HOWTO zu SSH (Secure Shell)

$Id: ssh-HOWTO.txt,v 1.62 2020/02/24 06:36:18 tsbirn Exp $

Dieses Dokument beschreibt Grundlagen und Einsatz von SSH (Secure Shell)
unter UNIX/Linux (sicheres telnet, rcp, rsh, rlogin, ftp + "VPN fÃ¼r Arme").

INHALTSVERZEICHNIS

---

1) EinfÃ¼hrung

SSH (Secure Shell) stellt mit "ssh", "scp" und "sftp" (Clients) + "sshd"
(Server) eine gesicherte (verschlÃ¼sselte) Kommunikation Ã¼ber unsichere Netze
zur VerfÃ¼gung, und bildet einen sicheren Ersatz fÃ¼r die Standard-UNIX-Tools
"telnet", "ftp", "rlogin", "rcp", "rsh", "rcmd", ... (Remote Login, Remote
Kommando-AusfÃ¼hrung bzw. Dateitransfer) mit folgenden Eigenschaften:

  * LeistungsfÃ¤hige Authentifizierung von Client UND Server
  * Sichere VerschlÃ¼sselung der Ã¼bertragenen Daten (und Anmeldung)
  * IntegritÃ¤tssicherung (Verifizierung) der Ã¼bertragenen Daten
    (verhindert "Man-in-the-middle" Attacken)

SSH sollte daher aus SicherheitsgrÃ¼nden alle oben aufgezÃ¤hlten veralteten
UNIX/LINUX-Remote-Kommandos ersetzen. Bei den meisten LINUX/UNIX-Versionen ist
SSH inzwischen Standard, auf ftp/telnet wird von vornherein verzichtet:

Weiterhin unterstÃ¼tzt SSH den Schutz von X11-Verbindungen (X Window System)
und die Weiterleitung ("Port-Forwarding") von beliebigen TCP-Verbindungen
Ã¼ber einen kryptografisch gesicherten Kanal und bietet somit VPN-FunktionalitÃ¤t
("VPN fÃ¼r Arme"). Kurzer Funktionsumfang:

  * Login auf Remote-Host
  * AusfÃ¼hren von Kommandos auf Remote-Host (interaktiv / nicht-interaktiv)
  * Kopieren von Dateien zwischen Hosts
  * Port-Forwarding/Tunneling (VPN)
  * Komprimierung der Daten wÃ¤hrend Transfer

Eigenschaften:

  * Automatische und transparente VerschlÃ¼sselung der gesamten Kommunikation
  * Sechs Verfahren der Client-Authentifizierung gegenÃ¼ber dem Server mÃ¶glich
  * RSA-basierte Authentifizierung (IP-, Routing-, DNS-Spoofing nicht mÃ¶glich)
  * Systemweite + benutzerbezogene Konfigurations-Dateien
  * Ã¼bertragen von BinÃ¤rdaten (+ Komprimierung) mÃ¶glich
  * Arbeitet gut mit anderen Tools zusammen
    (z.B. "rsync", "X", "cvs", "svn", "unison").
  * Port 22 (SSH-Port) fÃ¼r Kommunikation verwendet

Bekannte Implementierungen:

* OpenSSH: Kommt ohne patentierte Algorithmen aus (--> http://www.openssh.com)
* Windows-Client: Putty (suchen in Google nach: putty download)

2) Ã-^\bersicht Konfigurations- und SchlÃ¼sseldateien

Es gibt zentrale (rechnerspezifische) Konfigurations-Dateien im Verz. "/etc/ssh" und benutzerspezifische im Verz. "~/.ssh" jedes Benutzers. Wird eine Einstellung in beiden Konfigurations-Dateien getroffen, dann Ã¼berschreibt die benutzerspezifische Einstellung die zentrale. Existiert die benutzerspezifische Konfigurations-Datei nicht, dann gilt die zentrale vollstÃ¤ndig.

Vorrang SSH-Client-Konfiguration (von oben nach unten absteigend):

-o OPT=VAL ...	Kommandozeilen-Option (mehrfach angebbar)
-F FILE	Konf.datei (zentrale + benutzerspez. ignoriert)
~/.ssh/config	Benutzerspezifische Konfiguration
/etc/ssh/ssh_config	Zentrale Konfiguration

Vorrang SSH-Server-Konfiguration (von oben nach unten absteigend):

-o OPT=VAL ...	Kommandozeilen-Option (mehrfach angebbar)
-f FILE	Zentrale Konf.datei explizit angeben
/etc/ssh/sshd_config	Zentrale Konfiguration (Std falls -f FILE fehlt)

Zur VerschlÃ¼sselung der Kommunikation gibt es mind. ein SchlÃ¼sselpaar pro Rechner und mind. eines pro Benutzer. Da es mehrere VerschlÃ¼sselungsverfahren gibt (RSA/DSA/ECDSA/ED25519 --> veraltet/modern, unsicher/sicher, patentbehaftet/patentfrei), kÃ¶nnen auch bis mehrere SchlÃ¼sselpaare existieren, von denen pro Authentifizierungsvorgang eines ausgewÃ¤hlt wird.

* SchlÃ¼sselpaare fÃ¼r Rechner werden bei der Installation des Rechners bzw. des SSH-Paketes AUTOMATISCH angelegt (per "ssh-keygen").

* SchlÃ¼sselpaare fÃ¼r Benutzer mÃ¼ssen MANUELL pro Benutzer per Kommando "ssh-keygen" angelegt werden.

