CO7099 Assignment 01
Marking Criteria

Released Jan 30, 2025

Deadline Friday Feb 21, 2025 12:00 noon

Marks for this assignment are divided as follows:

- Execution testing: 25 marks
- Code inspection: 70 marks
- Readability: 5 marks

The following is a more detailed description of each component. They might need to be slightly adapted if "unexpected" situations happen.

Execution testing (25 marks)

- The purpose of this is to make sure you are able to write an actual working program, and just not dumping all the sample code (e.g., lab class solutions) together in some meaningless way.
- **Your programs will only be tested on the departmental linux platform. Please make sure your programs work there.**
- I am not your human debugger, and this is not meant to be a thorough testing with unit test cases and code coverage and all that. It will only check if your program appears to behave in the expected way.
- There will be small penalties for things such as userid/filenames being hardcoded (when the question says they shouldn't), needing some fixed path ("C:\Users\abc123\Desktop\alice.prv" - I'm not abc123, I don't have such a user account on my PC and furthermore we are using Linux), wrong upper/lowercase filenames (please note that the Linux filesystem is case-sensitive so alice.prv is not the same as Alice.prv), hardcoded host and port so command-line arguments not used, wrong command-line argument order, or otherwise not conforming to

the assignment specifications. The marking is largely manual with the help of some automation scripts, and all these problems make semi-automated marking more difficult or impossible.

Marks are given according to some simple bands based on how successful your programs run, such as: (these are just indications and are not exhaustive)

- 0 - Does not compile, or does not do anything meaningful
- 5 - Attempt to do at least part of what is required but never worked (e.g., appears to send something but the other side never received)
- 10 - None of encryption/signature/hashing is handled correctly (e.g., messages appear sent/received but both encryption and signature either did not happen or are always wrong, and unhashed filename is sent). This may be detected based on the screen outputs, or that everything still works when the wrong keys are supplied.
- 15 - Either encryption, or both signature/hashing, are not correct (similar ways of detection as above)
- 20 - Encryption works but one of signature/hashing is not correct; or there are other non-crypto errors (e.g., required information not printed; server quits after one client)
- 25 - Always runs successfully

An important element to be assessed is whether you know who should own which key. If your program crashes because it requires a key that it should not have, you will lose most or all the marks here. This is not a "small" mistake.

Code inspection (70 marks)

Again, each of these are marked in a banded way, based on how complete/correct it is.

It will only be concerned about whether you are using the right API methods, feed them with the right data and use their output. Marks are approximately distributed as follows.

- RSA en/decryption - 15
- AES key generation and en/decryption - 20
- MD5/hex - 10
- Signature generation/verification - 20
- Other client-server interaction – 5

For each item above, a percentage of the allocated mark is given according to:

- 0% - There is nothing
- 20% - Some fragment of code attempting to use those API methods but not correctly
- 40% - Correct ways of using the API methods in principle, but not adapted to the question in hand, or with major errors, or very incomplete
- 60% - Completed "one half" (e.g., encryption without decryption; send without receive; generating without verifying signatures) broadly correctly; or like below but with more errors
- 80% - Used the API methods for the tasks broadly correctly but with some small issues
- 100% - All correct

Readability (5 marks)

- Quite frankly I'm only concerned with two things:
    1. proper indentation;
    2. meaningful variable/method names.
- I am not really that concerned about object-orientedness and modularity, factoring of common functions, etc. Neither are other "nice" coding practice like catching exceptions (except those that the assignment specifies you must handle).
- Efficiency of your code (how quick it is) is not part of the marking criteria (within reason).
- Only write comments that are helpful. Most of the code are standard stuff, and if you are doing it the right way, there is very little need for comments. In any

case, good code should be self-explanatory. In particular, do not write comments that simply translate your code back into English. For example, please do not write comments like this:

```
// first add 1 to the variable x
x++;

// then print the value of variable x
System.out.println(x);
```

So here is the marking criteria for readability:

- 5 - The marking of your code went so smoothly that the issue of readability never came to my mind; or
- 5 - Your comments actually helped me understand some clever bits of your code that I otherwise would have great difficulty understanding
- 4 - Your indentation/variable names are not misleading
- 3 - Could have misled someone, but not me
- 2 - Actually misled me and wasted a lot of my time
- 1 - You should enter the IOCCC if this were C
- 0 - There is too little code, it is impossible to be unreadable; or
- 0 - You only pasted my code samples together; there is nothing actually written by you. I'm not going to award readability marks to my own code