```
Configuration of Server
-----------------------
File: /etc/ssh/sshd_config

Disable direct root login to avoid ssh attacks:
  PermitRootLogin no

Use only protocol 2:
  change Protocol 2,1 to just Protocol 2


Configuration of Client
-----------------------
File: /etc/ssh/ssh_config

Use only protocol 2:
  change Protocol 2,1 to just Protocol 2


Setting up SSH to Allow Remote Login w/o a Password
---------------------------------------------------
On the local machine, generate a key pair:
> ssh-keygen -t dsa

Since we do not want a password/passphrase,
just hit return twice when prompted to enter.

Creates two files in ~/.ssh:
id_dsa  and  id_dsa.pub

The .pub file contains the public key, and it needs
to be (securely) copied to every machine you want
to be able to remotely login to.

On these machines, it needs to be placed into the file:
~/.ssh/authorized_keys2

The file can be securely copied to a remote machine using scp:
> scp id_dsa.pub carver@carverpc.cs.siu.edu:/home/carver/temp

Then SSH-in to the remote machine and do something like:
> cat temp/id_dsa.pub >> .ssh/authorized_keys2


Securely Copy Files to Remote Machine:
--------------------------------------
Can use scp command, virtually like cp:
  scp [[user@]host1:]file1 [...] [[user@]host2:]file2

E.g., scp localfile carverpc.cs.siu.edu:remotefile

file2 can be a directory, and works like cp

Supports -r option for recursive copying of directory.


Securely Execute a Command on a Remote Machine:
-----------------------------------------------
ssh allows one to execute a command line on remote machine:
  ssh user@host command
```

E.g., ssh carverpc.cs.siu.edu ls -l

This is an easy way to restart services or run other one-line commands without needing to establish a remote shell.

Encrypting X Sessions

ssh -X hostname

(default on most distributions now)

Combine with command execution to open window from remote application:

E.g., ssh -X carverpc.cs.siu.edu kwrite&

Note suggested method is: ssh -f -X carverpc.cs.siu.edu kwrite

(check on process remaining open issue)

Forwarding Ports

You can forward a port on your local host directly to a port on a remote host. For example, let's say I'd like to be able to use a MySQL GUI application on my local machine to work with MySQL on my Web server. All I need to do is run the command:

ssh -f -N -L 3306:localhost:3306 username@webserver

-f option tells ssh to go into the background.
-N option tells ssh not to execute any remote commands (e.g., a shell).

-L option tells ssh to forward a port from the local host to a port on the remote host. In this case, both ports are 3306, so any traffic directed to port 3306 on the local machine will be directed to port 3306 on the Web server. Note that you will need to be logged in as root to forward the privileged ports below 1024.

Can now run local MySQL GUI client and point it at localhost port 3306, it will actually be communicating with MySQL on the Web server over an encrypted connection.

If there's a firewall between local machine and Web server so cannot connect directly to the Web server via SSH, if working with a Linux firewall, or other firewall that's running SSH, you can still connect. Forward a local port to port 25, in order to connect to SMTP over a secured connection:

ssh -f -N -L 2205:webserver:25 username@firewall

2205:webserver:25 tells ssh to forward traffic from local port 2205 to port 25 on host Web server.
username@firewall specifies firewall as the host to forward traffic through, so the traffic will pass through that host.

Traffic from local host to the firewall will be encrypted, but traffic between the firewall and the Web server will not be.

Can also forward traffic from a remote host to your local host, by using the -R option instead of the -L option:

```
ssh -f -N -R 8081:localhost:8080 webserver
```

This will forward all traffic from port 8081 on the host webserver to port 8080 on the local machine. Note that you need to be a privileged user on the remote host to open privileged ports.