Ein zusammengehÃ¶rendes SchlÃ¼sselpaar wird in 2 Dateien abgelegt:

```
XXX       # Privater Teil (bleibt auf Rechner, nur fÃ¼r EigentÃ¼mer lesbar)
XXX.pub   # Ã-^Vffentlicher Teil ("pub"=public, darf beliebig verteilt werden)
```

Konfigurations-Dateien der beiden Kommunikations-Partner:

Lokal (Client)		Remote (Server)
Client: ssh, scp, sftp Putty, ...	<==========> Kommunikation	Server-Daemon: sshd Firewall mit offenem SSH-Port 22
/etc/ssh/ssh_config ~/.ssh/config (ACHTUNG: kein d=daemon)	Konfiguration	/etc/ssh/moduli /etc/ssh/sshd_config (ACHTUNG: d=daemon)
/etc/ssh/ssh_known_hosts ~/.ssh/known_hosts	<========== Server-Authentifiz.	/etc/ssh/ssh_host_rsa_key (P2) /etc/ssh/ssh_host_rsa_key.pub /etc/ssh/ssh_host_dsa_key /etc/ssh/ssh_host_dsa_key.pub /etc/ssh/ssh_host_ecdsa_key /etc/ssh/ssh_host_ecdsa_key.pub /etc/ssh/ssh_host_ed25519_key /etc/ssh/ssh_host_ed25519_key.pub /etc/ssh/ssh_host_key (P1) /etc/ssh/ssh_host_key.pub
~/.ssh/id_rsa (P2) ~/.ssh/id_rsa.pub ~/.ssh/id_dsa ~/.ssh/id_dsa.pub ~/.ssh/id_ecdsa ~/.ssh/id_ecdsa.pub ~/.ssh/id_ed25519 ~/.ssh/id_ed25519.pub ~/.ssh/identity (P1) ~/.ssh/identity.pub	==========> Client-Authentifiz.	~/.ssh/authorized_keys

```
  |                           |============>| /etc/ssh/sshrc                    |
  |                           |Login-Shell  | ~/.ssh/rc                         |
  |                           |Konfiguration| ~/.ssh/environment                |
  +---------------------------+             +-----------------------------------+
  |                           |============>| ~/.hushlogin                      |
  |                           |   Login-    | /etc/motd                         |
  |                           | verhalten   | /etc/nologin                      |
  +---------------------------+             +-----------------------------------+
  |                           |============>| ~/.rhosts                         |
  |                           | Host-based  | ~/.shosts                         |
  |                           |   Login     | /etc/hosts.equiv                  |
  |                           |             | /etc/shosts.equiv                 |
  +---------------------------+             +-----------------------------------+
  |                           |============>| /etc/hosts.allow                  |
  |                           | TCP-Wrapper | /etc/hosts.deny                   |
  +---------------------------+             +-----------------------------------+

HINWEISE:

* Server-Konfigurations-Datei "sshd_config" (MIT d=daemon!) und
  Client-Konfigurations-Datei "ssh_config" (OHNE d=daemon!) NICHT verwechseln!

* BesitzverhÃ¤ltnisse und Zugriffsrechte sÃ¤mtlicher Konfigurations-Dateien
  und ihrer Verz.pfade mÃ¼ssen einige Bedingungen erfÃ¼llen, damit SSH eine
  schlÃ¼sselbasierte Anmeldung erlaubt --> 8) Tipps


3) Prinzipielle Arten von VerschlÃ¼sselung
-----------------------------------------
a) SYMMETRISCH: Gleicher SchlÃ¼ssel zum Ver- und zum EntschlÃ¼sseln:
   - SchlÃ¼sselaustausch schwierig (muss Ã¼ber geheimen Kanal erfolgen)
   - SchlÃ¼ssel muss geheim gehalten werden
   + SchlÃ¼ssel kurz (128-256 Bit)
   + Schnell (SchlÃ¼sselgenerierung und VerschlÃ¼sselung)

b) ASYMMETRISCH: Ein SchlÃ¼sselpaar bestehend aus einem  Ã¶ffentlichen
   (public key) und einem privaten (private key) Teil:
   + SchlÃ¼sselaustausch einfach (den public key darf JEDER kennen!)
   + Nur privater SchlÃ¼ssel muss geheim gehalten werden
   - SchlÃ¼ssel lang (1024-16384 Bit)
   - Langsam (SchlÃ¼sselgenerierung und VerschlÃ¼sselung)

c) HYBRID: Verbindung der Vorteile beider Verfahren.
   Beim Verbindungsaufbau wird Ã¼ber eine asymmetrische VerschlÃ¼sselung ein
   symmetrischer "SitzungsschlÃ¼ssel" vereinbart und darÃ¼ber die eigentliche
   Kommunikation durchgefÃ¼hrt.

SSH verwendet hybride VerschlÃ¼sselung: Das asymmetrische Verfahren RSA (Rivest,
Shamir, Adelman), DSA (Digital Signature Algorithm), ECDSA (Elliptic Curve
Digital Signature Algorithm) oder ED25519 (Elliptic D?) wird zu
Authentifizierung der Gegenstellen verwendet. Danach tauscht SSH einen
symmetrischen SchlÃ¼ssel aus, der die Sitzungsdaten verschlÃ¼sselt. Dieser
SchlÃ¼ssel wird in bestimmten ZeitabstÃ¤nden (etwa alle 60 Minuten) erneuert.

Grund fÃ¼r den stÃ¤ndigen Wechsel des SitzungsschlÃ¼ssels: Eine aufgezeichnete
SSH-Sitzung kÃ¶nnte, wenn der zugehÃ¶rige private SchlÃ¼ssel spÃ¤ter irgendwie
