

OpenSSH Tutorial - Teil 1: OpenSSH Server installieren

Google +

Wenn es um sichere Verbindungen zwischen zwei Computern geht, ist SSH (SSH = Secure Shell) das Mittel der Wahl. So lässt sich zum Beispiel der Home- oder Webserver über eine verschlüsselte Verbindung aus der Ferne per SSH steuern. Wie man den OpenSSH Server (Software) auf Linux (Ubuntu) installiert, [einrichtet](#) und [absichert](#) möchte ich euch in der nachfolgenden Artikelserie zeigen.

Als Beispiel Szenario gehen wir davon aus, dass wir einen Linux-Computer haben der als Server dient und zwei weitere Computer, einer mit Ubuntu Linux und einer mit Windows 8, die unsere Arbeitsgeräte bzw. Clients darstellen.

OpenSSH Server installieren

Im ersten Schritt muss die Server Software heruntergeladen werden. Der SSH-Client (zum Verbinden des Servers mit einem weiteren PC) kann bei Bedarf gleich mit installiert werden.

```
1 sudo apt-get update
```

```
2 sudo apt-get install openssh-server openssh-client
```

Nach der Installation muss der OpenSSH-Server eventuell noch händisch gestartet werden. Dies geht mittels:

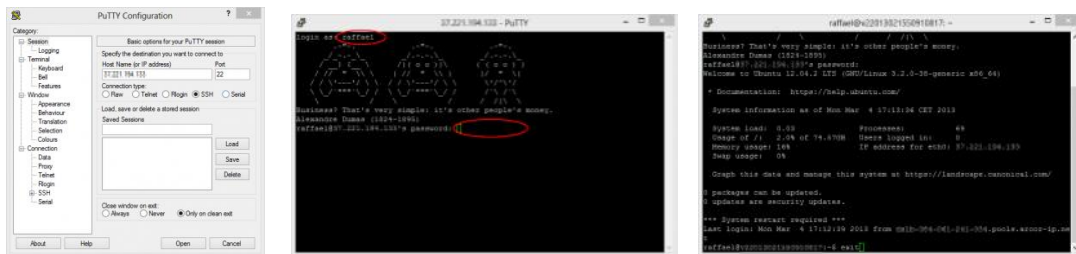
Nun sollte der Server laufen. Für einen ersten Test begeben wir uns an einen der anderen beiden Computer. Um uns nun zu dem Server zu verbinden, benötigen wir nun noch einen SSH-Client für das jeweilige Betriebssystem. Nachfolgend werde ich jeweils auf Windows als auch auf Linux eingehen.

SSH Verbindung unter Windows mittels Putty herstellen

Der vermutlich populärste Client unter Windows ist Putty. Alternativ kann aber auch z.B. OpenSSH für Windows installiert werden, darauf gehe ich aber noch im nächsten Abschnitt ein.

Zuerst wird natürlich einmal Putty selbst benötigt. Den Downloadlink [findet ihr hier](#). Wer lieber eine portable Version für seinen USB-Stick mag, [findet diese hier](#). Die Installation von Putty selbst, sollte unmissverständlich sein.

Nach der Installation starten wir Putty. Im Connection Tab geben wir die IP/den Hostnamen des Servers ein und wählen als Verbindungstyp SSH. Der vorausgewählte Port (22) kann so belassen werden. (22 ist der Standardport für das SSH-Protokoll.) Mit einem Klick auf Connect geht es auch schon los.



Sollte alles geklappt haben, erscheint im Konsolenfenster die Aufforderung zur Eingabe des Usernames sowie des Passworts. Zum Test melden wir uns einmal am Linux Server an und beenden die Verbindung daraufhin mittels des "exit"-Befehls wieder.

SSH Verbindung unter Windows mittels OpenSSH für Windows

OpenSSH für Windows gibt es im Gegenteil zu Putty nicht als portable Version. Das Setup könnt ihr [von hier herunterladen](#). Während der Installation werdet ihr gefragt, ob ihr nur den Client oder auch den Server installieren wollt. Für unser Tutorial reicht der Client. Wenn ihr den Server installiert, euch jedoch später gar nicht auf euren Windows PC per Remote einloggen wollt, dann stellt der Server nur ein unnötiges "Sicherheitsrisiko" dar.

Nach der Installation öffnet ihr die Konsole (Win+R / Ausführen -> "cmd"). Um zu überprüfen, ob die Installation geklappt hat, gebt ihr folgenden Befehl ein.

