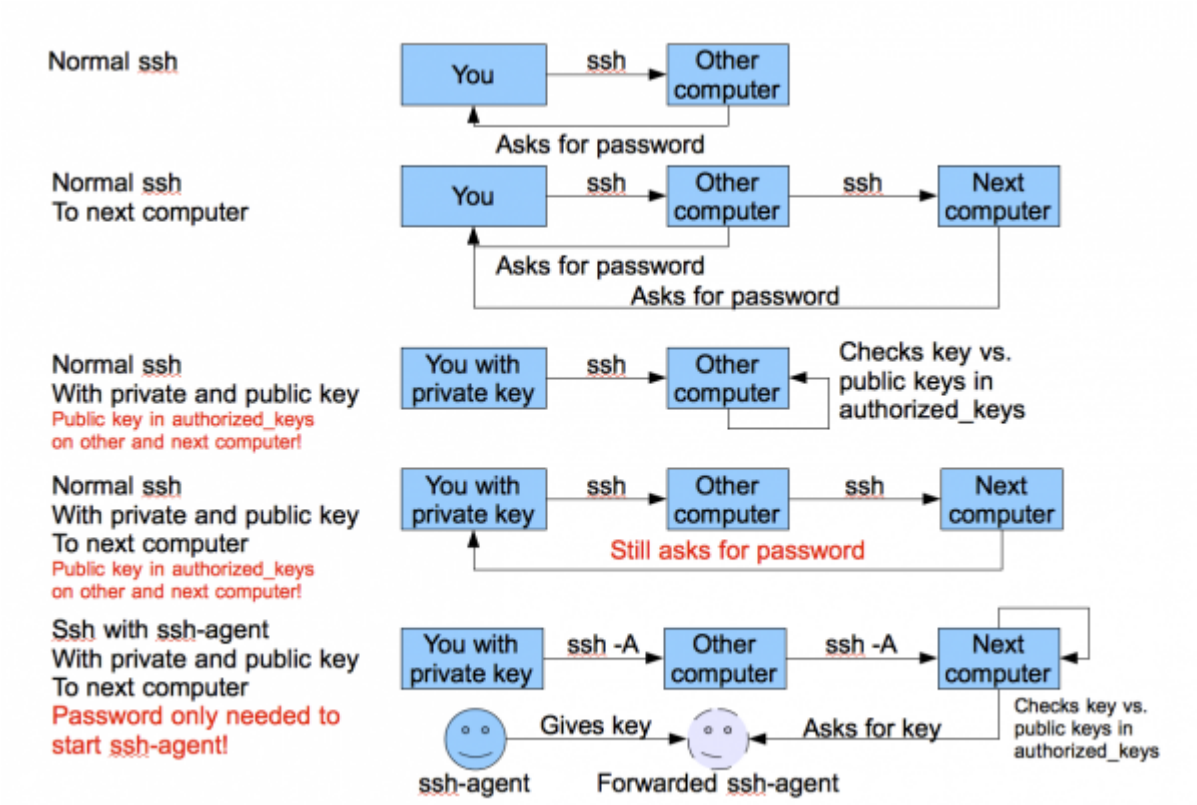
SSH Usage on CEP

We use the Secure Shell (ssh) on CEP to connect to different systems. This page explains how this can be used without having to supply a password each time you want to connect to a system. The image below tries to explain the process:

With normal ssh you always have to give a password. If you use a private and public key, you can access systems where your public key is in $HOME/.ssh/authorized_keys from the system where you have the private key. With the ssh-agent explained below, you can ssh to *any* system without having to supply a password. (very useful to run things on nodes of a cluster, or other remote machines.



Generating keys

Linux or OS X

The first thing you need to do is generate an authorisation key using the DSA algorithm, which means you need to do the following once. You need to have a somewhat recent version of OpenSSL on your system for this to work:

```
ssh-keygen -tdsa
cp .ssh/id_dsa.pub .ssh/authorized_keys
```

Use cat or some editor if authorized keys already exists and can't be simply copied. Copy your `.ssh/authorized_keys` to the `$HOME` of each system you want access to. Please make sure you

use a passphrase to encrypt your private key, to prevent easy access. When using the instructions below on the ssh-agent, you'll only have to provide it once each time you use the systems.

Windows

TBD probably requires an explanation on installing Putty

Using an SSH-Agent

An ssh-agent is a small program that when you start work is used to unlock the passphrase protected private key you generated above. The ssh-agent will from that point on automatically supply the right answers to any ssh session, if you use `ssh -A` each time you connect to another system.

Detailed information on how to setup ssh agent forwarding can be found here and here.

Linux

The ssh-agent runs in the user's local PC, laptop, or terminal. Authentication data need not be stored on any other machine, and authentication passphrases never go over the network. Also, the connection to the agent is forwarded over SSH remote logins, and the user can thus use the privileges given by the identities anywhere in the network in a secure way.

If you start the ssh-agent it creates a socket in `/tmp` and then generates a few commands on `stdout` which serve to set environmental variables and to tell you which PID the agent has. An example:

```
SSH_AUTH_SOCK=/tmp/ssh-jpIaV4861/agent.4861; export SSH_AUTH_SOCK;
SSH_AGENT_PID=4862; export SSH_AGENT_PID;
echo Agent pid 4862;
```

By 'eval'uating this code, the variables SSH_AUTH_SOCK and SSH_AGENT_PID will be set. You can use the SSH_AGENT_PID, for example, to kill the agent when you log off. The agent itself recognises the variable and commits suicide when you type:

```
ssh-agent -k
```

The SSH_AUTH_SOCK variable is mainly used by the program `ssh-add`, which uses it to determine with which agent to communicate. To get your agent running in (t)csh type:

```
eval `ssh-agent`
ssh-add
```

Please note the back-quotes. If you have bash, you might use the more readable:

```
eval $(ssh-agent)
ssh-add
```

More advanced ways to use ssh-agent on Linux.

OS X

Install SSH Agent 1.1 and set it to *Open at Login* or use the same commands as on Linux.

Windows

Use Pagent provided by Putty

- TBD

From:
https://www.astron.nl/lofarwiki/ - **LOFAR *Wiki***

Permanent link:
https://www.astron.nl/lofarwiki/doku.php?id=public:ssh-usage&rev=1254387292

Last update: **2009-10-01 08:54**