National University of Singapore
School of Computing

CS2105 **Assignment 3** Semester 2 AY14/15

Submission Deadline

12 April, 2015 (Sunday), 6pm sharp. No late submission will be accepted.

Learning Objective

In this assignment you will have a taste of Java APIs for cryptography. This programming assignment is worth 6 marks and shall be completed **individually**.

Writing Your Programs

You are free to write your program on any platform/IDE that you are familiar with.

However, you are responsible to ensure that your program runs properly on **`sunfire` server** because **we will test and grade your program on `sunfire`.**

Program Submission

Please submit your program to **`CodeCrunch`**: https://codecrunch.comp.nus.edu.sg. You just need to submit **`Alice.java`**. We will use our own **`Bob.java`** (which is the same as the one given to you) for testing. If you split **`Alice.java`** into several programs, submit all of them to **`CodeCrunch`** simultaneously.

Note that **`CodeCrunch`** is just used for program submission and no test case has been mounted on it.

Grading

Your programs will be graded according to their correctness using a grading script:

- **[1 point]** **`Alice.java`** is compilable on **`sunfire`**.
- **[1 point]** **`Alice.java`** follows specified **`Java`** command to launch (see section below).
- **[4 points]** **`Alice.java`** correctly decrypts messages and save them to a file. Grading script doesn't care what messages your program prints on the screen. It just checks if the generated file is identical to the one sender (Bob) has.

Introduction

Bob wants to send a confidential document to Alice. Here is how they will do it.

1. Alice keeps a copy of Bob's RSA public key. She generates an AES session key and encrypts it with Bob's public key. Alice then sends this encrypted session key to Bob. Bob uses his RSA private key to decrypt it and get the session key.
2. The confidential document Bob keeps is a text file of exactly 10 lines. Bob sends it to Alice over **TCP** using the following algorithm:

```
for counter i from 1 to 10
        read a line and encrypt it with AES session key
        send encrypted message as a SealedObject to Alice
```

3. Alice receives encrypted messages, decrypted them and save plaintext to file.

Skeleton Programs

Bob.java is complete and given to you. Your task in this assignment is to complete the given skeleton program **Alice.java** so that Alice can (1) successfully generate and send session key to Bob, (2) receive all messages from Bob, decrypt and save them to a file. The file Alice generates should be exactly the same as the confidential document Bob keeps.

Bob.java

The **Bob.java** program is complete and shouldn't be changed. To run it, type command:

```
java Bob <BobPort>
```

For example:

```
java Bob 9000
```

listens at port 9000 and waits for Alice to connect.

Upon startup, Bob will read his RSA private key from the file **private.key** stored in the same directory as **Bob.java**.

Bob runs a TCP server. Once connected by Alice, Bob first receives an encrypted session key from Alice and decrypts it with RSA private key. He then sends the file **docs.txt** line by line to Alice, each line encrypted with the session key.

You should read **Bob.java** carefully because most of the codes you are supposed to write in **Alice.java** have counterparts (i.e. reference) in **Bob.java**.

Note that **Bob.java** and **Alice.java** should be placed in two directories.

Alice.java

To run Alice, type command:

```
java Alice BobIP BobPort
```

For example:

```
java Alice sunfire.comp.nus.edu.sg 9000
```

connects to Bob running on **sunfire.comp.nus.edu.sg** at port 9000.

Upon startup, Alice will read Bob's RSA public key from the file **public.key** stored in the same directory as **Alice.java**.

Alice runs a TCP client. When connected to Bob, Alice generates an AES key, encrypts it with Bob's public key and sends it Bob. She then receives 10 encrypted messages from Bob, decrypts and saves them to the file **msgs.txt** in the same directory as **Alice.java**.

Note that Bob's RSA public and private keys remain unchanged across sessions. Each time **Alice.java** launches, a new AES key should be generated and sent to Bob.

Useful Java Cryptography Classes

Java **Cipher** class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework. A **Cipher** object needs to be initialized to **ENCRYPT_MODE** or **DECRYPT_MODE** before respective operations. Encryption algorithms such as RSA, DES and AES are supported (there is no need to study the feedback mode and padding scheme of these algorithms).

Bob encapsulates (and encrypts) messages as Java **SealedObject**s before transmission. **SealedObject**s are containers which encrypt and decrypt their contents (objects) with the help from a **Cipher** object. The reason we use **SealedObject**s is that it is very easy to transmit objects over TCP. An alternative approach is to transmit byte arrays over TCP (e.g., **doFinal** method of **Cipher** class actually returns encrypted message as a byte array). However, we will stick to using **SealedObject** in this assignment.

To generate an AES key, Alice should use **KeyGenerator** class to generate an AES **SecretKey** object. Alice also encapsulates AES key as a **SealedObject** before sending it to Bob. However, sealing an AES **SecretKey** object with RSA presents a problem because RSA algorithm imposes a size restriction on the object being encrypted (typically 117 bytes). An AES **SecretKey** object is too large for RSA encryption. The way out is to use RSA to seal the "encoded form" of an AES **Key** object (encoded form can be retrieved using **getEncoded** method of **SecretKey** class). Bob will restore AES **Key** object from the received code.

You may need to read relevant Java API documentation for more information on aforementioned Java classes. You are also suggested to study **Bob.java** program for example usage. There is no need to use other Java cryptography classes not imported in the given skeleton (using other classes implies you are taking a complex approach).

Plagiarism Warning

You are free to discuss this assignment with your friends. But, ultimately, you should write your own code. We employ zero-tolerance policy against plagiarism. If a suspicious case is found, student would be asked to explain his/her code to the evaluator in face. Confirmed breach may result in zero mark for this assignment and further disciplinary action from the school.

Question & Answer

If you have any doubts on this assignment, please post your question on IVLE forum or consult the teaching team.