Secure Shell SSH

ssh, or Secure Shell is an encrypted network communication protocol to allow remote login and other network services to operate securely over an unsecured network. All data is encrypted which protects it against spoofing, interception, and even intermediate data manipulation on systems you connect through. Common applications include remote login and remote command execution, but any network service can be secured with ssh.

ssh has two main components: the client and server applications. The client is what you will use on your local machine, which allows you to access remote systems running the ssh server. There are several open source and commercial ssh projects available, but OpenSSH, an open source solution, is the most popular. ssh is a major and venerable project, dating back into the 1990's, with a number of forks. Many of the available commercial applications are actually just clones or wrappers around OpenSSH.

Unix-based systems such as OS X and Linux distributions come with at least the client version of OpenSSH pre-installed. Windows is different — see below

Manual Key Generation: ssh-keygen

To use public key authentication, which is stronger than UNIX passwords, first use "ssh-keygen -t rsa -b 4096" to generate a personal 4096-bit private & public key pair. You can name your keys with your name and their date of creation.

When creating a key you are prompted for a passphrase to protect your private key. This passphrase is important — it should be a strong passphrase (at least 12 characters; not all letters or numbers). Pick something you will remember because there is no way to recover a key if you forget the password.

1. With Terminal still open, copy and paste the text below. -C adds a comment — you can specify your email and the date.
Use a very strong passphrase. For more information see Working with SSH key passphrases.

```
ssh-keygen -t rsa -b 4096 -C your_email@example.com
# Creates a new ssh key, using the provided email as a label
# Generating public/private rsa key pair.
```

2. We suggest keeping the default settings as they are, so when you're prompted to "Enter a file in which to save the key", just press Enter to continue.

```
# Enter file in which to save the key (/Users/YOU/.ssh/id_rsa): [Press enter]
```

3. You'll be asked to enter a passphrase.

```
# Enter passphrase: [Type a passphrase]
# Enter same passphrase again: [Type passphrase again]
```

4. After you enter a passphrase, you'll be given the fingerprint, or id, of your ssh key. It will look something like this:

```
# Your identification has been saved in /Users/you/.ssh/id_rsa.
# Your public key has been saved in /Users/you/.ssh/id_rsa.pub.
# The fingerprint is:
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your_email@example.com
```

- DO NOT email us your fingerprint (in green above)
- Now you should have the following files (and perhaps more) in $HOME/.ssh:

```
pepper@teriyaki:~$ ls -l .ssh/
-rw-------   1 pepper  staff  1743 Dec 12  2008 .ssh/id_rsa
-rw-r--r--   1 pepper  staff   402 Dec 23  2008 .ssh/id_rsa.pub
```

5. Show your public key with the command:

```
cat ~/.ssh/id_rsa.pub
```

6. Public Key Installation for New Accounts

- If you are requesting a new account, paste your public key file id_rsa.pub into the account request form and we will install it for you:

```
 lski1907:.ssh perixj$ more id_rsa.pub ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDTma3hCuHD866j0Lq40NsuelvECBz+zdyM8AKXWREjsheHzd9vVsL2BAmJbKhLa6BfRg
QgeeeynzampaX2rTblBlesiZTtPqlMXtZV57PSjoWUXCGWxivgbJrLkeVMdeh81ZYAPiMMYUm3CR25oRP6AcnnpsSvkUBiyU1s
Ru5jIsC7JDazPKJFi44UIeSoQOUHoFnKENpIgF5cL8Y1c50mV+zBbUds3FaGi4Vv6xsYcvxgAsmyYj1U9CrxWyG8RP4c++NT+5
hTVqj4KxmvQJASjYAkpIu23m2Fb4DL6Kbmd8PsnWtYcJ5SsRpdrFyP6ne4nsFC5SphATpKHDHgI2MBCCrq+p4hMYS
/6Umb4FhLyQaLWe+qj1zXRyhLiyiRzPe0BkbcjbhqXOh2M3zWtGACudOGRFG2uXGrZ4KoUGaiuSq+BxYffHEVPkIJfMzOotd3P
YxtlBgDw5MLBiLLKP+hzdix04ce6jjKEjBYMy5VlB4kqMOBzSdKtDSS5VbYc+gczgOMjYig5fJaMBLemxWZmLQaLlkoHgVWCm8
rsPPK7CQys8Mq+z3Q27VmW4tYhpJAARtFxMLofKs/owm9/ThGCZm5Qlr3LUfnIjm4nHyunzlswTNa6uiGHLR
/sAxBc05y5NGkcMsJ8f7Cxb2WxDSjK34i+gOywv6cOKqP9ILjiP== juancerixn@gmail.com
```

WARNING: Your private key file (id_rsa) is effectively a password, so be careful with it! Private keys should be kept secure, and should not be stored on insecure machines (or any multi-user machines, if possible). Check with our system administrators if you need to use a key in batch jobs.

