Computer Networks 159.334 Assignment 3 due 29/May/2015

Encryption - a simple implementation of the RSA algorithm

Introduction

In this assignment your task is to implement RSA in the client/server application using TCP sockets. As you might remember, the client can type messages to the server, and the server echoes back the message. You have to modify the client so it *encrypts* all communication. You have to modify the server, so you can print the *decrypted* message locally.
The first time a connection is established, the server will inform the client about its *public key* for the current session. The client will use the *public key* to generate the encrypted messages. The server uses its *private key* to decrypt the original encrypted messages.
In subsequent sessions (after the client disconnects and connects again), the key should *not* be the same (see notes 2).
The server should be able to generate the following answer, as seen in the example below:

- The client types: `hello`

- The server prints locally:
```
The original encrypted message was "asddf"
After decryption, the message is: "hello"
```

Implementation details

There are many details in the implementation of a real RSA that might have to be left out in order to keep it under the scope of an assignment. You have to ask the following questions regarding some decisions about the implementation:
a) What are the sizes of the keys?
b) Encryption should be done character by character or computed over a set of characters?
c) Should padding be used?
d) Do you need an arbitrary precision library or the keys are small enough to use a simple exponential code?
e) How are the keys sent to the client (your protocol design: format, message order, etc).

The answers to these questions will establish how your own "encryption protocol" works.
Follow the following instructions:
1) You are allowed to use relatively small prime number pairs, but do not use pairs that will make the same keys for encryption and decryption (so avoid the pairs such as 5 and 7 for example).
2) The keys should be different for subsequent connections. You can use a limited set of keys *defined statically*, or compute a new key when the client asks for the establishment of the connection. In this case, the server has to run the Extended Euclidean Algorithm, or try to find **e** and **d** by brute force.
3) You are allowed to use server2015.c and client2015.c, found in the 159334 Stream. Modify these sources for your implementation of encryption protocol.

Testing the assignment:

The assignment is going to be marked based on *functionality and design*. The marks are distributed as follows:
- 3 marks: successful connection using RSA keys.
- 3 marks: for protocols that include the possibility of using use of different keys for every session (you can repeat the key after two or more sessions).
- 3 marks: minimum implementation, able to send a set of alphanumeric characters (at least lower-case a-z, digits 0-9 and space). For that to work, you need to limit the key sizes, or use encoding/decoding.
- 6 marks: correct implementation for different RSA keys, i.e., the encryption/decryption results are always correct for any of the keys used.

Notes:

1 - Submit your files electronically via Stream.
2 - This assignment is worth **15 marks**.
3 - Marks will be subtracted for obvious copying and/or for delays without justification.