SSH-Authentifizierung durch Schlüsseldatei
(ggf. ohne Passworteingabe)

Hier die Schritt-für Schritt-Anleitung (nur für ssh2):

Ich gehe davon aus, dass das Paket openssh auf Server und Client installiert ist und das die Versionen einigermaßen aktuell sind und folglich rsa und ssh2 unterstützen. Ich gehe auch davon aus, dass sich die Adressen Deiner Hosts auflösen lassen, sonst musst Du natürlich die IP-Adressen verwenden. Ich gehe auch davon, dass Du weißt, warum Du einen solchen Schlüssel nicht für den root erzeugst und dann irgendwo liegen lässt...

1. Schritt:

Du erzeugst mit :

```
user@rechner:~> ssh-keygen -b 1024 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):ENTER
Enter passphrase (empty for no passphrase):ENTER
Enter same passphrase again:ENTER
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
e3:14:23:32:6f:d5:d9:cd:a0:d5:7c:64:45:ca:61:45
user@rechner:~>
```

ein Schlüsselpaar. Du wirst zu einer Passworteingabe aufgefordert. Du kannst dabei ein Passwort eingeben - musst aber nicht. Gibst Du ein Passwort ein, dann musst Du es Dir gut merken. Dazu später mehr.

Die Dateien werden in Deinem Heimatordner im Unterordner ".ssh" gespeichert. Mit...

```
user@rechner:~>ls -l ~/.ssh
```

... solltest Du mindestens folgende Ausgaben erhalten:

```
-rw------- 1 user users  887 25. Jun 20:05 id_rsa
-rw-r--r-- 1 user users  224 25. Jun 20:05 id_rsa.pub
user@rechner:~>
```

Ein wenig Theorie:

Der Befehl „**ssh-keygen -b 1024 -t rsa**“ erzeugt also zwei Schlüsseldateien:
„`id_rsa`“ ist eine Art privater Schlüssel, ich nenne ihn künftig „user authentification key“, also „Benutzer-Authentifizierungs-Schlüssel“. „`id_rsa.pub`“ dient als Gegenstück auf dem „Server“ - das ist der (oder sind die Rechner mit denen Du Dich verbinden verbinden willst.)

Die Option „-b 1024“ sorgt dafür, dass ein 1024 bit- langer Schlüssel erzeugt wird. Ohne diese Angabe wird ein 2048 bit langer Schlüssel erzeugt, der etwas übertrieben ist. Die Mindestlänge ist 768 bit. Die Option „-t rsa“ sorgt dafür, dass es ein „RSA-Schlüssel“ ist. Man könnte auch „DSA“ verwenden. Das ist gleichwertig. Ich bleibe deshalb hier beim RSA-Schlüsselpaar.

Falls Du mal einen Schlüssel für ein veraltetes Linux oder Unix (z.B. einen alten SCO-Server) erzeugen musst verwende „**ssh-keygen -b 1024 -t rsa1“** Mit aktuellen Linux-Versionen wird der Schlüssel nicht funktionieren.

2. Schritt:

Hier muss gesehen werden, ob Du am Client oder am Server angemeldet bist:

Version 1: Du sitzt am Client?
Du musst den „server authentification key“ auf den Server übertragen. Dies geschieht, indem Du ihn an die dort befindliche Datei ~/.ssh/authorized_keys2 anhängst. Existiert diese nicht, dann wird der folgende Befehl sie erzeugen:

```
user@client:~> cat ~/.ssh/id_rsa.pub | ssh user@server 'cat - >>
~/.ssh/authorized_keys2'
```

[ohne den Zeilenumbruch eingeben!]

```
Password: **********
user@client:~>
```

Das einzugebende Passwort ist natürlich das für Dein Benutzerkennzeichen auf dem entfernten host ("Server"). Falls Du zum ersten Mal eine Verbindung zum Server aufbaust wirst Du gefragt, ob Du den Fingerprint akzeptierst.