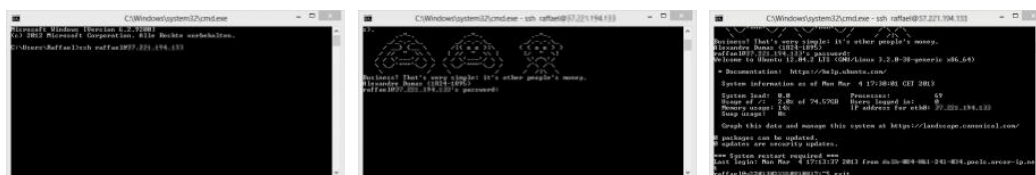
Ihr solltet nun die installierte OpenSSH Version angezeigt bekommen. Erscheint stattdessen nur der Hinweis, dass der Befehl "ssh" ungültig ist, so müsst ihr noch die Umgebungsvariable für Windows anpassen. Dies geht ebenfalls direkt aus der Kommandozeile.

```
1setx PATH "C:\Program Files (x86)\OpenSSH\bin;%path%"
```

(Wobei "C:\Program Files (x86)\OpenSSH\bin" den Pfad zur eurer OpenSSH Installation darstellt und bei euch ggf. abweichen kann!) Achtung: Nachdem ihr den setx-Befehl ausgeführt habt, müsst ihr das cmd-Fenster schließen und ein Neues öffnen. In dem Fenster in dem setx ausgeführt wurde, werden die Änderungen noch nicht beachtet.

Nun können wir uns mit ssh zu unseren Linux Server verbinden. Hier zu ist folgender Befehl nötig:

"user" gibt hierbei den Benutzernamen an, unter dem ihr euch auf dem Linux-Server anmelden wollt. "meine_ip" müsst ihr durch den Hostnamen bzw. die IP-Adresse des Linux-Servers ersetzen.



Wenn alles geklappt hat, solltet ihr nach dem Passwort für "user" gefragt werden. Gebt dieses ein und schließt danach die Verbindung mittels des "exit" Befehls.

SSH Verbindung unter Linux mittels OpenSSH

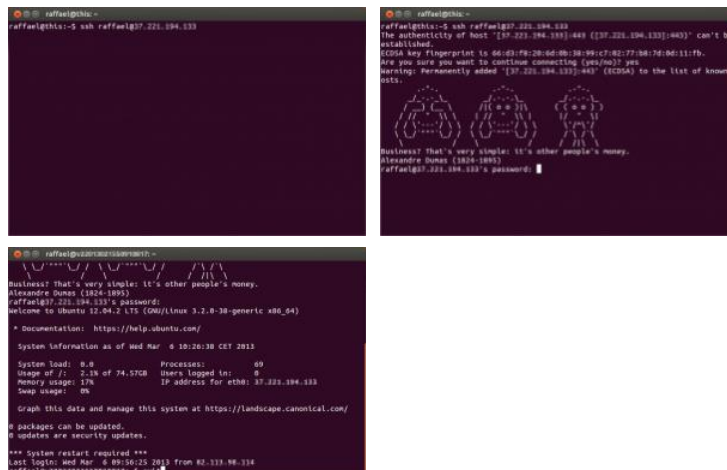
In den meisten Fällen sollte der OpenSSH Client unter Linux schon installiert sein. Dies können wir testen, indem wir probieren die installierte Versionsnummer abzufragen. Hier zu öffnen wir einen Terminal/Shell und geben folgenden Befehl ein.

Wenn der OpenSSH-Client installiert ist, dann sollten wir die Versionsnummer angezeigt bekommen. Ist dies nicht der Fall müssen wir zuerst noch den OpenSSH-Client installieren. Bei auf Ubuntu basierenden Linux-Distributionen geht das zum Beispiel wie folgt.

```
1sudo apt-get install openssh-client
```

Nach der Installation kann es auch schon losgehen. Die Verbindung wird genauso aufgebaut wie bei der OpenSSH-Variante für Windows, welche ich weiter oben im Artikel schon beschrieben habe.