bekannt wird, nachtrÃ¤glich entschlÃ¼sselt werden. Aufgrund des Wechsels und
"Vergessen" des alten SitzungsschlÃ¼ssels ist eine nachtrÃ¤gliche EntschlÃ¼sselung
der Sitzung nicht oder nur fÃ¼r kleine Abschnitte der Sitzung mÃ¶glich
("Replay"-Angriff).


4) SSH installieren, aktivieren und testen
------------------------------------------


4.1) SSH-Daemon "sshd"
----------------------
PrÃ¼fen, ob SSH installiert ist:

  rpm -qa | grep "ssh"     # --> z.B. "openssh-3.0p1-33 + openssh-askpass-3.0p1-33"
  dpkg -l | grep "ssh"     # --> z.B. "openssh-server"

Auf dem eigenen Rechner muss der SSH-Client "ssh" (oder "Putty") installiert
sein, auf dem Remote-Rechner muss ein SSH-Server "sshd" laufen (lauscht auf
Port 22). Evtl. Firewalls mÃ¼ssen Verbindungen auf Port 22 erlauben bzw.
durchlassen.

SSH-Daemon auf Remote-Rechner (Server) aktivieren:

  rcsshd start              # TemporÃ¤r oder
  /etc/init.d/sshd start    # TemporÃ¤r oder
```

```
  initctl ssh start         # TemporÃ¤r (Upstart) oder
  chkconfig -a sshd         # Permanent (-a=add)

SSH-Daemon auf Remote-Rechner (Server) deaktivieren:

  rcsshd stop               # TemporÃ¤r oder
  /etc/init.d/sshd stop     # TemporÃ¤r oder
  initctl ssh stop          # TemporÃ¤r (Upstart) oder
  chkconfig -d sshd         # Permanent (-d=del)

ÃM-^\berprÃ¼fen ob SSH-Daemon "sshd" aktiv ist:

  /etc/init.d/sshd status   # (SysV)    oder
  initctl ssh status        # (Upstart) oder
  service ssh status        # (Systemd) oder
  rcsshd status             # (Suse)    oder
  ps -aux | grep "sshd"     #

Bei ÃM-^Dnderungen an der SSH-Server-Konfigurations-Datei "/etc/ssh/sshd_config"
MUSS der SSH-Server neu gestartet und SOLLTEN vorhandene Verbindungen neu
aufgebaut werden (bleiben mit alten Einstellungen bestehen):

  /etc/init.d/sshd restart  # (SysV)    oder
  initctl ssh restart       # (Upstart) oder
  service ssh restart       # (Systemd) oder
  rcsshd restart            # (Suse)

Alternativ kann auch nur die Konfiguration des SSH-Servers neu geladen werden:

  /etc/init.d/sshd reload   # (SysV)    oder
  initctl ssh reload        # (Upstart) oder
  service ssh reload        # (Systemd) oder
  rcsshd reload             # (Suse)

Zum Testen des SSH-Daemons das Debugging einschalten (verschiedene Stufen),
er startet dann nicht als Daemon und nur 1x. Seine Fehlermeldungen werden auf
dem Fehlerkanal "stderr" ausgegeben:

  sshd -d     # Debug-Meldungen
  sshd -dd    # Mehr Debug-Meldungen
  sshd -ddd   # Noch mehr Debug-Meldungen (mit sshd_config-Parametern!)

Neben BENUTZERSPEZIFISCHEN SchlÃ¼sseln gibt es auch RECHNERSPEZIFISCHE
SchlÃ¼ssel, diese werden in der Regel bei der Systeminstallation erzeugt.
SÃ¤mtliche SchlÃ¼ssel sollten unbedingt an einer vertraulichen Stelle gesichert
werden (z.B. ausgedruckt und in einem verschlossenen Umschlag hinterlegt
werden), damit sie bei einem Systemfehler oder einer Neuinstallation wieder
hergestellt werden kÃ¶nnen! Durch sie kann bestimmt werden, auf welche Rechnern
Ã¼berhaupt eine SSH-Verbindung erlaubt ist:

  +-------------------------------+-------------------------------------+
  | SchlÃ¼ssel-Datei               | Version                             |
  +-------------------------------+-------------------------------------+
  | /etc/ssh/host_key_rsa         | 2 (privater RSA2-SchlÃ¼ssel)         |
  | /etc/ssh/host_key_rsa.pub     | 2 (Ã¶ffentlicher RSA2-SchlÃ¼ssel)     |
  | /etc/ssh/host_key_dsa         | 2 (privater DSA-SchlÃ¼ssel)          |
  | /etc/ssh/host_key_dsa.pub     | 2 (Ã¶ffentlicher DSA-SchlÃ¼ssel)      |
  | /etc/ssh/host_key_ecdsa       | 2 (privater ECDSA-SchlÃ¼ssel)        |
  | /etc/ssh/host_key_ecdsa.pub   | 2 (Ã¶ffentlicher ECDSA-SchlÃ¼ssel)    |
  | /etc/ssh/host_key_ed25519     | 2 (privater ED25519-SchlÃ¼ssel)      |
  | /etc/ssh/host_key_ed25519.pub | 2 (Ã¶ffentlicher ED25519-SchlÃ¼ssel)  |
  +-------------------------------+-------------------------------------+
  | /etc/ssh/host_key             | 1 (privater RSA1-SchlÃ¼ssel)         |
  | /etc/ssh/host_key.pub         | 1 (Ã¶ffentlicher RSA1-SchlÃ¼ssel)     |
  +-------------------------------+-------------------------------------+


4.2) SSH-Protokoll
------------------
Es gibt 2 Protokoll-Versionen von SSH:

  +---------+--------------------------+------------+
  | Version | VerschlÃ¼sselung          | Protokoll  |
