How to remotely connect to a HPC

by Xiaofeng Liu, Ph.D., P.E.
Assistant Professor
Department of Civil and Environmental Engineering
Penn State University

Note: This tutorial is a demonstration of how to connect to a computer cluster remotely. I used my research cluster as an example. Details may vary if you want to connect to other computers. You should consult your HPC user manual or documentation.

1 Introduction

To access the cluster machines, you will login to *xliuws0514.cee.psu.edu* using your assigned username and password. This short instruction demonstrates how to connect to the cluster using *PuTTY*.

2 SSH connection using *PuTTY*

2.1 Download and install *PuTTY*

This is a short guide to getting started using *PuTTY* to access the cluster. If you need more information on how to use PuTTY, please see the official *PuTTY* documentation located at: http://www.chiark.greenend.org.uk/ sgtatham/putty/docs.html

To install PuTTY on your own computer, you need to download a copy at its official website: http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html.

For the case of installing *PuTTY* on windows machine, it is very straightforward. Just double click the executable file you downloaded and do a regular installation.

2.2 Login

Launch the installed *PuTTY*, and enter the following information (varies depending on specific HPC):

- Host name or IP: *xliuws0514.cee.psu.edu*

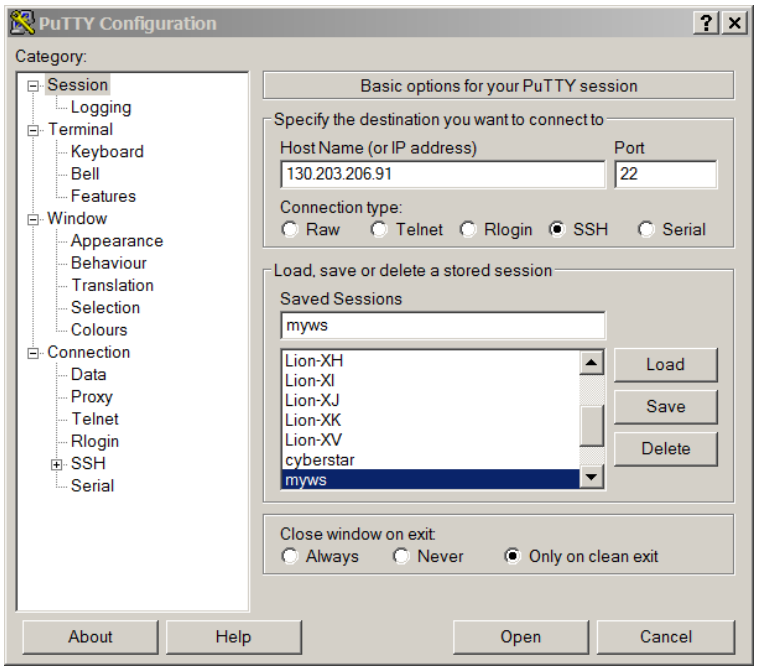

Figure 1: Configuration screen of *PuTTY*

- Port: *22*
- Connection type: *SSH*

To avoid the entering of these information every time, you can save them into a session. Next time you only need to load the saved session.

Click "Open", it will connect to the cluster and the following windows will appear. Follow the instruction to input your user name and password. If everything is correct, your login will be successful. As a final note, to log off properly, just type `exit` in the terminal.

3 Logging into the cluster from other Linux machines using *ssh*

If you have access to other Linux machines, you can login to the cluster using the *ssh* command.

```
[app321@someLinuxBox ~] $ ssh -P 1209 app321@takara.hpc.utsa.edu
```

The syntax and options for the *ssh* command can be found from its manual by typing `man ssh`. In this case, the option "-P 1209" specifies the port number as 1209. The default