“user” gibt hierbei wieder den Benutzernamen an, mit dem ihr euch auf dem Linux-Server anmelden wollt. “meine_ip” müsst ihr wieder durch den Hostnamen bzw. die IP-Adresse des Linux-Servers ersetzen.



```
raffaello@win-:~$ ssh raffaello@17.221.194.133
raffaello@17.221.194.133:~$ ssh raffaello@17.221.194.133
The authenticity of host '17.221.194.133 (17.221.194.133)' can't be
established.
ECDSA key fingerprint is 66:03:F8:20:6d:0b:38:09:c7:02:77:04:7d:0d:11:7b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '17.221.194.133' (ECDSA) to the list of known
hosts.
Business? That's very simple: It's other people's money.
Alexandre Dumas (1824-1895)
raffaello@17.221.194.133's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
System information as of Wed Mar  6 10:26:38 CET 2013
System load:  0.0      Processes:    69
Usage of /:   2.1% of 74.57GB   Users logged in:  0
Memory usage: 17%      IP address for eth0: 17.221.194.133
Swap usage:   0%

Graph this data and manage this system at https://landscape.canonical.com/

0 packages can be updated.
0 updates are security updates.

*** System restart required ***
Last login: Wed Mar  6 09:50:33 2013 from 82.133.98.134
raffaello@17.221.194.133:~$ exit
```

Auch hier gilt. Wenn alles geklappt hat, solltet ihr mit eurem Linux-Server verbunden und nach dem Passwort für den User auf dem Linux-Server gefragt werden. Wenn ihr eingeloggt seid, könnt ihr die Verbindung mittels des Befehls “exit” wieder trennen.

OpenSSH Tutorial – Teil 2: Grundlegende Absicherung

Google +

Nachdem wir [im ersten Teil](#) den OpenSSH-Server auf dem Linux Rechner aufgesetzt und danach jeweils erfolgreich eine Verbindung mittels Putty, mittels OpenSSH unter Linux und mittels OpenSSH für Windows hergestellt haben, soll es in diesem (zweiten) Teil der Artikelserie darum gehen, wie man seinen OpenSSH-Server absichert.

Standard Port ändern

Als Erstes sollten wir den Port ändern, auf dem der OpenSSH-Server läuft. Doch warum das? Hierbei geht es um einen klassischen Fall von [Sicherheit durch Obskürität](#) (engl. Security by Obscurity). Theoretisch verändert sich der Grad der Sicherheit nicht. Praktisch zeigt sich jedoch, dass die Anzahl der Attacken auf den SSH-Server sinkt. Dies liegt daran, dass viele Angreifer nur auf Port 22, den Standard-SSH-Port, scannen. Ändern wir den Port, fallen wir aus dem Schema. Sicher kann jemand, der speziell unseren Server als Angriffsobjekt auserkoren hat, einen Portscan vornehmen, dennoch lohnt sich die Maßnahme, um die eben angesprochenen "planlosen" Scanner auszuschließen.

Um den Port zu ändern, müssen wir eine Änderung in der `sshd_config` des OpenSSH-Servers vornehmen. Öffnet hierzu einen Terminal auf dem Rechner, auf dem euer OpenSSH-Server läuft.

```
1sudo vim /etc/ssh/sshd_config
```

In der `sshd_config` suchen wir nun die Zeile "Port 22" und ersetzen diese durch "Port X". (Wobei X für einen Port eurer Wahl steht.) Beachtet bei der Portwahl, dass ihr keinen Port wählt, der schon in Benutzung ist. Läuft auf eurem Linux Rechner also zum Beispiel ein Webserver auf Port 80, so eignet sich Port 80 logischerweise nicht mehr als neuer Port für euren OpenSSH-Server.

Nachdem ihr den Port angepasst habt, speichert ihr die Änderungen, schließt die `sshd_config` und startet den OpenSSH-Dienst neu.

Hinweis: Wer mit der Bedienung von VIM nicht klarkommt, kann in den Befehlen auch jeweils das "vim" durch ein "nano" ersetzen.

LogLevel erhöhen

Damit der OpenSSH-Server fehlerhafte Anmeldungen und potenzielle Angriffe protokolliert, müssen wir das Log-Level anpassen. Dies geht ebenfalls in der `sshd_config`.

```
1sudo vim /etc/ssh/sshd_config
```

Standardmäßig loggt der OpenSSH-Server nur wenig mit. Wenn der Verdacht aufkommt, dass der Dienst unter "Beschuss" steht oder gerne detaillierte Logs haben möchte, der sollte das LogLevel in der sshd_config anpassen. Hierzu muss folgende Zeile angepasst werden.

```
1
2
3 LogLevel INFO
4 LogLevel VERBOSE
5
```

Wie auch schon bei der Portänderung muss der OpenSSH-Dienst nach dem Speichern der Änderungen noch neu gestartet werden.

Ist das LogLevel auf "VERBOSE" gestellt, schreibt der OpenSSH-Server die Logs nach "/var/log/auth.log".

