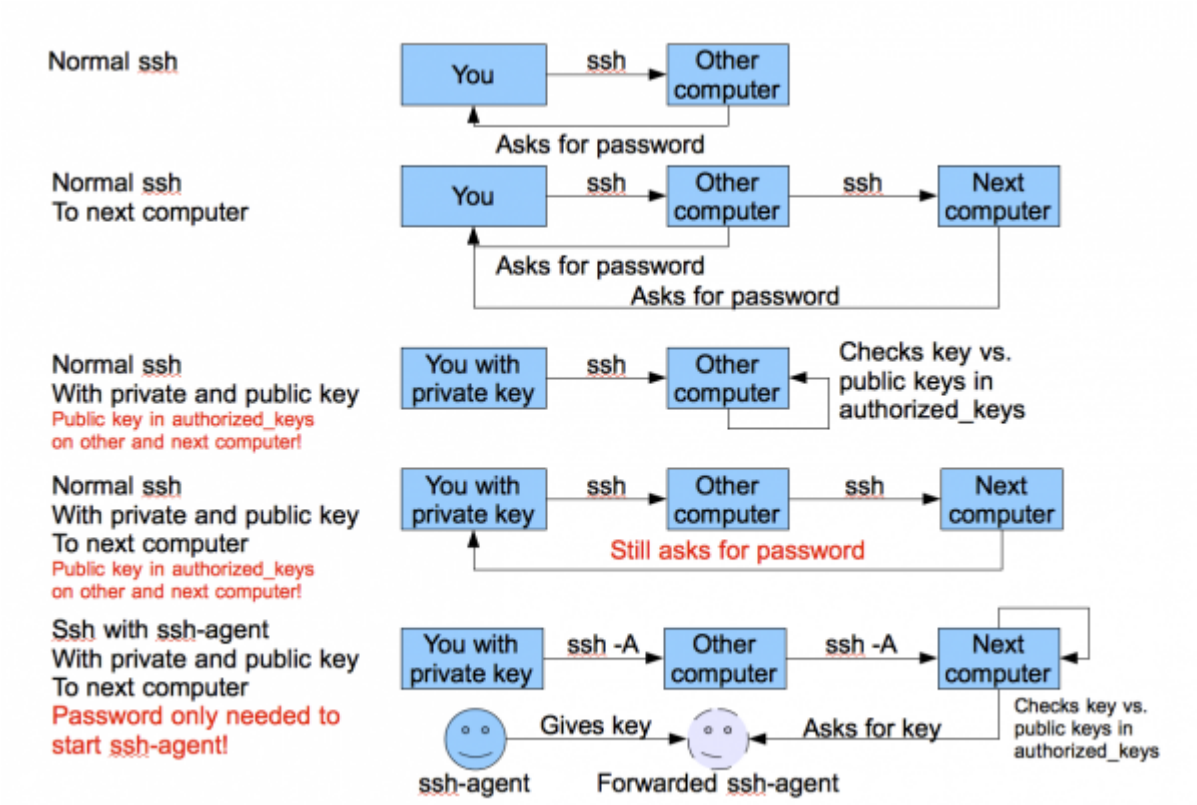
SSH Usage on CEP

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.

