Application Security – Lab Session 1
Java Security

sadry@orange.fr

Instructions

In this lab session, you will have to answer questions in this document, and complete two java projects: 3DES and RSA.

You can answer in English or French to the questions, with complete sentences.
You can work in a team of 2 students (not more).

This document has to be completed, zip it together with your JAVA projects and send it to sadry@orange.fr as follows:

- Zip it and rename it to **LabSession1-FirstNameStudent1-FirstNameStudent2.ZIP**
- Send it to sadry@orange.fr with following subject: **[Application Security] – LabSession1**
- Put in CC the second student if any.

This lab session will be evaluated on 20, and will count for 25% of the final Application Security grade.

Application Security – Lab Session 1
Java Security

sadry@orange.fr

Application Security – Lab Session 1
Java Security

sadry@orange.fr

3DES

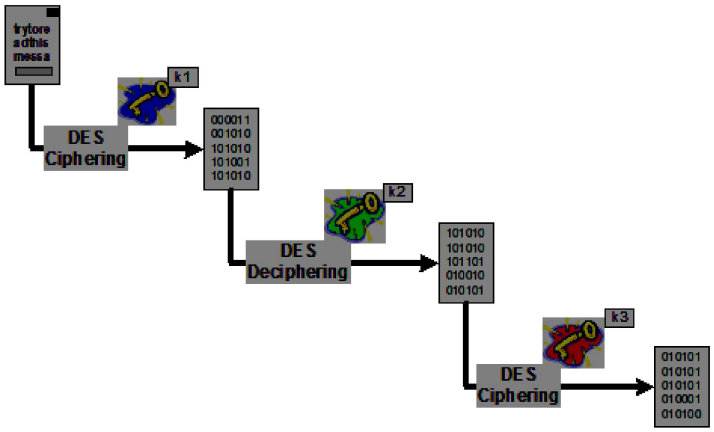
(10 points)

In this exercise, you have to answer the questions in this document, and complete the file `./src/com/polytech/security/TripleDES.java`.

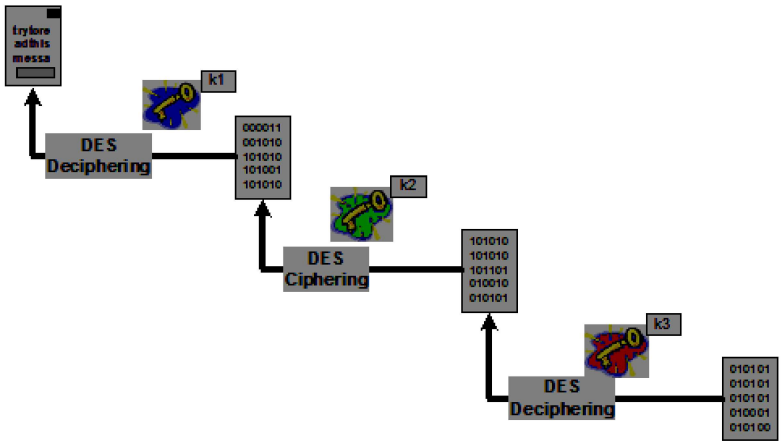
Background information

3DES is a symmetric key block cipher, which applies three times the DES cipher/decipher algorithm.

3DES encryption is performed as depicted in the following figure:



3DES decryption is performed as depicted in the following figure:



Several keying configurations are possible with 3DES.

Application Security – Lab Session 1
Java Security

sadry@orange.fr

In this lab session, K1, K2 and K3 will be independent.

Application Security – Lab Session 1
Java Security

sadry@orange.fr

1. Data Encryption Standard (1 point)

What is the size of DES cipher key's size ? (0.5 point)

DES est composé de 8*8 bits mais on utilise un bit de parité on a donc 8*7 = 56 bits pour la clé

Dans notre cas k1, k2 et k3 sont indépendants donc on a 3*56bits de clé soit 168 bits pour le 3DES

What are the size of the cipher blocks ? (0.5 point)

Sipher blocks = 8*8 = 64 bits

Application Security – Lab Session 1
Java Security

sadry@orange.fr

2. DES/CBC/NoPadding (4,5 points)

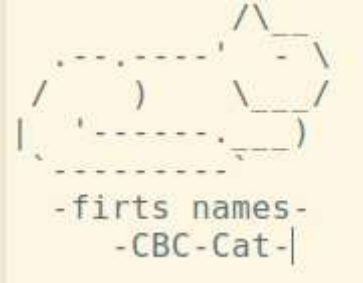
2.1. Explain CBC with diagram (0,5 point)

cf.ApplicationSecurity_LabSession1/CBC_encryption.svg.png

Cipher block chaining --> chaque bloc chiffré agit sur le suivant avant son chiffrement grâce à une opération XOR entre yi et xi+1.