Um einen schnellen Blick auf eventuell fehlgeschlagene Logins zu werfen, kann folgender Befehl behilflich sein.

```
1 cat /var/log/auth.log | grep "Failed"
```

(Der obige Befehl kann immer nur eine Hilfe sein und ersetzt natürlich keinen ausführlichen Blick auf das Logfile.)

Root-Login verbieten

Eine weitere Möglichkeit zum Absichern des OpenSSH-Servers ist es, die Anmeldung als "root" zu verbieten und gegebenenfalls sogar nur bestimmte Nutzer zuzulassen. Auch hierzu müssen wieder Änderungen in der sshd_config vorgenommen werden.

```
1 sudo vim /etc/ssh/sshd_config
```

Um den Login als Root zu unterbinden muss die Direktive "PermitRootLogin" angepasst werden.

```
1
2
3 PermitRootLogin yes
4 PermitRootLogin no
5
```

Steht die Direktive auf no, kann sich niemand mehr als root einloggen. Sollte man selbst als Nutzer mal Root-Rechte benötigen, kann man nach dem Login mittels "su" oder "sudo" erhöhte Rechte akquirieren.

Nutzer-Whitelist anlegen

Ein weiterer Schritt ist das Anlegen einer Whitelist mit erlaubten Nutzern. Möchte man zum Beispiel, dass sich nur die Benutzer(-konten) "raffael" und "peter" anmelden können, so muss folgende Zeile in die sshd_config eingefügt werden.

(Ist die Direktive AllowUsers schon in der sshd_config vorhanden, so muss diese natürlich angepasst und nicht ein zweites Mal eingefügt werden.)

Wenn man sogar weiß, dass ein bestimmter Nutzer immer dieselbe IP-Adresse hat (was bei DSL-Anschlüssen ohne Weiteres ja leider nicht der Fall ist), so kann man die Whitelist noch restriktiver gestalten.

```
1AllowUsers raffael@192.168.2.100
```

Die obige IP-Adresse muss dann natürlich durch eure angepasst werden und dient hier nur exemplarisch. Der Aufbau mit Angabe eines Hosts geschieht nach dem Schema: user@host.

OpenSSH Tutorial – Teil 3: Login mit Public- und Private-Key

Google +

Nachdem wir im [ersten Teil](#) einen Open-SSH Server aufgesetzt und im [zweiten Teil](#) die grundlegende Konfiguration zur Absicherung vorgenommen haben, wollen wir nun im dritten Teil die Anmeldung per Schlüsseldatei (nachfolgend auch "Key") konfigurieren.

Die SSH-Key Authentifizierung basiert auf dem Prinzip der [asymmetrischen Verschlüsselung](#). Wir benötigen also ein Schlüsselpaar aus öffentlichem (public) und privatem (private) Schlüssel. Der öffentliche Schlüssel wird dann auf dem OpenSSH-Server hinterlegt und der private Schlüssel geht an den Client.

Oberstes Gebot dabei ist es, den privaten Schlüssel wirklich immer geheim zu halten. Kommt jemand anderes in den Besitz des privaten Schlüssels, kann er sich damit auf eurem Server einloggen.

Als kleine Hürde kann man den Private-Key selbst auch noch einmal mit einem Passwort versehen. Dennoch gilt, ist der Private-Key nicht mehr "privat", so sollte schnellstens dafür gesorgt werden, dass das Gegenstück, der entsprechende Public-Key, vom OpenSSH-Server entfernt wird.

Um ein Schlüsselpaar zu generieren, gibt es mehrere Möglichkeiten. Zwei davon möchte ich nachfolgend weiter erläutern. Eine Möglichkeit ist die Erstellung mittels des Kommandozeilen-Tools "ssh-keygen". Dieses kommt bei der Installation von openssh mit und ist somit unter Linux verfügbar. Bei dem OpenSSH for Windows Paket, war zum Zeitpunkt meiner Tests, das Tool leider nicht dabei.

Wer lieber mit einer grafischen Oberfläche arbeitet, sollte sich die zweite vorgestellte Möglichkeit ansehen. Mittels des Programms "PuttyGen" lässt sich das benötigte Schlüsselpaar auch ohne Kommandozeile und unter Windows erstellen.

Schlüsselpaar mittels ssh-keygen erzeugen

Eine Möglichkeit ein Schlüsselpaar zu erzeugen ist die Verwendung von ssh-keygen. Dieses Tool ist jedoch leider nicht bei der OpenSSH für Windows Installation enthalten. Entweder besorgt ihr euch also Zugriff auf einen Linux Rechner oder führt ssh-keygen auf dem Linux-Server aus, auf dem euer OpenSSH-Server läuft.