Private keys must not be readable by any other account aside from the super-user, so mode 600 is suggested. Public keys and their parent directories (including /home and $HOME) must not be writable by other users. Keys with insecure permissions are silently ignored.

Manual Public Key Installation

If you already have an account, your $HOME/.ssh/authorized_keys file should contain one or more public keys, one per line. The private keys which match these keys provide access to this account (private keys do not need to be on a server to ssh into it). id_rsa is the default ssh version 2 private key; it should be present on your workstation. The ssh command first tries any keys available in an agent, then ~/.ssh/id_rsa if present, and finally falls back to password authentication.

To install your public key on a server for authentication, use a command like “scp id_rsa.pub SERVER:.ssh/authorized_keys” (this copies the key). Alternatively you could log in with your password and use a text editor like vi to create the $HOME/.ssh/authorized_keys file. Afterwards ssh *$SERVER* sho uld prompt for your key’s encryption passphrase rather than your UNIX password on *$SERVER*. If it doesn’t work, use your account password to login and check permissions (ls -la ~/.ssh). Make sure that there are no line breaks in the key. Each key should be one single line. If you still have problems contact hpc-request@cbio.mskcc.org.

Workstation Global Configuration

You can configure ssh globally on your desktop or laptop. Most OpenSSH configuration is in sshd_config & ssh_config; the files are in /etc under Mac OS X, or /etc/ssh under Linux. User configuration goes in ~/.ssh/config (ssh) & ~/.ssh/authorized_keys (sshd).

The old version 1 ssh protocol has serious known flaws and should not be used. ForwardX11Trusted is required on Mac OS X for interoperation with certain Linux programs (mostly Red Hat python utilities).

/etc/sshd_config should include Protocol 2, and possibly X11Forwarding yes.

/etc/ssh_config should generally include:

```
ForwardAgent yes
ForwardX11 yes
ForwardX11Trusted yes
Protocol 2
```

You can also do this on per user basis with an ssh config file $HOME/.ssh/config

Windows

On Windows 10, you can use Microsoft's OpenSSH. Alternatively download and run the PuTTY MSI installer; if you cannot run the installer on your PC, you can simply download the `.zip` archive. This gives you PuTTY, pscp, psftp, PuTTYgen, and pageant (you do not need PuTTYtel).

Launch PuTTYgen and click Generate to create a key (this will require waving the mouse pointer around over a rectangle so it can extract randomness from your motions). Use “Save private key” in the lower-right, and save your new private/public keypair in a .ppk file in a safe place (such as C:\keys\YOURNAME.ppk), for PuTTY to use later. Use a strong password to encrypt your .ppk file.

Then select the complete contents of the text area at the top of the window, which should start with ‘ssh-rsa’; make sure to get the entire public key, which is likely to extend below the bottom of the text area, and paste this into your New HPC account request. Optionally, create a new plain text file in Notepad and save it as C:\keys\id_rsa.pub.

Launch PuTTY, create a new configuration, and save it. The new configuration should include username@hostname (like joe7@lilac.mskcc.org) in the server field.

Assuming you will not use pageant, at least initially, from the left-side menu, select SSH:Auth, and specify your .ppk file at bottom-right.

Scroll back to Session at the top of the left-side menu, and Save your configuration. Now you can double-click it in PuTTY to connect. Once the system administrators have created your account and installed your public key, double-clicking your saved configuration in PuTTY should connect you to the server, after prompting for the passphrase you used to encrypt your .ppk file.

pageant

If you find yourself entering your encryption passphrase too often, you can use pageant to cache your private key.

If you will use pageant to cache your private key(s), make sure the .ppk file type is assigned to pageant in Windows. Remove any .ppk keys from SSH: Auth for all configurations to force PuTTY to use pageant for authentication.

