Lingfeng Yang / Xingjian Yan
604251317/ 104304403
yanglingfeng@ucla.edu
yxjnb@ucla.edu

Lab 4 Design - Transmitting Encrypted Files

Specification of the Feature's Behavior:

In order to make p2p file transfer more secure, we intend to make file contents invisible from network snoopers and unauthorized peers. Besides, we also aim to hide the existence of the file from unauthorized peers. To do so, we encrypt the contents of the file byte by byte.

(We added two new flags to the program to implement the desired functionality. Using '-e', we can run the program in encryption mode. Using '-p', users can change their password. We use '1234' as the default password. Running '-ep' enables a user to change the default password and then enter the encryption mode.) change to--> The current interface is as follows: use "-s" to trigure secure mode and asks if the user wants to

set a new passkey. the pass key will hold through out the entire session. A peer who does not have the passkey is assumed to be unauthorized.

Ideally, the peer client would have a list of peers that has different authorizations, and the key would be established though an encrypted message using Public Key encryption.

Plan for Implementation:

Our first step is to choose the encryption method for file contents. We first did some research on encryption and decide to use a straight forward "session key" method to encrypt the files, using the passkey as the session key, though out an interaction with another peer. More specifically, for the sake of demonstration, our algorithm is xor bit shift for every byte in the file. We implemented an encryption function which can also be used as a decryption function due to the nature of xor bit shift. When we upload a file, the encryption function would open the file and create a temporary file for the original file. The passkey acts as a session key in this case. The function loops through the file until the end of the file, apply ^password to each byte of the file and then copy the byte to the temporary file. After that, the upload function send the temporary file instead of the original file to the downloading peer. When we download the file, we're

Lingfeng Yang / Xingjian Yan
604251317/ 104304403
yanglingfeng@ucla.edu
yxjnb@ucla.edu

We also ask for a password before trying to download a file from a peer. For our purpose, we use a default password which is set every time we run the program. However, for an more complete password system, we would want the local client to have a file list of peers that have different levels of tust and exchange passkey autoatically, or have the user decide weither to trush or not on each secure interaction (--we would have to implement a database and look to the database to search for passwords and change passwords. For our program, a single password is suffice for the purpose.--) If the user cannot provide the correct password, he cannot begin downloading in the first place. Even with the downloaded file, if a user does not have the correct password, he will not be able to decrypt the file.

The third part of the design problem requires us to hide what filed we have from unauthorized peer

Summary of Results:

We successfully implemented file encryption and decryption with a password system. Without the correct password, the user cannot download and decrypt the file contents. However, the system is limited for one password and could be compromised by brute force.

Who Did What:

Lingfeng did research on encryption method and decided to use xor bit shift encryption and decryption. Then Xingjian implemented the encryption/ decryption function, and then implemented the part that request for password confirmation. Then Lingfeng and Xingjian completed the design report.