# Security & Firewalls: HowTo: Secure Shell (SSH) - Public Key Authentication

## Sinn & Zweck

Public Key Authentication dient dazu, sich per SSH auf einem anderen Rechner einzuloggen oder Dateien zu übertragen, ohne sich jedes Mal mit einem Paßwort authentifizieren zu müssen.

Dazu wird ein Schlüssel erzeugt. Ein so genannter Private-Key (der bleibt auf dem Client-Rechner) und ein Public-Key (der geht auf den Server). Diese beiden nennt man dann "Schlüssel-Paar". Der Private-Key ist der wichtigere und schützenzwertere Teil.

Wenn man sich vom Client aus am Server anmelden möchte, werden die Schlüsselpaare ausgetauscht und dadurch authentifiziert sich der Client ohne das User-Paßwort.

## Sicherheit

Der Private-Key ist dann dem normalen Passwort gleichgestellt. Im Gegensatz zum Passwort existiert existiert er aber als Datei. Diese gilt es natürlich vor fremdem Zugriff schützen. Deswegen besteht die Möglichkeit, den Private Key wiederum mit einer Passphrase (wie Passwort, nur länger) zu schützen.

Damit der der Sinn & Zweck erhalten bleibt gibt es den SSH-Agent, der in der Lage ist, sich die Passphrase für die Dauer ein Sitzung zu merken.

Alternativ besteht auch die Möglichkeit, eine leere Passphrase zu verwenden. Dann ist aber der Private-Key ungeschützt! Er kann also verwendet werden, wenn er in Hände Dritter fällt.

Es gilt zusätzlich zu beachten, daß man den Private-Key niemals unverschlüsselt überträgt (FTP, Email), auf File-/Samba-Servern legt oder gar mit falschen Leserechten versieht.

## SSH-Version

Im Unix-Umfeld sind drei gängige Varianten im Umlauf:

SSH1 und SSH2 von ssh.com und OpenSSH, welches beide Protokolle unterstützt.

Im Linux-Umfeld trifft man vorallem auf OpenSSH.

Die Versionen der vorhandenen Client und Server erfährt man z.B. so:

```
#Client-Version:
ssh -V
#Server-Version:
```


(c) 2025 huschi <huschi@huschi.net>

```
telnet localhost 22
```

Fehlersuche

Der häufigste Fehler:

Die Verzeichnisse `~/.ssh` oder `~/.ssh2` müssen schreib- und lesegeschützt sein. Lediglich der der User selbst darf darauf Schreib und Leserechte haben:

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/*
```

Erzeugung des Keys:

Unter Windows nutzt man i.d.R. Putty und als Keygen `puttygen.exe`.
Da dabei alles recht verständlich ist, wird jetzt hier lediglich auf Linux-Rechner eingegangen:

Auf dem Rechner, von dem aus man sich auf einem anderen Rechner ohne Paßwort einloggen will, geht man wie folgt vor:

```
#Verzeichnis anlegen (falls nicht vorhanden)
mkdir .ssh
chmod 700 .ssh
cd .ssh
# bzw. für ssh2
mkdir .ssh2
chmod 700 .ssh2
cd .ssh2

# ein SSH1-Key mit ssh1:
ssh-keygen
# ein SSH1-Key mit OpenSSH:
ssh-keygen -t rsa1
# ein SSH2-Key mit ssh2:
ssh-keygen2
# ein SSH2-Key mit OpenSSH:
ssh-keygen -t dsa
```

Mit `ssh-keygen` werden zwei Schlüssel erzeugt.


(c) 2025 huschi <huschi@huschi.net>
URL: https://www.huschi.net/index.php?action=artikel&cat=14&id=141&artlang=de

Je nach Version fragt das Programm:

- In welcher Datei der Schlüssel gespeichert werden soll.

  Hier kann man einfach den vorgegebenen default übernehmen.

- Danach wird eine Passphrase für die Schlüssel abgefragt.

  Gibt man hier nichts ein, so wird der Private Key unverschlüsselt abgelegt.

Der Public Key wird evtl. auf dem Bildschirm ausgegeben. Wegen Zeilenumbrüchen ist es aber besser die ebenfalls erstellte Datei zu nutzen, statt per Cut/Paste die Daten zu übertragen.

Einrichten des Servers

Zunächst übertragen wir also den Public Key auf den Server (noch mit Passwort-Abfrage).

(Windows-Nutzer nehmen natürlich WinSCP dazu.)

Hier bleiben wir jetzt am Beispiel für SSH1. Verzeichnise und Dateinamen bitte entsprechend anpassen.

```
scp $HOME/.ssh/identity.pub user@server:
#und auf den Server wechseln:
ssh user@server

#Auch hier muß das entsprechende Verzeichnis (achtet auf ssh2!) angelegt werden:
mkdir .ssh
chmod 700 .ssh

#hochgeladene Datei reinschieben
#(vorsicht weil evtl. authorized_keys bereits existiert)
cat identity.pub ››.ssh/authorized_keys
rm identity.pub
chmod 600 .ssh/*
```

Jetzt sollte alles so eingerichtet sein, daß man sich ohne Paßwort einloggen kann.


(c) 2025 huschi <huschi@huschi.net>

Feinheiten:

**a)** Bei SSH2 (nicht OpenSSH) auf dem Server, werden die Public Keys nicht in authorized_keys zusammen gefaßt, sondern bleiben in Ihren Dateien vorliegen. Damit der SSH-Server weiß, welche Dateien er nutzen soll, gibt es entsprechende Einträge in `.ssh2/authorization`.

Sinnvollerweise benennt man den Public-Key entsprechen um:

```
mv id_dsa_2048_a.pub .ssh2/mein_client_dsa.pub
echo "Key mein_client_dsa.pub" ››.ssh2/authorization
```

**b)**Bei SSH2 (nicht OpenSSH) auf den Client, muß den ssh-client mitgeteilt werden, welche Schlüssel er nutzen soll:

```
mv id_dsa_2048_a .ssh2/mein_client_dsa
echo "IdKey id_dsa_2048_a" ››identification
```

**c)** Wenn ein ohne OpenSSH erstellter SSH2-Public-Key auf einem OpenSSH-Server eingesetzt werden soll,
braucht er noch eine Umwandlung:

```
ssh-keygen -i -f id_dsa_2048_a.pub ››.ssh/authorized_keys
```

**d)** Umgekehrt: Ein OpenSSH erstellter Key, der auf einem SSH2-Server eingesetzt werden soll muß noch auf dem Client umgeändert werden:

```
ssh-keygen -e -f id_dsa.pub ›id_dsa_1024_a.pub
```

*Eindeutige ID: #1140*
*huschi*
*2006-03-02 15:06*


**(c) 2025 huschi <huschi@huschi.net>**
URL: https://www.huschi.net/index.php?action=artikel&cat=14&id=141&artlang=de