  +---------+--------------------------+------------+
  | SSHv2   | RSA, DSA, ECDSA, ED25519 | P2         |
  | SSHv1   | RSA                      | P1         |
  +---------+--------------------------+------------+

Sie unterscheiden sich in der Aushandlung des symmetrischen SchlÃ¼ssels
(sicherheitstechnische und patentrechtliche GrÃ¼nde) und im Protokollablauf
(Kommunikation zwischen Client und Server). ACHTUNG: Die Version "SSHv1" sollte
aus SicherheitsgrÃ¼nden nicht mehr verwendet werden:
```

```
  1) Protokoll Version 1 (SSH1)
     * Client baut Verbindung zum Server auf Port 22 auf
     * VerstÃ¤ndigung auf Protokollversion und sonstige Kommunikations-Parameter
     * Transfer: Langlebiger Host-Key + kurzleb. Session-Key (1x pro h geÃ¤ndert)
     * Bekannt?
     * 256-Bit-Session-Key erzeugt, verschlÃ¼sselt und an Server Ã¼bermittelt
     * Server entschlÃ¼sselt Session-Key
     * Session-Key wird fÃ¼r symmetrischen VerschlÃ¼sselung benutzt

  2) Protokoll Version 2 (SSH2)
     * Client baut Verbindung zum Server auf Port 22 auf
     * VerstÃ¤ndigung auf Protokollversion und sonstige Kommunikations-Parameter
     * Transfer: Langlebiger Host-Key
     * Diffie-Hellman SchlÃ¼ssel-Vereinbarung --> geheimer Session-Key
     * Session-Key wird fÃ¼r symmetrischen VerschlÃ¼sselung benutzt

Welche Protokolle ein SSH-Server spricht, ist seiner Antwort auf den Befehl
"telnet HOST 22" zu entnehmen (erst Protokoll-Version, dann Implementierung-Version
- schlecht, wie vermeiden? z.B. "SSH-2.0-OpenSSH_6.9"):

  "SSH-1.5-OpenSSH_..."    # --> Nur SSHv1
  "SSH-1.99-OpenSSH_..."   # --> SSHv1 und SSHv2
  "SSH-2.0-OpenSSH_..."    # --> Nur SSHv2

Die wichtigsten Server-Einstellungen in "sshd_config":

  +----------------------+-----------------------------------------------------+
  | Port          22     | Standard-Port (fÃ¼r interne Anwendung Ã¤nderbar)   |
  | Protocol      2      | Probier-Reihenfolge Protokoll-Vers. (nicht 2,1!) |
  | ListenAddress 0.0.0.0| Std: Auf allen NW-Karten lauschen                   |
  | ForwardX11    yes    | X11-Forwarding aktiv                                |
  +----------------------+-----------------------------------------------------+


4.3) SSH-Client "ssh"
---------------------
Welche Protokoll-Version fÃ¼r die SSH-Verbindung verwendet wird (ein Server kann
beide anbieten), legt der Client beim Verbindungsaufbau fest:

  ssh -1 ...   # Protokoll Version 1
  ssh -2 ...   # Protokoll Version 2

Testen ob ein HOST eine SSH-Verbindung prinzipiell annimmmt:

  telnet HOST 22   # Antwort z.B. "SSH-1.99-OpenSSH_3.0p1"
                   # oder         "SSH-2.0-OpenSSH_5.1"

WÃ¤hrend einer SSH-Anmeldung an einem Remote-Rechner kÃ¶nnen Kommandos an den
SSH-Client abgesetzt werden, die durch das Sonderzeichen "~" eingeleitet werden
mÃ¼ssen. Diese "Escape-Sequenzen" dienen zur interaktiven Bedienung der Sitzung
und sind nur am Zeilenanfang eingebbar (ansonsten hat "~" normale Bedeutung):

  +-----+-------------------------------------------------------------------+
  | ~.  | Verbindung abbrechen (alle darÃ¼ber gemultiplexten Sitzungen)      |
  | ~B  | BREAK an Remote-Rechner senden                                    |
  | ~C  | Kommandozeile Ã¶ffnen (SSH-Optionen angeben)                       |
  | ~R  | Neuen SitzungsschlÃ¼ssel anfordern (nur SSH2)                      |
  | ~^Z | Anhalten (^Z=Strg-Z, mit "fg" weiterlaufen lassen)                |
  | ~#  | Weitergeleitete Verbindungen auflisten (forwarded)                |
  | ~&  | In Hintergrund stellen (auf Verb.Ende warten, Tunnel noch genutzt)|
  | ~?  | Hilfe zu Escape-Sequenzen anzeigen (diese Liste)                  |
  | ~~  | Tilde-Zeichen selbst eingeben (2x tippen notwendig)               |
  +-----+-------------------------------------------------------------------+

Die wichtigsten Client-Einstellungen in "ssh_config":

  +-------------------------+-----------------------------------------------+
  | Port                22  | Standard-Port (fÃ¼r interne Anwendung Ã¤nderbar)   |
  | Protocol            2   | Probier-Reihenfolge Protokoll-Vers. (nicht 2,1!) |
  | ForwardX11          yes | X11-Forwarding aktiv                          |
  | TCPKeepAlive        yes | TCP Keep alive Meldungen senden               |
  | ServerAliveInterval 300 | SSH-Keep alive Meldungen alle 5 Minuten senden|