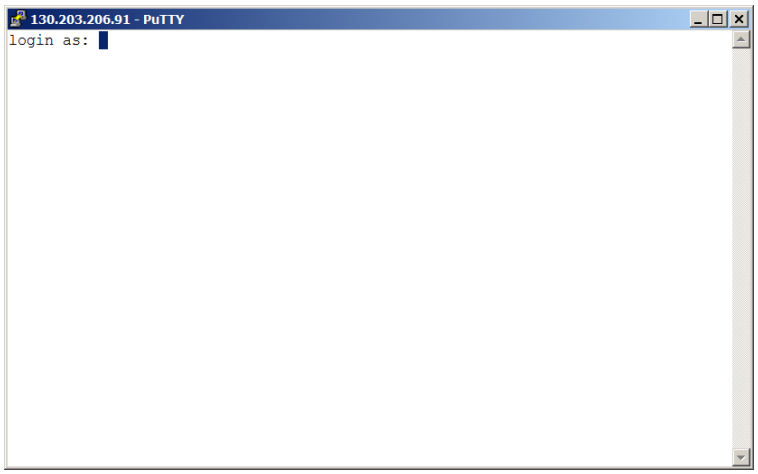

Figure 2: The login screen to type in your user name

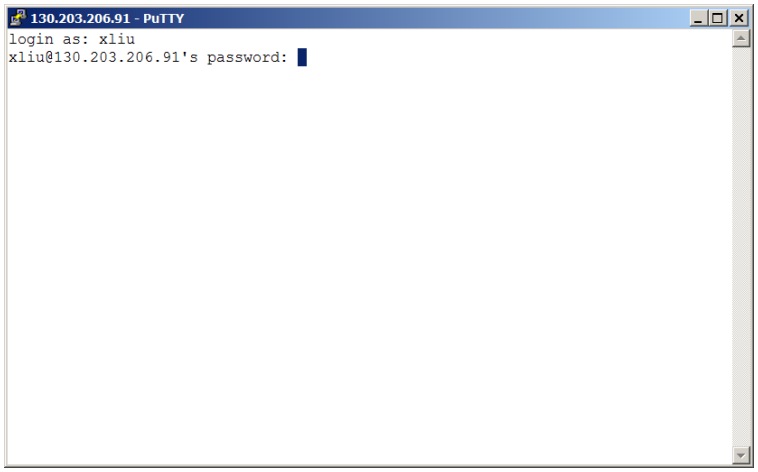

Figure 3: The login screen to type in your password

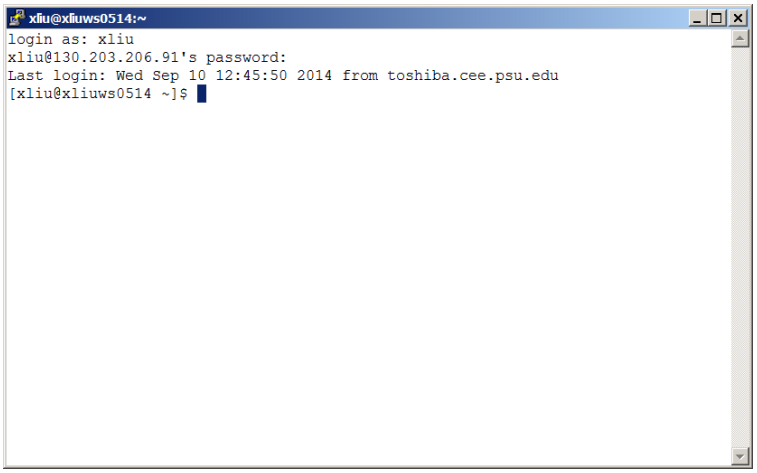

Figure 4: The screen after your successful login

for SSH port is 22.

4 Remote desktop connection via VNC

Connection through *PuTTY* is limited to command line, though graphical interface is possible through X11 forwarding. A more convenient way is to use the remote desktop connection via VNC. This part will guide you through setting up the VNC session and connecting to it from Windows machine. On the Windows machine, *PuTTY* should be installed already. You also need to install a VNC viewer. There are several choices, such as UltraVNC viewer and RealVNC viewer. Free versions, though with less functions, are available from their websites. In this tutorial, the RealVNC viewer is demonstrated. You should download and install it on your Windows machine.

The setup and connect through VNC, do the following steps

1. Connect to the Linux machine using *PuTTY* with your user name and password (Figure 5).

2. Start the VNC server on the cluster host by typing (Figure 6)

   ```
   vncserver -depth 24 -geometry 1024x768
   ```

   Replace 1024x768 with desired resolution. If this is the first time starting vncserver, you'll be asked to setup the password that will be use to connect to the vncserver. It is recommend to setup the password. Otherwise, anybody can connect through the

