02230: Computer Security
Lab3: Cryptography in Practice

Bojan Pajkovski

Autumn 2005

The goal of this lab is to provide hands-on experience with cryptography and cryptographic techniques using Java. Your task would be do develop a simple client/server application using Java, which requires you to do some Java programming.

1 Crypto Client/Server Application

The application developed for this exercise consists of two simple programs: a `CryptoClient` and a `CryptoServer`. These parties communicate using Java Sockets, and this communication is encrypted using symmetric cryptography, which preserves confidentiality of the messages sent and received by both parties. This requires both parties to share the same symmetric key. For the sake of simplicity, both `CryptoServer` and `CryptoClient` programs are placed in the same directory, and the shared key is generated by one of the parties, in this case the `CryptoServer`.

In these programs, we use DES as a symmetric key algorithm to encrypt and decrypt messages.

Both `CryptoClient` and a `CryptoServer` consist of three methods:

`cryptoSetup` – Set up the cryptographic environment for the given application. This is the method, where all necessary keys are generated and saved on the disk (this is performed only once), or loaded from the disk.

`encryptMsg` – Encrypt a plaintext message and in return obtain a ciphertext message.

`decryptMsg` – Decrypt the ciphertext message and in return obtain the plaintext message.

`sendAndReceive()` – Establish an encrypted communication channel with the other party using Java Sockets.

2 Task

For this lab, you are given this crypto client/server application. The first task consists in making the appropriate changes in both `CryptoClient.java` and `CryptoServer.java`, so that they use the AES cryptosystem instead of DES.

The second task requires you to write two methods: `signMsg` and `verifyMsg` in both `CryptoClient` and `CryptoServer`. The `signMsg` method should create a signature to a message before it is sent to the other party, and the signature should be verified using `verifyMsg` method.

3 Evaluation

You will have two hours to make the necessary changes in both `CryptoClient` and `CryptoServer` programs. An evaluation sheet will be available for download at the beginning of the lab. After successful demonstration of your program, one of the demonstrators will sign your evaluation sheet and collect it for latter use. There will be no formal feedback for this lab.

4 Getting Started

Download the `CryptoClient` and `CryptoServer` as a ZIP-file from
`http://www.imm.dtu.dk/courses/02230/labs/BouncyCastle/crypto-1.0.zip` and unzip the files in your user home directory.

The current implementation of Sun's Java Development Kit 1.5 supports only a limited number of cryptosystems. Therefore, in this lab we will use the cryptographic provider from the Legion of the Bouncy Castle. Hence, you would have to download the Bouncy Castle crypto provider (for Java Development Kit 1.5 or other version that suits you the most) from `http://www.bouncycastle.org/download/bcprov-jdk15-130.jar` and include this library as an "external library" in your project.

NOTE: In both client and server programs you would have to add the Bouncy Castle provider in the list of security providers in Java, before you could actually use it.

5 Useful Links

Java Development Kit 1.5.0 API Specifications
`http://java.sun.com/j2se/1.5.0/docs/api/`

Bouncy Castle JDK 1.5.0 API Specifications
`http://www.bouncycastle.org/docs/docs1.5/`