  | EscapeChar          ~   | Escape-Zeichen                                |
  +-------------------------+-----------------------------------------------+


5) Einsatz von SSH
------------------


5.1) Anmelden per SSH-Client
```

```
---------------------------
Per SSH-Client an Remote-Rechner anmelden (aktueller oder anderer Benutzer):

  ssh localhost                   # Auf eigenem Rechner als aktueller Benutzer
  ssh 172.23.19.56                # Als aktueller Benutzer auf Remote-Rechner
  ssh -l tom 172.23.10.56         # Als anderer Benutzer (-l=login)
  ssh tom@172.23.10.56            # Als anderer Benutzer (@=at)
  ssh -p 2222 tom@172.23.10.26    # Auf anderem Port (-p=port)

Test des Verbindungs-Aufbaus (-v=verbose, auch mehrfach -vv, -vvv, ...):

  ssh -v   tom@172.23.10.56   # Meldungen zum Verbindungsaufbau
  ssh -vv  tom@172.23.10.56   # Mehr Meldungen zum Verbindungsaufbau
  ssh -vvv tom@172.23.10.56   # Noch mehr Meldungen zum Verbindungsaufbau

Beim ERSTEN Verbindungsaufbau zu einem unbekannten Server, d.h. dessen
Host-SchlÃ¼ssel auf Client-Seite unbekannt ist (nicht in "~/.ssh/known_hosts"
enthalten), folgende Meldung durch Eintippen von "yes" beantworten, um den
Host-SchlÃ¼ssel ohne Verifizierung zu akzeptieren (unsicher!):

  The authenticity of host '192.168.1.252 (192.168.1.252)' can't be established.
  RSA key fingerprint is 8a:58:19:a1:27:95:2b:e4:e2:3e:5f:41:c7:58:2a:d2.
  Are you sure you want to continue connecting (yes/no)?

ACHTUNG: Text "yes" ist GENAU SO einzutippen und mit <RETURN> abzuschlieÃM-^_en,
         wenn die Verbindung zum unbekannten Server aufgebaut und sein
         Host-SchlÃ¼ssel in "~/.ssh/known_hosts" eingetragen werden soll.


5.2) Server-Authentifizierung
-----------------------------
Wird die 1. Verbindung zu einem Host mit "yes" (3 Buchstaben!) akzeptiert, dann
wird in der lokalen Datei "~/.ssh/known_hosts" des aktuellen Benutzers der
rechnerspezifische Ã M-^VFFENTLICHE SchlÃ¼ssel der Gegenstelle eingetragen (mit
Zuordnung zu einer IP):

  +-----------------------------------+----------------------------+
  | SchlÃ¼sseldatei                    | Version                    |
  +-----------------------------------+----------------------------+
  | /etc/ssh/ssh_host_rsa_key.pub     | 2 (Ã¶ffentlicher SchlÃ¼ssel) |
  | /etc/ssh/ssh_host_dsa_key.pub     | 2 (Ã¶ffentlicher SchlÃ¼ssel) |
  | /etc/ssh/ssh_host_ecdsa_key.pub   | 2 (Ã¶ffentlicher SchlÃ¼ssel) |
  | /etc/ssh/ssh_host_ed25519_key.pub | 2 (Ã¶ffentlicher SchlÃ¼ssel) |
  | /etc/ssh/ssh_host_key.pub         | 1 (Ã¶ffentlicher SchlÃ¼ssel) |
  +-----------------------------------+----------------------------+

Das ist unsinnig fÃ¼r Rechner, die stÃ¤ndig ihre IP-Adressen Ã¤ndern. In der
Konfigurations-Datei "/etc/ssh/ssh_config" bzw. "~/.ssh/config" kann daher das
Verhalten eingestellt werden:

  StrictHostKeyChecking Ask          # "Yes" oder "No" mÃ¶glich (neben "Ask")

Die Einstellung "Ask" ist bequem aber auch GEFÃM-^DHRLICH, da der Host-SchlÃ¼ssel
kompromittiert sein kÃ¶nnte ("man-in-the-middle"-Angriff). Daher sollte er Ã¼ber
einen getrennten Kanal transportiert werden (z.B. USB-Stick, Diskette, CDROM).

Zumindest sollte aus dem Ã¼bertragenen Host-SchlÃ¼ssel sein "Fingerprint" (16
Byte lange Hexadezimalzahl) gebildet und mit dem des Original-SchlÃ¼ssels
verglichen werden (z.B. am Telefon vorlesen oder von Visitenkarte ablesen).

  ssh-keygen           -l  -f KEYFILE   # -l=list, -f=file (STD: sha256-Hash)
  ssh-keygen -E sha256 -l  -f KEYFILE   # -l=list, -f=file (sha256-Hash)
  ssh-keygen -E md5    -l  -f KEYFILE   # -l=list, -f=file (md5-Hash)
  ssh-keygen           -lv -f KEYFILE   # -v=verbose --> ASCII-Grafik zusÃ¤tzlich!

Der "Fingerprint" wird von SSH bei der Erstellung eines neuen SchlÃ¼ssels bzw.
bei der ersten Verbindungsaufnahme angezeigt (md5- bzw. sha256-Hash):

  2048 MD5:da:0a:08:7a:ee:31:40:20:b6:f6:82:df:f3:94:78:84 tsbirn@exeter (RSA)
  2048 SHA256:LQu4Q/uhWpU/hrjztIJNdLQu+jnHdqKTmQmxsExaP6M tsbirn@exeter (RSA)

Die inzwischen ebenfalls verfÃ¼gbare Darstellung als ASCII-Grafik eignet sich
zur groben optischen Kontrolle, ob man mit dem richtigen Rechner verbunden ist:

  +--[ RSA 2048]----+
  |o.               |
  |+ .              |
  | +               |
  |+ .  .           |
  |+. .E . S        |
  |.+.o o +         |
  |. * = = .        |
```

```
  | o o * .         |
  | .o   o          |
  +-----------------+
```