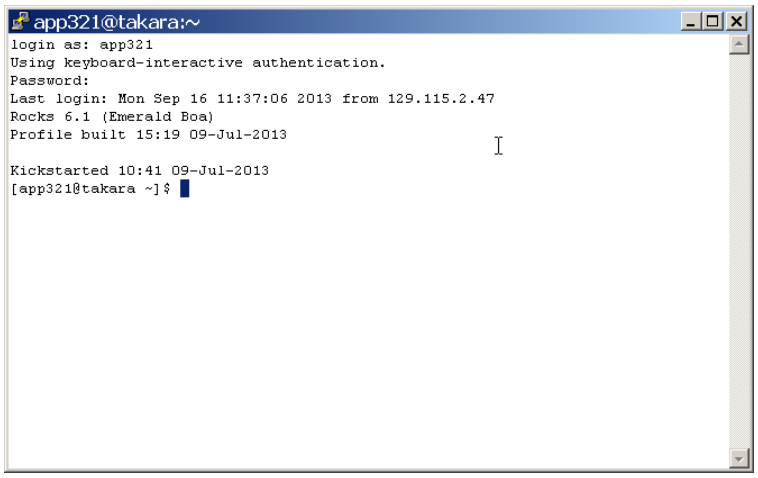

Figure 5: Connect to the the Linux machine using *PuTTY*

VNC.

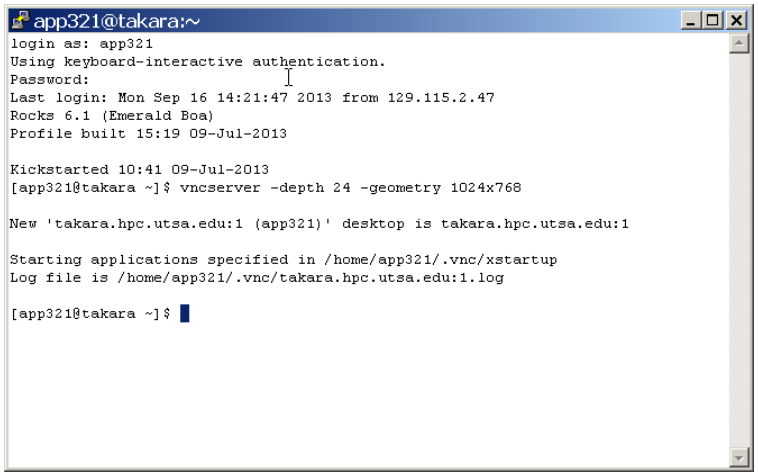

Figure 6: Startup of the VNC Server

Notice the desktop number is
`New 'xliuws0514.cee.psu.edu:1 (app321)' desktop is xliuws0514.cee.psu.edu:1`

3. On your Windows machine, launch the RealVNC viewer. Specify the name of the server as the desktop number you noticed in the previous step. Click "OK" (Figure 7)
4. VNC viewer program will ask for the passphrase authentication. Type in the passphase you set for the VNC connection. Click "OK" (Figure 8).
5. If everything works ok, then the screen of the VNC remote desktop will appear as

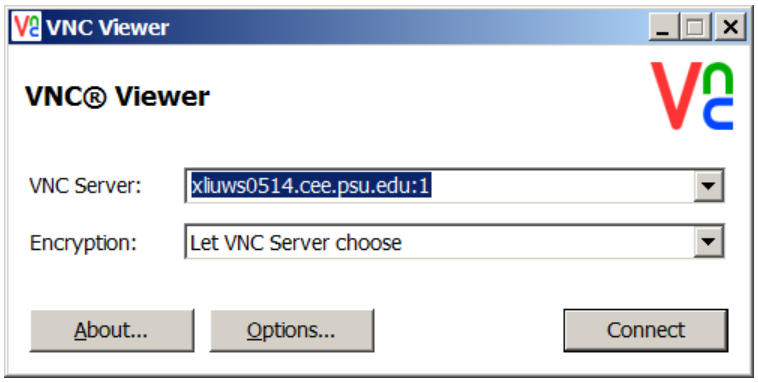

Figure 7: Connect to the VNC Server by launching the VNC viewer on your Windows machine

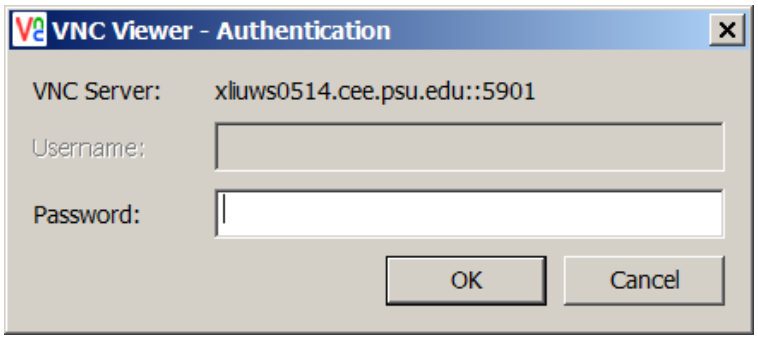

Figure 8: Passphrase authentication for the VNC connection

in Figure 9. You can use the cluster as if you are sitting in front of the machine physically. A useful tip is that you can type the command `vncconfig` and see a new window popup. Leave this window open and you should be able to share the clipboard memory between Linux and Windows. That is convenient to copy and paste between to machines.

