

# COMP3441/9441 Assignment 2

Due Fri Nov. 7, 2003

## Submission Instructions

- Your submission should include your name and student number in the beginning of every file. Please provide both your surname/family name and your given name. Also, each file should state clearly that the submission is for Assignment 2, Session 2, 2003.
- Submit your assignment using the `give` command. Further instructions will be available closer to the due date.
- This assignment is worth 25% of your total mark for the subject. The assignment will be marked out of a total of 25 marks. Late submissions will be penalized by deducting 5 marks from your score for each day **after Nov. 14**.
- **Plagiarism:** You are reminded of the school's policy on plagiarism (see <http://www.cse.unsw.edu.au/school/facilities/yellowform.html>). You may discuss the *general approach* to solving a problem with other students, but your specific solutions on this assignment must be your own work. You must cite the source of any other author's material that you include. Anyone found to have plagiarized on any part of this assignment will be awarded 0 marks for the whole assignment.

## Assignment Specification

Alice and Bob would like to play the coin tossing game “two-up” online. To do so, they would like a trustworthy source of randomness, that ensures to both parties that the outcomes of coin flips are indeed random, and cannot be biased by either party. To achieve this, they decide to use a public-key based fair coin flipping protocol, described in B. Schneier, *Applied Cryptography*, 2nd ed, section 4.10, pp 89-91. If you do not have this book, it is available in Open Reserve in the University library.

The protocol goes as follows. Its uses the fact that RSA is commutative.

1. Alice and Bob each generate a public-key/private-key key pair.
2. Alice generates two messages, one indicating heads and the other indicating tails. These messages should contain some unique random string, so that she can verify their authenticity later in the protocol. Alice encrypts both messages with her public key and sends them to Bob in random order.

$$E_A(M_1), E_A(M_2)$$

3. Bob, who cannot read either message, chooses one at random. He encrypts it with his public key and sends it back to Alice.

$$E_B(E_A(M)), \text{ where } M \text{ is either } M_1 \text{ or } M_2$$

4. Alice, who cannot read the message sent back to her, decrypts it with her private key and then sends it back to Bob.

$$D_A(E_B(E_A(M))) = E_B(M_1) \text{ if } M = M_1, \text{ or } E_B(M_2) \text{ if } M = M_2$$

5. Bob decrypts the message with his private key to reveal the result of the coin flip. He sends the decrypted message to Alice.  
 $D_B(E_B(M_1)) = M_1$  or  $D_B(E_B(M_2)) = M_2$
6. Alice reads the result of the coin flip and verifies that the random string is correct.
7. Both Alice and Bob reveal their key pairs so that both can verify that the other did not cheat.

The assignment is to implement this protocol in Java, using the Java cryptography package. Note that you do *not* need to implement the “two-up” game.

Your implementation should define two classes, one for Alice and one for Bob, with each of the steps of the protocol corresponding to method call on the objects. More precisely, your implementation should fill out the class structure in the files `HonestAlice.java` and `HonestBob.java` available on the course web page. These classes are defined by the interfaces in the files `Alice.java` and `Bob.java`. Remember that you will submit *only* `HonestAlice.java` and `HonestBob.java`, so you should not rely on any changes you make to any other files. Of course, you can still add code to other classes to help you debug, but you will not be able to submit those other files.

To run the protocol, use the `CoinFlip` class. You may want to annotate this class with `println` commands to show the progress. Your code will be tested by running *different* implementations for each of Alice and Bob against your code, in an attempt to induce a statistically significant deviation from 0.5/0.5 statistics for the coin flip outcome. Your code will also be run against parties that try to cheat by sending incorrect messages, so the verification aspects of the protocol should be correctly implemented.

You must use the RSA cipher in this assignment. Unfortunately, Sun’s JDK does not come with the RSA cipher, so you need to download and install a provider that implements RSA. Instructions for this is available on the subject web page.

If you have any questions regarding the assignment, please proceed in the following manner:

1. Consult the assignment FAQ, available from the subject web page.
2. Consult the Java resources available via the links on the subject web page.
3. Email to `halvards.cs3441@cse.unsw.edu.au`. If your questions has not been answered in the FAQ, and if the answer may not be easily found in the resources, you will get a reply within 48 hours (excluding weekends).

Please be aware that questions of the type “How do I compile a Java program?” or “Can you debug my code?” will be ignored.