To load a key into pageant at login, put a shortcut to it into your personal Startup folder. You can open the Startup folder from the Start bar:Windows menu: All Programs:(Right-click) Startup:Open.

Authentication Forwarding

To use authentication forwarding on Windows, so you can ssh to into a server, and then ssh *again,* from there to a second server (this works best with pageant):

1. Open PuTTY.
2. Load the PuTTY session (configuration).
3. Check the authentication forwarding box in PuTTY:SSH:Auth.
4. Save the PuTTY session (configuration).

Agents

ssh-agent remembers the passphrase so you not need to type it every time you connect or to the server. Note that **IdentityFile and IdentitiesOnly prevent use of ssh-agent**.

Mac: Apple Keychain

When you access an encrypted ssh private key in Mac OS X 10.11 “El Capitan” and earlier, Mac OS X prompts you to save its passphrase (encrypted) in the Apple keychain. From then on your private key will be automatically decrypted and available on request. When this is working, "ssh-add -L" should show at least one key available on your Mac or Linux system. After you "ssh -A" to another host with agent forwarding, "ssh-add -L" should show the same key(s) available as on your local desktop.

This is not default behavior in Mac OS 10.12 “Sierra”. To use the keychain in Sierra add “AddKeysToAgent yes” and “UseKeychain yes” to your config file (~/.ssh/config). Here is an example config file:

Host *

```
  ForwardAgent yes
  Protocol 2
  AddKeysToAgent yes
  UseKeychain yes
```

You should be able to manually add your private key passphrase to the Apple Keychain with "ssh-add --apple-use-keychain ~/.ssh/id_rsa", assuming ~/.ssh /id_rsa is your private key on a current version of macOS; for older versions you may need "ssh-add -K ~/.ssh/id_rsa" instead. You only need to do this once per key.

Linux/UNIX: ssh-agent & ssh-add

An ssh agent loads (encrypted) keys from disk, decrypts them in memory, and makes the keys available to ssh clients such as ssh, sftp, and scp. OpenSSH’s ssh-agent caches keys, and ssh-add controls ssh-agent. To load keys from disk, use a command such as ssh-add -t 540 private-key, which will read private-key, prompt for the passphrase to decrypt it, make the key available to ssh client programs for 9 hours, and then clear it from memory. "ssh-add -L" lists all loaded keys.

ssh-agent is intended to run as a parent process, spawning shells or an X11 session as child processes. The children automatically inherit access to the parent agent via environment variables. This is inconvenient for configurations without a single parent process for each user. ssh ‘keychains’ such as Gentoo keychain work by sharing a single ssh-agent across multiple processes.

Windows: pageant

PuTTY‘s pageant handles the whole login session. See above.

ssh Authentication Forwarding

For people who `ssh` into one computer such as the HPC SSH bastion host, xbio.mskcc.org, and from there to another computer such as lilac.mskcc.org, authentication forwarding enables the intermediary computer (xbio) to use ssh-agent on the original client (Mac or PC) to authenticate to the remote server (lilac). For example I ssh from teriyaki to xbio, and then from xbio to lilac. Even though I don’t keep my private key on lilac, teriyaki authenticates my connection to lilac (through xbio) so I don’t need my private key on xbio.

For Mac: First add your private key passphrase to the Apple Keychain with "`ssh-add --apple-use-keychain ~/.ssh/id_rsa`". Then enable authentication forwarding to xbio by adding the following to `~/.ssh/config` on your Mac:

```
Host xbio.mskcc.org

  Port 2222

  ForwardAgent yes
```

Or use “`ssh -p 2222 -A xbio.mskcc.org`" each time.

For Windows: See the Windows section above.

Copying Keys

Email messages are scanned and stored in multiple places, so do not email private keys. You can use `scp` or a thumb drive to copy them to a new Mac, but don't leave private keys on a thumb drive any longer than necessary -- they are easy to lose or steal.

The easiest option is to copy the entire `~/.ssh` directory. It contains private and public keys, `ssh` configuration, and the `known_hosts` file. The `~/.ssh` directory is normally hidden in the macOS Finder, but you can make it appear with the Terminal command `open ~/.ssh` or "Go to Folder..." in the Finder's **Go** menu.

Additional Information

Read the manual pages on your workstation or a server: `man ssh`. If you have any issues, please contact hpc-request@cbio.mskcc.org.