COE817 Lab 3

2 Weeks lab

To be done in group of two

Objectives

- In this lab, you will familiar yourself with Java cryptography class libraries and work on a secure chat system. This lab is bigger than the previous, please start early. Two students will work as a group. Netbeans is the IDE.
- **Duration: two week.**

Required Exercise

To get started you must perform the following operations.

Create your course directory with the command:

mkdir coe817

1. Change to your course directory with the command:
   cd coe817
2. Create your lab1 sub-directory with the command:
   mkdir lab3
3. Change to your lab1 directory with the command:
   cd lab3

Exercise 1

Goals for this exercise:

1. Familiar yourself with Java cryptography class libraries
2. Program with DES and RSA algorithms

In exercise 1, you need to

(1) Design a class that encrypts a message with the DES algorithm provided by the Java cryptography class libraries. Name your new class "JEncrypDES**".**

Write a `main()` method in class JEncrypDES to do the following:

1. Ask the user to enter the message "No body can see me".
2. Generate a DES key
3. Encode the message using DES.
4. Print the encoded text.
5. Decode the message using DES.

6. Print the result of decoding the encoded text (which should be the original text).

(2) Design a class that encrypts a message with RSA algorithm provided by the Java cryptography class libraries. Name your new class "JEncrypRSA**".**

Write a `main()` method in class JEncrypRSA to do the following:

1. Ask the user to enter the message "No body can see me".
2. Generate the public and private keys
3. Encode the message using the public key.
4. Print the encoded text.
5. Decode the message using the private key.

Java Cryptography Class Libraries

Java's security and cryptography classes are divided into two main packages: java.security.* and javax.crypto.*. The reason for this division is U.S. export restrictions. Everything in java.security.* is exportable, most everything in javax.crypto.* is not. The javax.crypto.* package is collectively referred to as the Java Crypto Extensions or the JCE. Classes for cryptographic hashing and digital signatures can be found in security, whereas ciphers and MACs are located in the JCE.

The Java platform includes built-in providers for many of the most commonly used cryptographic algorithms, including the RSA, DSA, and ECDSA signature algorithms, the DES, AES, and ARCFOUR encryption algorithms, the MD5, SHA-1, and SHA-256 message digest algorithms, and the Diffie-Hellman and ECDH key agreement algorithms.

A good documentation site is:

http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html

Exercise 2

For exercise 2, you need to write a Java socket communication program to implement a secure chat system. The required security features are:

- An authenticated key distribution between the chat client and the chat server.
- All chat messages need to be encrypted using DES
- You must devise and implement a measure to combat replay attacks.

We will examine each of these features in detail below.

Key Distribution

We will use the following public key based key distribution scheme that has been explained in class.

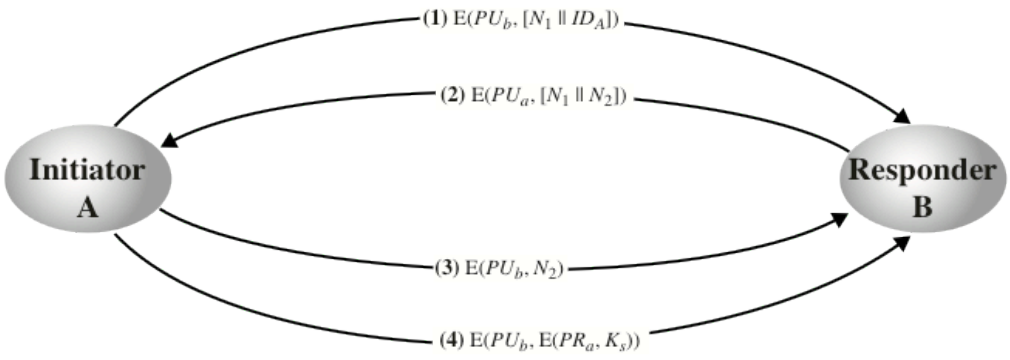

Figure 14.9 Public-Key Distribution of Secret Keys

Chat Message Encryption

Each message transmitted by either a client or the server must be encrypted using DES.

Chat message could be a text or an image. Client/server must be able to encrypt a text or an image and sends to the client/server.

Preventing Replay Attacks

Even after you secure chat messages with encryption, there is still an obvious replay attack. For example, an attacker captures a chat message. He then repeatedly sends that message – which is still a valid chat message – flooding the intended recipient and making the chat room unusable for the participants. Your solution should prevent an attacker from replaying a user's message.

Implementation

The chat client initiates a connection with the chat server. Upon connection, it will implement the key distribution protocol with the chat server. A session key Ks will be distributed and shared by both client and server.

The chat client and chat server will then encrypt texts or image using Ks.

Submission

Design your UI (user interface) for the secure chat solution. Your UI must allow the TA to see what message is actually sent and received over the wire at each point in the communication processes. Both chat client and server must show the following messages:

the cleartext of text or image that are sent
the received cipertext of text or image
the received and decrypted text or image

Make screenshots of your implementation on both chat client and chat server. Your screen shots will show the exchanged key distribution messages, the encrypted and decrypted chat messages (including texts and images)

Create a word file mysubmission.doc including your student names, ids (all members of your group), the screenshots and explain how to prevent replay attacks in your solution. Save the word file under your lab3 directory.

Submitting your lab

If you did the lab on a Departmental computer, you can do the following:

```
cd coe817
zip -r lab3.zip lab3
submit coe817 lab3 lab3.zip
```

If you did the lab on your own computer, zip the lab3 folder (remember to do this recursively so that all sub-folders are included), then transfer the zip file to a Departmental machine, logon to a Departmental machine which can be done remotely) and type in the submit command:

```
submit coe817 lab3 lab3.zip
```