameter.
- Then, create an object DESKey to build the key with KeyBuilder.buildKey() and comme paramètres KeyBuilder.TYPE_DES, KeyBuilder.LENGTH_DES3_3KEY et false.
- la méthode setKey(clé, offset) permet d'initialiser l'objet DESKey avec la clé.
- La méthode init() de l'objet Cipher permet de fixer l'opération à effectuer : chiffrement si le deuxième paramètre est égal à Cipher.MODE_ENCRYPT et déchiffrement si le deuxième paramètre est égal à Cipher.MODE_ DECRYPT. Le premier paramètre étant l'objet DESKey correspondant à la clé secrète.
- Le chiffrement/déchiffrement est effectivement réalisée à l'aide de la méthode doFinal() de l'objet Cipher.

Client ====== clear text =====> Applet (executes the method of data encryption)

Client < ========== encrypted data======Applet
Decrypts the data
and displays it

Client =======encrypted data ======> Applet (executes the method of data decryption)
Client <======= clear text ====== Applet
Displays the data