5.3) Kommando per SSH remote ausfÃ¼hren

Ein Kommando an eine SSH-Verbindung Ã¼bergeben und ausfÃ¼hren lassen:

```
  ssh USER@HOST "CMD"          # Kommando "CMD"
  ssh USER@HOST "ls -l"        # Kommando "ls -l"
```

ACHTUNG: Aufpassen bei Metazeichen (resultierenden Befehl ansehen mit "echo"):

Befehl	"*" ausgewertet auf...
ssh USER@HOST ls -l *	...lokalem Rechner
ssh USER@HOST ls -l *	...Remote-Rechner
ssh USER@HOST ls -l "*"	...Remote-Rechner
ssh USER@HOST ls -l '*'	...Remote-Rechner
ssh USER@HOST "ls -l *"	...Remote-Rechner
ssh USER@HOST 'ls -l *'	...Remote-Rechner

5.4) Dateien per SSH zwischen Rechnern kopieren
---

* ACHTUNG: ":" nicht vergessen, sonst erzeugt man eine LOKALE Kopie!
* Standard-Benutzer ist der aktuell angemeldete Benutzer
* Standard-Verzeichnisse:
  + Lokal: Aktuelles Verz. "." des aktuell angemeldeten Benutzers
  + Remote: Heimat-Verz. des Anmeldebenutzers "~USER"
* Passwort wird verlangt, wenn kein SchlÃ¼sselaustausch stattgefunden hat
* Die Struktur des Filesystems auf dem Remote-Rechner muss bekannt sein,
  damit man dort mit korrekten Pfaden zugreifen kann
* Zugriffsrechte auf Remote-Rechner mÃ¼ssen Zugriff gemÃ¤Ã-^_ Anmeldung erlauben

```
  scp FILE HOST:         # Remote=Ziel: Heimat-Verz. des aktuellen Users
  scp FILE HOST:/tmp     # Remote=Ziel: Verz. "/tmp"
  scp FILE USER@HOST:    # Remote=Ziel: Heimat-Verz. von USER ("-l" unmÃ¶gl!)
  scp HOST:FILE .        # Remote=Quelle: "FILE" im Heimat-Verz. (Ziel: akt. Verz.)
```

Man kann sogar zwischen zwei Remote-Rechnern kopieren:

```
  scp USER1@HOST1:FILE USER2@HOST2:FILE2   # 2x SchlÃ¼sseltausch bzw. anmelden!
```

Einige wichtige Optionen von "scp":

Option	Bedeutung
-C	Komprimierung der Ã¼bertragenen Daten [Compress]
-l LIMIT	Max. Bandbreite in Kbit/s [limit]
-o OPTION=VALUE ...	SSH-Option setzen (mehrfach erlaubt)
-p	Zugriffsrechte + -zeiten erhalten [preserve]
-P NR	Port NR statt 22 (GROSS!) [Port]
-r	Rekursiv (alle Verz. + ihre Inhalte)
-q	Weniger Meldungen (Fortschrittanzeige) [quiet]
-v	Ablaufmeldungen [verbose]
-vv	Mehr Ablaufmeldungen
-vvv	Noch mehr Ablaufmeldungen

5.5) X Window-Protokoll tunneln

X Window-Protokoll vom Server zum Client durchtunneln
(Variable "DISPLAY" anschlieÃM-^_end gesetzt):

```
  ssh -X 172.23.10.56      # Erlauben  (-X=enable)
  ssh -x 172.23.10.56      # Verbieten (-x=disable)
  ssh -Y 172.23.10.56      # Erlauben  (-Y=enable)
```

Um X Window-Anwendungen remote auf dem Remote-Server zu starten und dann lokal
auf dem Client anzuzeigen, mÃ¼ssen verschiedene Bedingungen erfÃ¼llt sein:

  * Remote-Server muss das unterstÃ¼tzen
    (Eintrag "X11Forwarding" in "/etc/ssh/sshd_config", Std. meist "no").