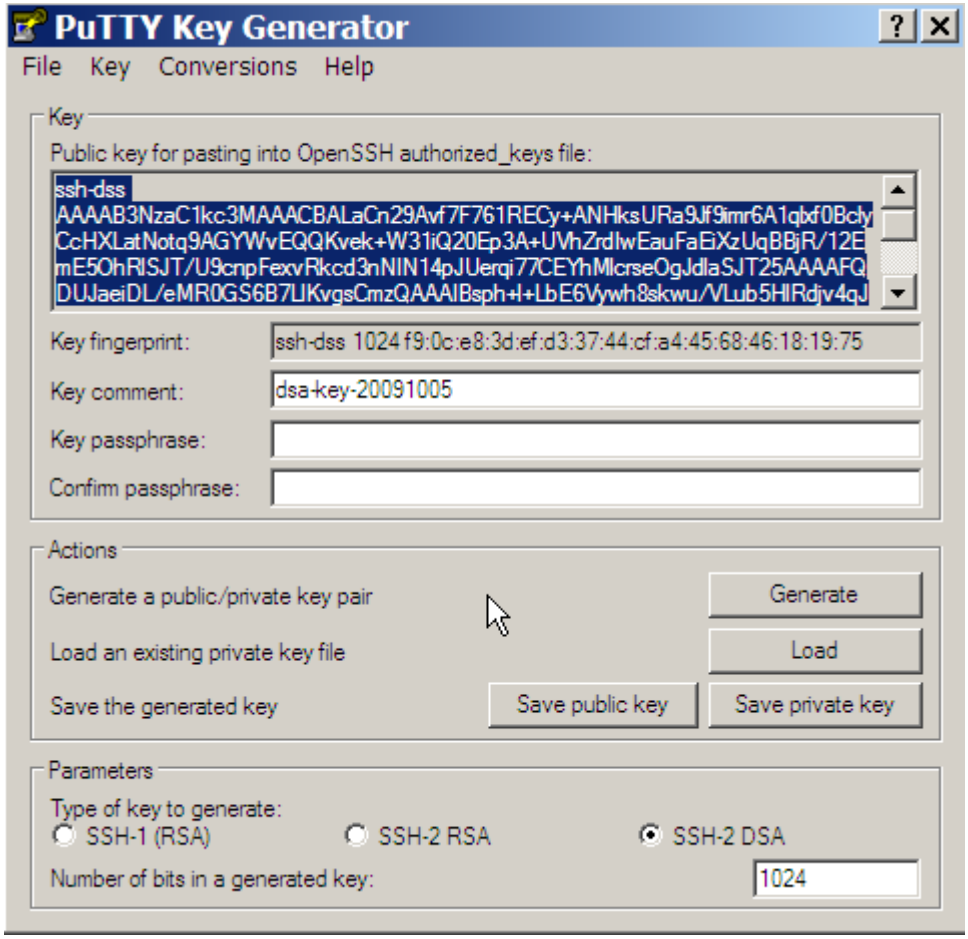
Windows

Windows users can download the Putty toolkit from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Please select the Windows Installer as that contains the whole package.

After installing, you'll have the Putty program, and several help programs such as Pageant.

Select from the Programs menu the Putty section and select PuttyGen.

- Select in the Paramaters section the key type (e.g., SSH-2 DSA).
- Press the Generate button.
- Now you'll have to move your mouse over the grey area below the progress bar.
- Once done, you'll get a screen like the one shown below.
- Enter a passphrase (and confirm).
- Now you can save the public and private key files in a location of your choice.