5 Moving files to and from the cluster

You may want to move files to and from the cluster. For example, when your simulations are done, you may want to copy the results to your computer for analysis. This section demonstrates how to do this in a secure way.

If you connect to the cluster from a Linux machine, you can use sftp. The following is an example to get a file from the home directory on the cluster:

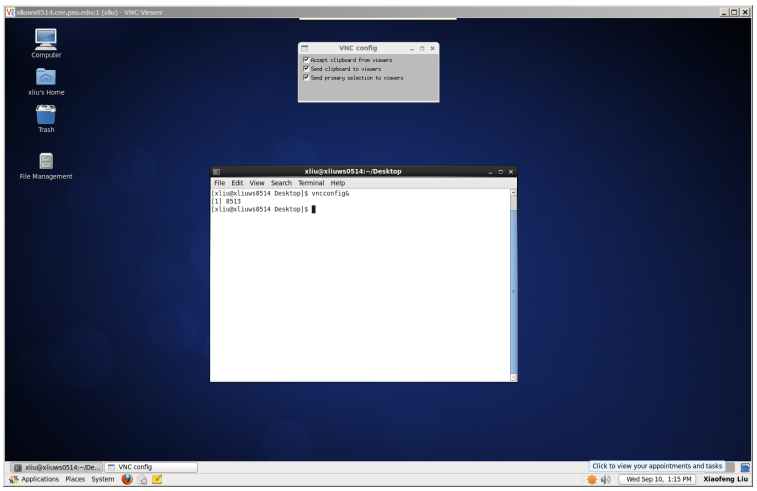

Figure 9: Screen for the VNC connection

```
[app321@SomeLinuxBox ~]$ sftp app321@xliuws0514.cee.psu.edu
Connecting to xliuws0514.cee.psu.edu...
Password:
sftp> get testfile.txt
Fetching /home/app321/testfile.txt to testfile.txt
/home/app321/testfile.txt ...
sftp> quit
```

If you connect to the cluster from a Windows machine, you can also use programs with graphical interfaces, such as `FileZilla` client. The free client program can be downloaded from its website:
https://filezilla-project.org/

1. Launch of the FileZilla program. After the installation, you can launch the FileZilla program by double clicking.
2. Setup a SFTP connection in FileZilla by clicking on the "Site Manager" button on the tool bar. A dialog window will appear as in Figure 11. Type in the following information:
   - Host: xliuws0514.cee.psu.edu
   - Port: 22
   - Protocol: SFTP - SSH File Transfer Protocal
   - Logon Type: Normal
   - User: `your user name`