Selbstverständlich kannst Du den Schlüssel auf diese Art und Weise auch auf weitere Rechner exportieren, auf die Du so Zugriff haben willst. Wiederhole einfach den Befehl.

Version 2: Falls Du aber am Server sitzt ist es einfacher:

```
user@server:~> cat ./ssh/id_rsa.pub  >> authorized_keys2
```

Jetzt solltest Du Deinen „user authenification key“ sicher verwahren. Auf einem Speicherstick zum Beispiel. Der kann (unter SuSE-Linux) als `/media/sda1` geladen sein:

```
user@client:~> mv ./ssh/id_rsa /media/sda1/schluessel_zum_glueck
```

Bitte beachte, dass der Schlüssel nicht kopiert, sondern verschoben wurde!

Die Verbindung testest Du mit:

```
user@client:~> ssh -i /media/sda1/schluessel_zum_glueck user@server
```

Das sollte etwa so aussehen:

```
Last login: Mon Jun 25 20:31:14 2007 from client.home
Have a lot of fun...
user@server:~>
```

Hast Du den Schlüssel mit einem Passwort versehen, so erfolgt clientseitig noch eine Abfrage.

3. Schritt:

Du musst als root die Datei `/etc/ssh/sshd_config` auf dem Server bearbeiten, denn noch ist die Anmeldung mit einem Passwort möglich!

In `/etc/ssh/sshd_config` findet sich ein Eintrag:

`„UsePAM yes“` Setze den auf: `„UsePAM no“.` Das nachfolgende server #> **`/etc/init.d/sshd restart`** oder (SuSE-Linux) ein `server #>` **rcsshd restart** sorgt dafür, dass die Anmeldung auf dem Server nur noch mit dem „user authenification key“ möglich ist.

Schlüssel verloren oder möglicherweise durch Unberechtigten kopiert?

Das ist nicht gut. Jeder, der ihn hat, kann eventuell ohne Passwort auf Deinem Server mit Deinen Rechten arbeiten. Bearbeite die Datei „`~/.ssh/authorized_keys2`“ auf dem Server. Lösche die Zeile in der sich der verlorene Schlüssel befindet und fange wieder bei Schritt 1 an.

Ein wenig Theorie:

Das Verfahren schützt Deinen Server nicht nur vor brute-force-Angriffen, sondern auch Deine künftigen Verbindungen vor „man in the middle“- Attacken.

Die Authentifizierung sowohl für „ssh“ als auch für „scp“ erfolgt jetzt mittels der Schlüssel. Ist der „user authenification key“ auf einem Speicherstick oder nicht (mehr) auf dem Client in `~/.ssh/id_rsa` gespeichert, dann musst Du den Schlüssel jeweils mit der Option „-i `/pfad/zum/schluessel`“ angeben.

Mit Deinem Schlüssel solltest Du vorsichtig umgehen. Dazu gehört, ihn nicht auf fremden Rechnern „herumliegen“ zu lassen, und falls Du ihn von fremden Rechnern aus benutzt oder Dein eigener Rechner Dir nicht vertrauenswürdig genug ist, ihn schon in Schritt 1 mit einem sicheren Passwort zu schützen. Dazu verschlüsselt „ssh-keygen“ Deinen „user authenification key“, also die Datei „`~./ssh/id_rsa`“ mit „3DES“. Bei der Verbindungsaufnahme wirst Du auf dem Client nach dem Passwort gefragt, der Schlüssel muss ja entschlüsselt werden. Dazu wird das Passwort vom ssh- oder scp- Client benötigt.

Bei „paranoiden“ Sicherheitsanforderungen wirst Du das Passwort nicht in einer X-Oberfläche eingeben, weil diese für viele Angriffe („keylogger...“) bessere Voraussetzungen liefert als eine Standard-Konsole. Aber das wusstest Du schon- stimmts?

Weitere wirklich lesenswerte Informationen liefert ein „man 1 ssh-keygen“ auf der Kommandozeile. Leider nur in englisch.