2.2. Explain NoPadding (0,5 point)

For the next two questions, you are asked to encrypt the following file
./ebc/clearTextFileEBC
You will also have to add (before encrypting) your firsts names as in the following image



The final document must contains :

- all the scripts
- jar files : 3DES.jar, RSA.jar and Alice_Bob.jar
- readme.txt file containing the how to

2.3. Perform 3DES in EBC mode encryption in
`TripleDES ::private Vector encryptEBC(...)` (1,75 points)

2.4. Perform 3DES in EBC mode decryption in
`TripleDES ::private void decryptEBC(...)` (1,75 points)

Application Security – Lab Session 1
Java Security

sadry@orange.fr

2. DES/EBC/NoPadding

2.1. Explain EBC and its advantages over CBC with diagram (1 point)

cf. ApplicationSecurity_LabSession1_2022/ApplicationSecurity_LabSession1/ECB_encryption.png

For the two following questions, you are asked to encrypt the following file :
`./cbc/clearTextCBC`

2.3. Perform 3DES in CBC mode encryption in
`TripleDES ::private Vector encryptCBC(...)` (2 points)
2.4. Perform 3DES in CBC mode decryption in
`TripleDES ::private void decryptCBC(...)` (2 points)

2.1 : Dans un EBC chaque bloc est indépendant. On peut donc alors parallélisé pour crypter plusieurs blocs simultanément. Il est aussi plus robuste à la corruption de données puisque chaque bloc est indépendant, on pourra donc toujours décrypter les blocs non corrompu.

Application Security – Lab Session 1
Java Security

sadry@orange.fr

sadry@orange.fr

RSA Signature Implementation

(10 points)

In this exercise, you have to answer to the questions into this document, and complete the file *./src/com/polytech/security/Asymetric.java* and *./src/com/polytech/security/Entity.java*

1. Use of Java Signature (3 points)

1.1. Generation of a public/private key pair (1 point)

Complete method *Entity::Entity()*

- Generate a keypairgenerator object of type `java.security.KeyPairgenerator` for RSA.
- Generate a keypair public/private.
- Store them in class members `Entity::thePublicKey` and `Entity::thePrivateKey`.

1.2. RSA Signature (1 point)

Complete method *Entity::sign()*

- Create an signature object *java.security.signature* for « `SHA1withRSA` ».
- Initialize the object with the private key in *SIGN_MODE*.
- Sign

1.3. Check signature (1 point)

Complete method *Entity::checkSignature()*

- Create an object *java.security.Signature*
- Initialize it in *VERIFY* mode with the public key
- Check the signature.

Application Security – Lab Session 1
Java Security

sadry@orange.fr

sadry@orange.fr

2. Implementation of your own RSA signature (5 points)

2.1. Signature (2.5 points)

Complete method *`Entity::mySign()`*
Implement your own signature using

- *`javax.crypto.Cipher`* with *`RSA`* in *`ENCRYPT_MODE`* mode
- *`java.security.MessageDigest`* with *`SHA1`*.

2.2. Check signature (2.5 points)

Complete method *`Entity::myCheckSignature()`*
Implement your own signature verification using

- *`javax.crypto.Cipher`* with *`RSA`* in *`DECRYPT_MODE`* mode
- *`java.security.MessageDigest`* with *`SHA1`*

3. RSA Ciphering (2 points)

Warning : RSA implementation by SUN does not support message greater than 127 bytes.

3.1. RSAEncryption (1 point)

Complete method *`Entity::encrypt()`*

3.2. RSADecryption (1 point)

Complete method *`Entity::decrypt()`*.

sadry@orange.fr

---

3. Secure session key exchange (2 points)

You have to implement the following protocol between Alice and Bob for a secure session key exchange.

1. Alice sends her public key to Bob.
2. Bob generates a DES session key.
3. Bob encrypts it with Alice's public key.
4. Alice decrypts the DES key with her private key.
5. Alice sends a message to Bob with her session key
6. Bob decrypts the message with the session key.

Please fill in the static method ***KeyExchangeProtocol()*** in ***Asymetric.java***.

You can also refer to the slides from the application security lecture for further details on this secure session key exchange.