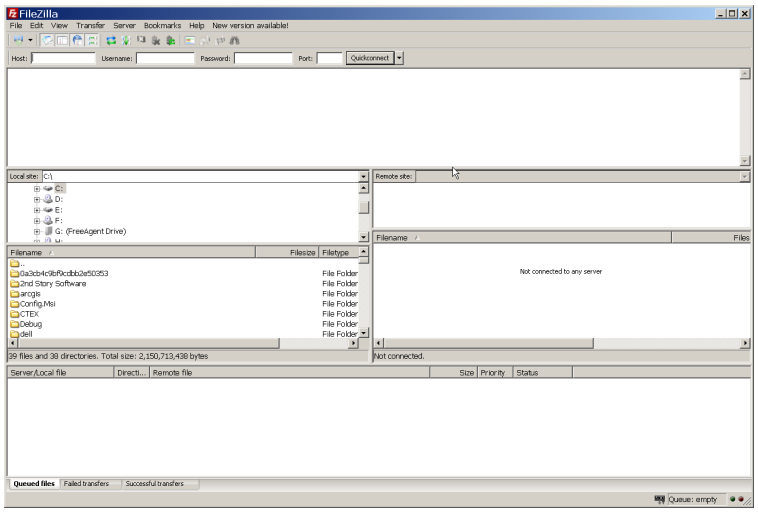

Figure 10: Launch of the FileZilla program

- Password: `your password`

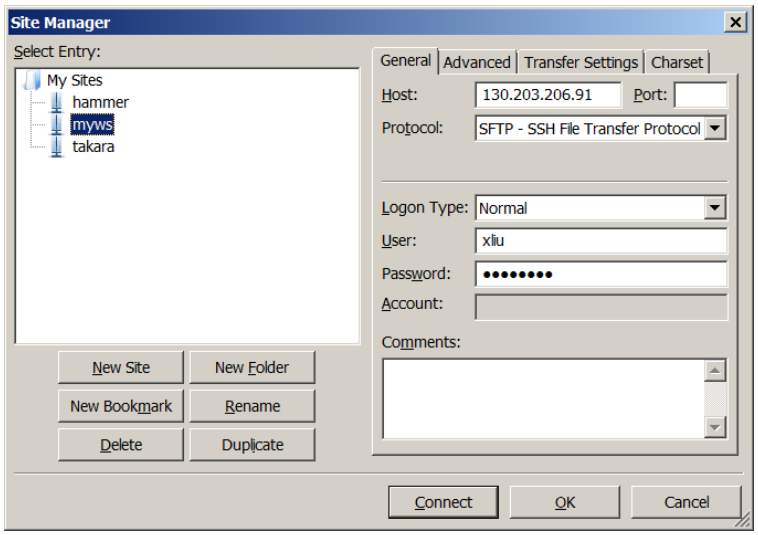

Figure 11: Setup of a SFTP connection in FileZilla

3. Click "Connect" and you will be connected to the cluster. By default, FileZilla will go to your home directory (Figure 12). As you can see from the figure, you can transfer files and directories from and to the cluster.

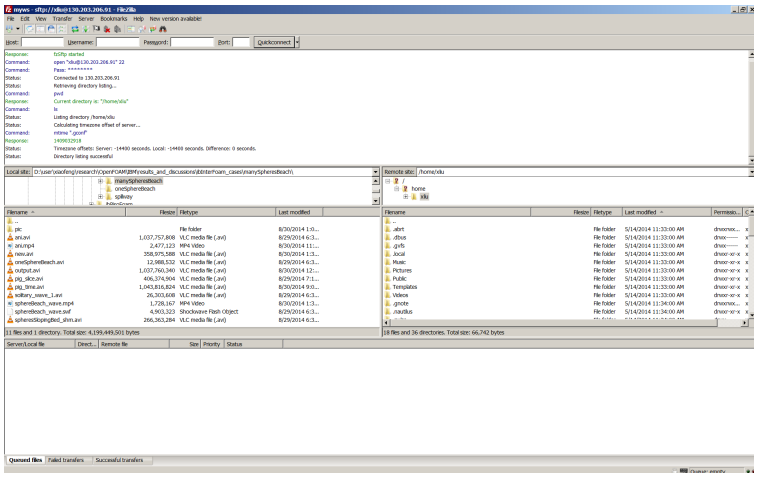

Figure 12: Screen for a connection session in FileZilla

6 Advanced topics

6.1 Generating public/private rsa key pair for automatic login.

The following information is from the Ubuntu website (all credits go to the original authors): https://help.ubuntu.com/community/SSH/OpenSSH/Keys.

6.1.1 Public and Private Keys

If your SSH server is visible over the Internet, you should use public key authentication instead of passwords if at all possible. It is important for security reason. Hundreds, even thousands, of attempts to guess password and break-ins can be easily logged for an ordinary computer.

With public key authentication, every computer has a public and a private "key" (a large number with particular mathematical properties). The private key is kept on the computer you log in from, while the public key is stored on the `.ssh/authorized_keys` file on all the computers you want to log in to. When you log in to a computer, the SSH server uses the public key to "lock" messages in a way that can only be "unlocked" by your private key. This means that even the most resourceful attacker break into or interfere with your session. As an extra security measure, most SSH programs store the private key in a passphrase-protected format, so that if your computer is stolen or broken into, you should have enough time to disable your old public key before they break the passphrase and start using your key.