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
```

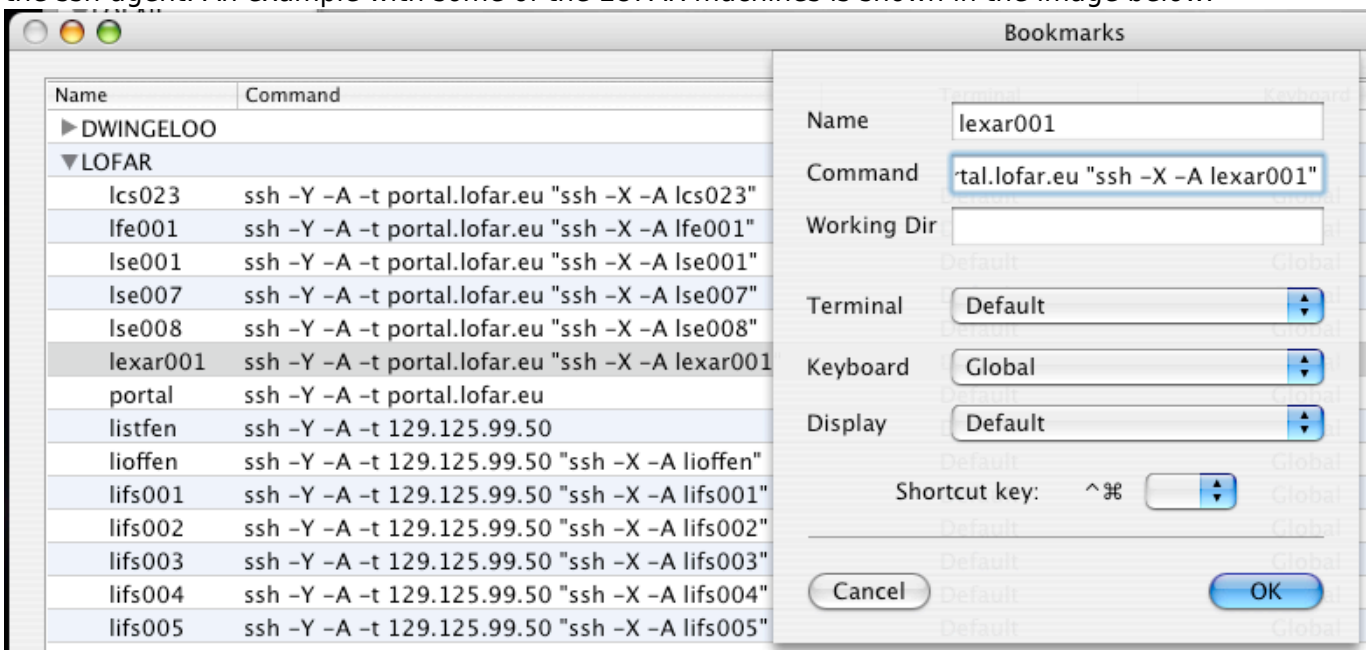
More advanced ways to use ssh-agent on Linux.

OS X

Recent versions of OS X (10.5 “Leopard” or later) have ssh-agent integration built in – and integrated with the system keychain for passphrase management. See this blog post for details.

On older versions of OS X, install SSH Agent 1.1 or SSHKeyChain and set it to *Open at Login* or use the same commands as on Linux.

If you have this set up, then you can easily make bookmarks in iTerm to access machines and forward the ssh-agent. An example with some of the LOFAR machines is shown in the image below.



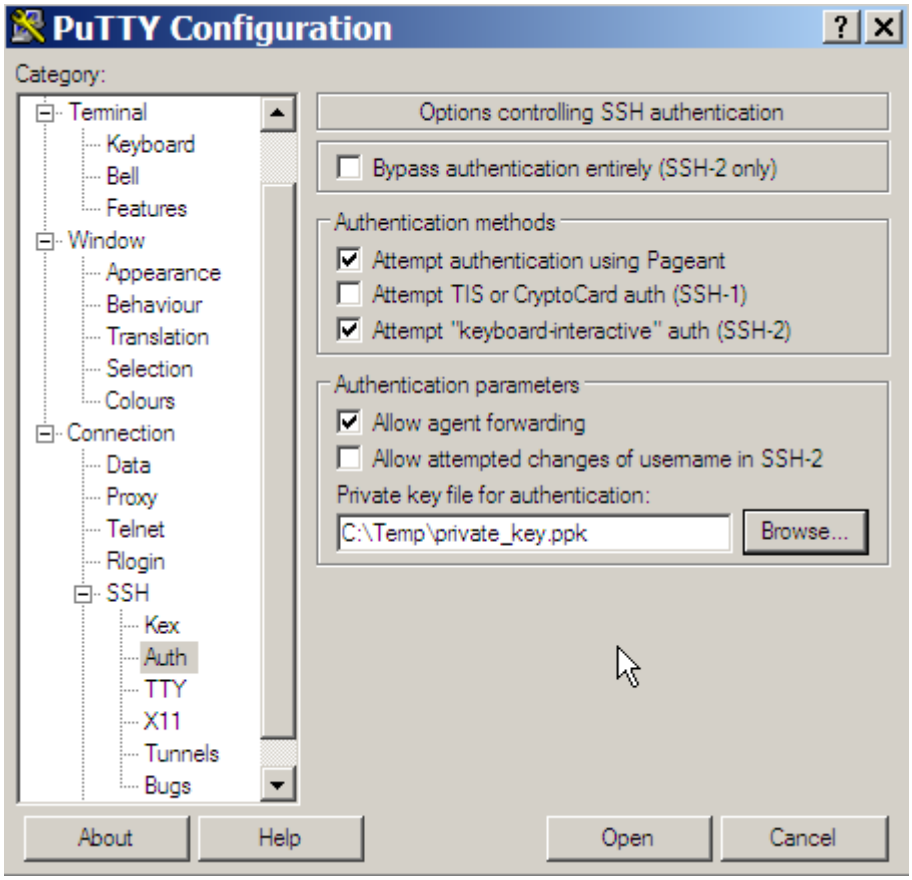
Windows

Windows users that have downloaded the `Putty` package (see above) and have keys generated and saved on their system can use the `Pageant` program in the `Putty` distribution.

To use it, select `Programs` → `Putty` → `Pageant`. Use the `Add Key` button to select the Private key file on your system. You will be asked to enter the passphrase.

Once you have done this, `Pageant` will be used to forward the keys to any system you have access to, and on which you have added the public key in the `$HOME/.ssh/authorized_keys` file.

To enable forwarding of your key in `Putty`, make sure that the session you use in `Putty` has the option 'Agent Forwarding' enabled and the location of the key file filled in the Connection/SSH/Auth section (see screenshot below).



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

Permanent link:
https://www.astron.nl/lofarwiki/doku.php?id=public:ssh-usage&rev=1255004016

Last update: **2009-10-08 12:13**