Hinweis: Dies solltet ihr aber nur tun, wenn ihr wirklich sicher seid, dass niemand anderes auf den Server Zugriff hat. Das wäre zum Beispiel bei einem Mediaserver bei euch im LAN der Fall, wenn ihr alle Netzwerknutzer kennt und der Server keinen Internetzugriff hat. Vor allem bei Shared-Server-Umgebungen solltet ihr hingegen Vorsicht walten lassen und den Key lieber lokal/clientseitig generieren lassen.

Um ein neues Schlüsselpaar mit ssh-keygen zu generieren, benötigt ihr folgenden

Befehl.

```
1ssh-keygen -t rsa -b 4096
```

Der Parameter -t gibt hierbei das Verschlüsselungsverfahren an. (Für SSH Version 2 wäre statt RSA, wie hier im Beispiel, auch DSA gültig.) Der Parameter -b gibt die Schlüssellänge in Bits an.

Während der Ausführung des Befehls werdet ihr nach einem Passwort gefragt. Hier gilt es, einen guten Kompromiss zwischen Komplexität und Aufwand für die jeweilige Eingabe des Passworts zu finden.

Wichtig: Zwar besteht auch die Option gar kein Passwort einzugeben, davon ist aber dringend abzuraten. Gebt ihr kein Passwort ein, so genügt einem Angreifer, wie in der Einleitung dieses Artikels beschrieben, lediglich der Schlüssel, um sich auf eurem Server einzuloggen! Vergebt ihr ein Passwort, so muss der Angreifer dieses erst knacken, bevor er sich einloggen kann. Der Zeitraum um das Passwort der Schlüsseldatei zu knacken sollte wiederum genügen, um den Public-Key von eurem Server zu entfernen und ein neues Schlüsselpaar anzulegen.

Nach der Generierung des Schlüsselpaars habt ihr nun 2 Dateien erhalten.

Die Datei mit der Endung .pub ist euer Public-Key und gehört auf den Server. Falls noch nicht vorhanden legt ihr einen Ordner namens ".ssh" in eurem Home-Verzeichnis an und kopiert die rsa_id.pub Datei in diesen Ordner.

```
1sudo mkdir -p ~/.ssh
```

```
2sudo mv id_rsa.pub ~/.ssh
```

Abschließend müsst ihr euren Public-Key noch in eine Datei namens authorized_keys einfügen. Dies könnt ihr mit folgendem Befehl tun.

```
1cd ~/.ssh
```

```
2sudo cat id_rsa.pub >> authorized_keys
```

Schlüsselpaar mittels PuttyGen erzeugen

Wer keinen Zugriff auf einen lokalen Linux-Rechner hat und den Key nicht serverseitig generieren möchte, der kann z.B. auch [PuttyGen](#) nehmen. Dies ist ein Schlüsselgenerator aus der Putty-Suite. PuttyGen könnt ihr kostenlos herunterladen. Eine Installation ist nicht nötig.



Nach dem Start wählt ihr im unteren Panel die Länge des Schlüssels in Bits und das Verschlüsselungsverfahren aus. Danach klickt ihr auf "Generate", um die Schlüsselgenerierung zu starten. Um die Generierung zu beschleunigen, könnt ihr mit

der Maus willkürlich auf der PuttyGen-Oberfläche umherfahren. Die Bewegungen werden als Zufallsdaten genommen und in die Berechnung des Schlüssels mit einbezogen.

Ist der Schlüssel erstellt, solltet ihr noch ein Passwort für den Private-Key anlegen. Dies schützt euch wie gesagt bei Diebstahl des Schlüssels. (Zumindest für eine gewisse Zeit.) Das Passwort schreibt ihr einfach in die Felder "Key passphrase" und "Confirm passphrase", welche nach der Generierung des Schlüssels sichtbar werden.

Nach dem Festlegen des Passworts können Private- sowie Public-Key mittels der beiden Save-Buttons gespeichert werden. Der Public-Key muss nun auf euren OpenSSH-Server übertragen werden (z.B. per USB-Stick, SSH/SCP, FTP etc.).

Einmal auf dem Server angekommen, muss der Key in eine Datei namens "authorized_keys", die sich in dem Ordner ~/.ssh befinden sollte, geschrieben werden.