Public key authentication is a much better solution than passwords for most people. In fact, if you don't mind leaving a private key unprotected on your hard disk, you can even use keys to do secure automatic log-ins. An automatic login allows a user to login from certain computers without typing the passphrase every time. Different SSH programs generate public keys in different ways, but they all generate public keys in a similar format:

```
<ssh-rsa or ssh-dss> <really long string of nonsense> <username>@<host>
```

6.1.2 Key-Based SSH Logins

Key-based authentication is the most secure of several modes of authentication usable with SSH, such as plain password (the default with most Linux systems) and Kerberos tickets. Key-based authentication has several advantages over password authentication, for example the key values are significantly more difficult to brute-force, or guess than plain passwords, provided an ample key length. Other authentication methods are only used in very specific situations.

SSH can use either "RSA" (Rivest-Shamir-Adleman) or "DSA" ("Digital Signature Algorithm") keys. Both of these were considered state-of-the-art algorithms when SSH was invented, but DSA has come to be seen as less secure in recent years. RSA is the only recommended choice for new keys.

Key-based authentication uses two keys, one "public" key that anyone is allowed to see, and another "private" key that only the owner is allowed to see. To securely communicate using key-based authentication, you need to create a public key for the computer you're logging in from, and securely transmit it to the computer you're logging in to.

Using key based logins with ssh is generally considered more secure than using plain password logins. This section of the guide will explain the process of generating a set of public/private RSA keys, and using them for logging into your Linux computer(s) via SSH.

6.1.3 Generating RSA Keys

The first step involves creating a set of RSA keys for use in authentication. This should be done on the client. To create your public and private SSH keys on the command-line:

```
mkdir ~/.ssh   (only needed if .ssh directory is not created yet)
chmod 700 ~/.ssh
ssh-keygen -t rsa
```

You will be prompted for a location to save the keys, and a passphrase for the keys. This passphrase will protect your private key while it's stored on the hard drive and be required

to use the keys every time you need to login to a key-based system:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/b/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/b/.ssh/id_rsa.
Your public key has been saved in /home/b/.ssh/id_rsa.pub.
```

Your public key is now available as `.ssh/id_rsa.pub` in your home folder. You now have a set of keys. Now it's time to make your systems allow you to login with them.

6.1.4 Choosing a good passphrase

Just like with physical keys, you need to change all your locks if your RSA key is stolen. Otherwise, your thief will be able to get access to all your stuff.

An SSH key passphrase is a secondary form of security that gives you a little time when your keys are stolen. If your RSA key has a strong passphrase, it might take your attacker a few hours to guess by brute force. That extra time should be enough to log in to any computers you have an account on, delete your old key from the `.ssh/authorized_keys` file, and add a new key.

Your SSH key passphrase is only used to protect your private key from thieves. It's never transmitted over the Internet, and the strength of your key has nothing to do with the strength of your passphrase. If you choose to use a passphrase, pick something strong and write it down on a piece of paper that you keep in a safe place. If you choose not to use a password, just press the return key without typing a password - you'll never be asked for one again.

6.1.5 Key Encryption Level

Note: The default is a 2048 bit key. You can increase this to 4096 bits with the -b flag (Increasing the bits makes it harder to crack the key by brute force methods).

```
ssh-keygen -t rsa -b 4096
```

6.1.6 Password Authentication

The main problem with public key authentication is that you need a secure way of getting the public key onto a computer before you can log in with it. If you will only ever use an

SSH key to log in to your own computer from a few other computers (such as logging in to your PC from your laptop), you should copy your SSH keys over on a memory stick, and disable password authentication altogether. If you would like to log in from other computers from time to time (such as a friend's PC), make sure you have a strong password.

6.1.7 Transfer Client Key to Host

The key you need to transfer to the host is the public one. If you can log in to a computer over SSH using a password, you can transfer your RSA key by doing the following from your own computer:

```
ssh-copy-id <username>@<host>
```

Where `<username>` and `<host>` should be replaced by your username and the name of the computer you're transferring your key to.

To specify a port other than the standard port 22, you can issue the command like this: `ssh-copy-id "<username>@<host> -p <port_nr>`. If you are using the standard port 22, you can ignore this.

Another alternative is to copy the public key file to the server and concatenate it onto the authorized_keys file manually. It is wise to back that up first:

```
cp authorized_keys authorized_keys_Backup
cat id_rsa.pub >> authorized_keys
```

You can make sure this worked by doing:

```
ssh <username>@<host>
```

You should be prompted for the passphrase for your key:

```
Enter passphrase for key '/home/<user>/.ssh/id_rsa':
```

Enter your passphrase, and provided host is configured to allow key-based logins, you should then be logged in as usual.