Secure Chat

In this assignment you are going to implement in Python 2.7 a client/server program for secure instant messaging.

Just as in the case of the tic-tac-toe implementation in assignment 1, you are going to write two programs: a client and a server. The server is going to be listening on some arbitrary port, while two clients connect to the server in order to chat with each other.

The idea is to have something similar to the UNIX "talk" utility which is a visual communication program which copies lines from your terminal to that of another user. The difference is that in your implementation the conversation is going to be encrypted.

The specifications for the encryption component is as follows: initially, using Diffie-Hellman, the two clients (via the server) agree on an initial seed for the RC4 stream cypher, which is going to generate the "keystream" used for XOR encrypting/decrypting of the conversation-stream at the two ends. The assumption is that the conversation is carried out with a "stream" of ASCII characters.

Some more details: client 1 will write some text, and by pressing 'return' the text will be encoded locally with RC4, sent to the server, and the server will forward the ciphertext to client 2, who will decode it into plaintext, and display it on the terminal. Note that the Diffie-Hellman key exchange occurs only once at the beginning, during initialization, in order to generate the seed. Also, note that it is insecure to have RC4 "restart" for each message; the two clients should each have two RC4 processes going: one for encrypting, and one for decrypting. Both processes should keep track of the number of characters received, and the number of characters sent, in order to use the appropriate segments of the stream.

Some reminders:

1. Work in groups, just as in assignment 1. Your group should ideally have 3 members, and not more than 4. Singletons and pairs are allowed, but five or more members is not allowed.
2. IP addresses and ports should not be hard-coded.
3. Include a manual on how to run your program. As for assignment 1, you should submit `client.py`, `server.py` and `doc.pdf`. The manual should be contained in the documentation. Only one group member needs to submit the assignment using SVN, but the names of all the group members should be clearly specified.
4. In order to be graded your program must run. Programs that do not run will get a grade of zero. If you are unsure, you can check your code with a TA during lab hours, to make sure that it runs. Late assignments will *not* be accepted this time; part of the assignment is to do good "time management" and "deliver the product on time"!