Habt ihr noch keine Keys dort gespeichert, so müsst ihr den Ordner erst anlegen.

```
1sudo mkdir -p ~/.ssh
2sudo mv id_rsa.pub ~/.ssh
```

Anschließend fügt ihr den Public-Key dann in die authorized_files Datei ein.

```
1cd ~/.ssh
2sudo cat id_rsa.pub >> authorized_keys
```

Anpassen der OpenSSH Server-Konfiguration

Im letzten Schritt muss noch die Konfiguration des OpenSSH-Server angepasst werden, sodass dieser den Login per Schlüsseldatei erlaubt. Öffnet hier zu die sshd_config.

```
1sudo vim /etc/ssh/sshd_config
```

Und ändert die Zeile, die mit "PasswordAuthentication" beginnt, wie folgt ab.

```
1PasswordAuthentication no
```

Durch diese Einstellung wird der Login per reinem Passwort untersagt und nur noch Nutzer mit Schlüsseldateien können sich einloggen. Somit werden sämtliche Bruteforce- und Passwortlisten-Attacken unbrauchbar.

Um die Änderungen zu übernehmen müsst ihr die sshd_config noch speichern, schließen und dann die geänderten Einstellungen einmal im OpenSSH-Dienst neuladen. Dies geht wie folgt.

Nun steht serverseitig alles, was wir für den Login mittels Schlüssel-Dateien brauchen. Nachfolgend möchte ich noch ganz kurz zeigen, welche Parameter bzw. Einstellungen nötig sind, um sich mit einem Client und Schlüssel-Datei auf einem SSH-Server einzuloggen.

Mittels Private-Key und OpenSSH einloggen

Der Login mit einem Private-Key funktioniert sowohl mit OpenSSH als auch mit

OpenSSH for Windows. Der benötigte Befehl ist dabei fast identisch mit dem, welchen ich euch [im ersten Teil der Artikelserie](#) gezeigt habe. Es kommt lediglich noch ein Parameter "-i" hinzu.

```
1ssh meinbenutzername@meineip -p meinport -i "PfadZuMeinemPrivateKey"
```

"meinbenutzername" ist der Name des Accounts auf dem SSH-Server, in den ihr euch einloggen wollt.

"meineip" kann entweder die IP-Adresse des Servers sein oder alternativ auch der Hostname.

"-p meinport" ist optional und sollte angegeben werden, wenn ihr den Port geändert habt, auf dem euer SSH-Server läuft. ""PfadZuMeinemPrivateKey" muss je nach Betriebssystem angegeben werden. Unter Windows mit Backslashes und unter Linux logischerweise mit Slashes. Beispiel Windows: "C:\meinekeys\id_rsa". Beispiel Linux: "~/meinekeys/id_rsa".

Mittels Private-Key und Putty einloggen

Um sich mit Putty und dem eigenen Private-Key einzuloggen, muss der Private-Key in den Einstellungen von Putty angegeben werden. Dies geht im Menüpunkt "Connection->SSH->Auth". Jedoch benötigt Putty den Key in seinem eigenen PPK Format. Sollte euer Private-Key also mit ssh-keygen generiert worden sein, so müsst ihr diesen zuerst umwandeln. (Wenn ihr dies schon getan habt könnt ihr den nächsten Absatz überspringen.)



Um den Private-Key in das PPK-Format umzuwandeln, benötigen wir wieder das Tool "PuttyGen". Nach dem Start von "PuttyGen" kann im Menü über "Conversions->Import key" der Private-Key eingelesen werden. Mittels des Buttons "Save Private-Key" lässt dieser sich dann im PPK-Format speichern.

Wenn ihr nun den Private-Key im PPK-Format vorliegen habt, öffnet ihr Putty. Dort wählt ihr eure Verbindungseinstellungen aus oder gebt neue an und wechselt vor dem Login in den Menüpunkt "Connection->SSH->Auth". Hier könnt ihr in dem Feld "Private key file for authentication:" den Pfad zu eurem Private-Key angeben. Das war es auch schon. Wenn ihr euch nun per Putty einloggt, wird der Login mittels Key-File vorgenommen.



Über den Autor: Dieser Artikel, sowie [358](#) andere Artikel auf [code-bude.net](#), wurden von [Raffael](#) geschrieben. – Seit 2011 blogge ich hier über Programmierung, meine Software, schreibe Tutorials und versuche mein Wissen, so gut es geht, mit meinen Lesern zu teilen. Zudem schreibe ich auf [derwirtschaftsinformatiker.de](#) über Themen meines Studiums. // [Email](#) • [Facebook](#) • [Twitter](#)