Note that the permissions on the socket file prevent people from accessing your agent - but on a regular Unix system the 'root' user can override these restrictions. Hence, 'root' can set SSH_AUTH_SOCK to your socket and use `ssh-add` to list/add/delete your keys. He can also log in on all of your systems without having to use a password. **Be warned.**

Persistent agent

In theory, you only need to start up the agent once on the host you use to connect to other systems (e.g. your laptop) and be done with it; all requests from all your shells may be handled by the very same agent. However, this requires the proper setting of the environmental variables. Alas, there is no simple way to find out which socket an agent uses. But a little script magic will do the trick. If you use `bash` you can copy the following code into your `.bashrc` to re-use your running agent:

```
# find out how many agents we have running. If more than 1 do nothing,
# if zero, start one up and if 1, re-use it
#
z=0
PA=$(ps -u $USER |awk '$NF=="ssh-agent" { print $1 }' )
for n in $PA
do
   ((z++))
done

# No agent runs for me, so start one:
if [ $z -eq 0 ]
then
   eval $(ssh-agent)
   ssh-add
fi

# There is an agent for me, so figure out the socketname and set the
variables
# ''manually'':
if [ $z -eq 1 ]
then
   # The socket has been created by the parent process, which has a PID one
   # less than that of the running agent. The PPID is needed to determine
the
   # name of the socket. Alternately you might get the PPID from the process
   # listing.
   #
   q=$PA
   ((q--))

   # Next, we search for the matching socket. There can only be one that has
   # the PPID in it's name.
   #
   candidate=$(ls /tmp/ssh-*/agent* |sed -n '/tmp\/ssh-
.*'$q'\/agent.'$q'$/p')
```

```
   # If what we found is a socket, set the environmental variables
   #
   if [ -S "$candidate" ]
   then
      export SSH_AGENT_PID=$PA
      export SSH_AUTH_SOCK=${candidate}
      ssh-add -L >/dev/null 2>&1
      if [ $? -eq 0 ]
      then
        echo "Reusing your existing ssh agent (pid is $PA)"
      else
        unset SSH_AGENT_PID
        unset SSH_AUTH_SOCK
      fi
   fi
fi

# A simple function to allow you to type ''gono <nodename> (as <identity>)''
# which will use ssh-agent forwarding to step right 'through' the
# portal to the end-node
#
gono() {
 if [ $# -eq 3 -a "$2" == "as" ]
 then
   ssh -A -t portal.lofar.eu "ssh -A $3@$1"
 else
   if [ $# -eq 1 ]
   then
     ssh -A -t portal.lofar.eu "ssh -A $1"
   else
     echo "Syntax error"
     echo "Usage: gono <nodename> as <user>"
   fi
 fi
}
```

To list the identities (keys) currently known by your `ssh-agent`, you can use:

```
ssh-add -L
```

Note that it is possible to FORCE the name of the socket which `ssh-agent` will use by specifying the `-a` flag. Hence, you might also consider putting the socket for your agent in your HOME directory. You could simplify the script accordingly.