  * Lokaler Client muss das unterstÃ¼tzen (Var. DISPLAY muss gesetzt sein,

```
    Eintrag "ForwardX11" bzw. "ForwardX11Trusted" in "/etc/ssh/ssh_config"
    bzw. "~/.ssh/config", Std. meist "no"; oder Schalter "-X" (enable)
    bzw. "-Y" (enable trusted) beim ssh-Aufruf mitgeben).

  * Evtl. "xhost +HOST" auf lokalem Rechner (macht eigentlich "xauth")


5.6) SchlÃ¼sselbasierte Client-Authentifizierung einrichten (kein Passwort)
--------------------------------------------------------------------------
Bisher: Bei der Verbindungsaufnahme authentifiziert sich der Server beim
Client und der Client muss zur Authentifizierung sein Passwort eingeben
(diese Kommunikation wird bereits verschlÃ¼sselt).

Neu: Die Authentifizierung des Clients soll OHNE Passworteingabe erfolgen.
Dazu muss fÃ¼r den anzumeldenden Benutzer ein SchlÃ¼ssel generiert werden,
dessen Ã¶ffentlicher Teil auf dem Remote-Rechner (Server) in die Datei
"~/.ssh/authorized_keys" einzutragen ist (in seinem Heimatverz.).

Ein Benutzer hat zunÃ¤chst kein Verz. "~/.ssh/" in seinem Heimat-Verz. Dieses
wird erzeugt, wenn fÃ¼r ihn ein SchlÃ¼sselpaar generiert wird (ohne "Passphrase"
= leere Passphrase, -t=type, "*.pub" = Ã¶ffentlicher SchlÃ¼ssel):

  +-----------------------+-------------------------------------------+
  | Aufruf                | Erzeugte SchlÃ¼sseldateien                 |
  +-----------------------+-------------------------------------------+
  | ssh-keygen -t rsa     | ~/.ssh/id_rsa     + ~/.ssh/id_rsa.pub     |
  | ssh-keygen -t dsa     | ~/.ssh/id_dsa     + ~/.ssh/id_dsa.pub     |
  | ssh-keygen -t ecdsa   | ~/.ssh/id_ecdsa   + ~/.ssh/id_ecdsa.pub   |
  | ssh-keygen -t ed25519 | ~/.ssh/id_ed25519 + ~/.ssh/id_ed25519.pub |
  +-----------------------+-------------------------------------------+
  | ssh-keygen -t rsa1    | ~/.ssh/identity   + ~/.ssh/identity.pub   |
  | ssh-keygen            | (analog)                                  |
  +-----------------------+-------------------------------------------+

Man erhÃ¤lt einen "Fingerprint" (16-stellige Hexadezimalzahl) des generierten
SchlÃ¼ssels ausgegeben, den man z.B. auf seiner Web-Seite verÃ¶ffentlicht oder in
seine Visitenkarte eintrÃ¤gt. Andere kÃ¶nnen damit verifizierten, ob der
Verbindungspartner wirklich der richtige ist.

  2048 MD5:da:0a:08:7a:ee:31:40:20:b6:f6:82:df:f3:94:78:84 tsbirn@exeter (RSA)
  2048 SHA256:LQu4Q/uhWpU/hrjztIJNdLQu+jnHdqKTmQmxsExaP6M tsbirn@exeter (RSA)

Der Fingerprint kann auch optisch durch eine ASCII-Grafik dargestellt werden:

  +--[ RSA 2048]----+
  |o.               |
  |+ .              |
  | +               |
  |+ .  .           |
  |+. .E . S        |
  |.+.o o +         |
  |. * = = .        |
  | o o * .         |
  | .o   o          |
  +-----------------+

NachtrÃ¤glich Fingerprint bzw. ASCII-Grafik zu einem SchlÃ¼ssel anzeigen:

  ssh-keygen -l           -f KEYFILE   # -l=list, -f=file (STD: sha256-Hash)
  ssh-keygen -l -E md5    -f KEYFILE   # md5-Hash
  ssh-keygen -l -E sha256 -f KEYFILE   # sha256-Hash
  ssh-keygen -lv          -f KEYFILE   # -v=verbose --> ASCII-Grafik zusÃ¤tzlich!

Der Weg, auf dem der Ã¶ffentliche SchlÃ¼sselteil anschlieÃM-^_end zum jeweiligen
Remote-Rechner transportiert wird, darf nicht manipulierbar sein (z.B. per
Hand, Telefon, Diskette, USB-Stick, NICHT per eMail!). Eine weitere
(unsichere) MÃ¶glichkeiten wÃ¤re der Transport per Maus in einer SSH-Anmeldung.

  cat ~/.ssh/id_rsa.pub                  # ÃM-^Vffentlichen Client SchlÃ¼ssel anzeig.
  ssh USER@HOST                          # Remote-Anmeldung per Passwort
  mkdir ~/.ssh                           # Unterverz. fÃ¼r SSH-Keys anlegen
  vi ~/.ssh/authorized_keys              # Datei mit Client-SchlÃ¼sseln anzeigen
  ... angezeigten SchlÃ¼ssel per Maus als oberste Zeile einfÃ¼gen ...
  :wq                                    # Datei speichern (w=write, q=quit)
  exit                                   # Remote-Abmeldung
  ssh USER@HOST                          # Anmeldung per SchlÃ¼sselaustausch

Oder mit "scp":

  scp ~/.ssh/id_rsa.pub USER@HOST:/tmp   # Anmeldung per Passwort
  ssh USER@HOST                          # Anmeldung per Passwort
  mkdir ~/.ssh                           # Unterverz. fÃ¼r SSH-Keys anlegen
```

```
  cat /tmp/id_rsa.pub >> ~/.ssh/authorized_keys   # ÃM-^Vffentl. Client-Key anhÃ¤ngen
  exit                                 # Remote-Abmeldung
  ssh USER@HOST                        # Anmeldung per SchlÃ¼sselaustausch

Oder alles auf einen Schlag:

  ssh USER@HOST "mkdir ~/.ssh;           # Anmeldung per Passwort
                 cat >> ~/.ssh/authorized_keys" < ~/.ssh/id_rsa.pub

Oder per Skript:

  ssh-copy-id            USER@HOST   # ~/.ssh/id_rsa.pub (Std)
  ssh-copy-id -i         USER@HOST   # ~/.ssh/identity.pub
  ssh-copy-id -i KEYFILE USER@HOST   # FILE

ACHTUNG: Kein Anfangs/EndstÃ¼ck des Ã¶ffentlichen SchlÃ¼ssels weglassen,
         muss aus EINER Zeile bestehen (keine ZeilenumbrÃ¼che).

Nun sollte ein Anmeldung per "ssh" OHNE Passworteingabe mÃ¶glich sein!


5.6.1) Grafik zum Ablauf einer Client-Authentifizierung
-------------------------------------------------------
Letzten Endes hat folgender Austausch des Ã¶ffentlichen SchlÃ¼sselteils der
jeweiligen Gegenstelle "Ã¼ber Kreuz" stattgefunden:

              USER1                                      USER2
                                ANMELDUNG
          ssh USER2@HOST ============================> sshd
           +-----------+       SSH-Protokoll      +---------------+
           |  Client   |                          | Server (HOST) |
           |  (local)  |                          |   (remote)    |
           +-----+-----+                          +-------+-------+
                 |                                        |
  (MANUELL erzeugt: ssh-keygen -t rsa)    (bei Installation AUTOMAT. erzeugt)
  /home/USER1/.ssh/id_rsa.pub             /etc/ssh/ssh_host_rsa_key.pub
                 |                                        |
                 |  (AUTOMATISCH bei 1. SSH-Verb. + "yes") |
           +-----|----------------------------------------+
           |     |
           |     +-----------------------------------------+
           |                 (MANUELL pro Benutzer)          |
           v                                                 v
  /home/USER1/.ssh/known_hosts # oder        /home/USER2/.ssh/authorized_keys
  /etc/ssh/ssh_known_hosts                  (/etc/ssh/authorized_keys NICHT!)

TIPP: Sammeln der bei allen Benutzern (inkl. "root") vorh. "~/.ssh/known_hosts"
in zentraler Datei "ssh_known_hosts" (doppelte EintrÃ¤ge nur 1x):

  cat /home/*/.ssh/known_hosts > /tmp/uuu          # Umlenken + sudo nicht glz.
  sudo cp /root/known_hosts /tmp/rrr               #
  cat /tmp/uuu /tmp/rrr | sort | uniq > /tmp/hhh   # Umlenken + sudo nicht glz.
  sudo cp /tmp/hhh /etc/ssh/ssh_known_hosts        #

HINWEIS: Ob das eine gute Idee ist, sei dahingestellt! Die zentralen SchlÃ¼ssel
in "/etc/ssh/ssh_known_hosts" sollten VERIFIZIERT sein. Benutzerspezifische
SchlÃ¼ssel in "~/.ssh/known_hosts" kÃ¶nnten durch Eingabe von "yes" bei der
ERSTEN Verbindungsaufnahme akzeptiert sein und sind dann NICHT verifiziert.

NACH dem Austausch der Ã¶ffentlichen SchlÃ¼ssel wird hÃ¤ufig in
"/etc/ssh/sshd_config" die Authentifizierung via Passwort deaktiviert
(Std = vordefinierte Defaultwerte, Sec = sichere Werte):

  +------------------------+---+---+-------------------------------------------+
  | Konfigurationsvariable |Std|Sec| Bedeutung                                 |
  +------------------------+---+---+-------------------------------------------+
  | RSAAuthentication      |yes|yes| Reine RSA-Authentifiz. erlaubt (nur P1)   |
  | PubkeyAuthentication   |yes|yes| Public-Key Authentifiz. erlaubt (nur P2)  |
  | PasswordAuthentication |yes|no | SSH fragt selber nach dem Passwort        |
  | UsePAM                 |no |no | Sonst wird Passwort von PAM verlangt      |
  +------------------------+---+---+-------------------------------------------+

Sicherheitstechnisch wichtig Einstellungen in "/etc/ssh/sshd_config":

  +------------------------+---+---+-------------------------------------------+
  | Konfigurationsvariable |Std|Sec| Bedeutung                                 |
  +------------------------+---+---+-------------------------------------------+
  | PermitRootLogin        |yes| no| Direkte root-SSH-Anmeldung nicht mÃ¶glich  |
  | PermitEmptyPasswords   | no| no| Leere Passworte verboten                  |
  | IgnoreRhosts           |yes|yes| "~/.rhosts" "~/.shosts" nicht benutzen    |
  | IgnoreUserKnownHosts   | no|yes| Userspez. "~/.ssh/known_hosts" verboten,  |
  |                        |   |   | Systemspez. "/etc/ssh/known_hosts" nÃ¶tig  |
```

```
  +------------------------+---+---+-------------------------------------------+

Weitere sinnvolle Einstellungen in "/etc/ssh/sshd_config":

  +------------------------+---+---+-------------------------------------------+
  | Konfigurationsvariable |Std|Sec| Bedeutung                                 |
  +------------------------+---+---+-------------------------------------------+
  | StrictModes            |yes|yes| Zugriffsrechte auf Dateien prÃ¼fen         |
  | X11Forwarding          | no| no| X11-Protokoll tunneln                     |
  | X11DisplayOffset       | 10| 10| X11-Display Startnummer                   |
  | TCPKeepAlive           |yes|yes| TCP-Keepalive Meldungen schicken          |
  +------------------------+---+---+-------------------------------------------+
  | Port                   |122| 22| Port 22 --> 122 (gegen Bot-Attacken)      |
  | Protocol               |2,1|  2| Protokoll 1 ist unsicher                  |
  | ChallengeResponse-     |yes| no| CR-Authentifizierung erlaubt (PAM)        |
  |   Authentication       |   |   |                                           |
  +------------------------+---+---+-------------------------------------------+


5.7) Privaten SchlÃ¼ssel mit Passphrase sichern
----------------------------------------------
Der private SchlÃ¼ssel sollte durch eine "Passphrase" (das ist ein lÃ¤ngerer Text
Ã¤hnlich einem Passwort) gesichert werden, mit der er "Ã¼berschlÃ¼sselt" wird,
damit er nicht im Klartext gespeichert ist. Diebstahl oder Kompromittierung des
Rechners fÃ¼hrt dann nicht zur Kompromittierung des privaten SchlÃ¼ssels.
Beim Remote-Login per SSH wird dann nach der Passphrase des SchlÃ¼ssels gefragt
(damit dieser entschlÃ¼sselt werden kann) statt nach dem Passwort.

  $ ssh-keygen -p -f ~/.ssh/id_dsa     # -p=passphrase, -f=file
  $ ssh-keygen -p                      # Automatisch Datei ~/.ssh/id_rsa
  Enter file in which the key is (/home/tsbirn/.ssh/id_rsa):
  Enter old passphrase: XXXXXXXXX
  Key has comment '/home/tsbirn/.ssh/id_rsa'
  Enter new passphrase (empty for no passphrase): YYYYYYYYY
  Enter same passphrase again: YYYYYYYYY
  Your identification has been saved with the new passphrase.

Die "Passphrase" ist jedesmal einzugeben, wenn ein Zugriff auf den privaten
SchlÃ¼ssel notwendig ist (fÃ¼r automatischen Verbindungs-Aufbau ist das unsinnig,
man kann aber auch mehr als ein SchlÃ¼sselpaar erzeugen, um dies zu umgehen).

Soll die "Passphrase" wieder entfernt werden, den privaten SchlÃ¼ssel analog
mit einer LEEREN Passphrase Ã¼berschlÃ¼sseln.


5.8) SSH-Agent zum SchlÃ¼ssel verwalten
--------------------------------------
Alternativ kann die Passphrase Ã¼ber einen SSH-Agenten zur VerfÃ¼gung gestellt
werden, so dass sie nur 1x eingegeben werden muss, anschlieÃM-^_end im Speicher
steht und bei jedem SSH-Kommando automatisch zur VerfÃ¼gung gestellt wird:

  ssh-agent   # Linux
  Pageant     # Windows (Putty Agent)

Er dient als "Wrapper" fÃ¼r beliebige andere Prozesse (z.B. die Shell oder den
X-Server) und hÃ¤lt fÃ¼r sie entschlÃ¼sselte private SchlÃ¼ssel im Speicher vor.

  ssh-agent              # SSH-Agenten starten --> ergibt Shell-Befehle
  eval `ssh-agent -c`    # -c=C-Shell Befehle erzeugen (C-Shell Syntax)
  eval `ssh-agent -s`    # -s=Bourne-Shell Befehle erzeugen (Bourne-Syntax)
  eval $(ssh-agent -s)   # -s=Bourne-Shell Befehle erzeugen (Bash-Syntax)

Der Agent produziert als Ausgabe Shell-Befehle, die zwei Umgebungs-Variablen
SSH_AUTH_SOCK und SSH_AGENT_PID setzen, mit deren Hilfe andere Prozesse den
Agenten finden und mit ihm kommunizieren kÃ¶nnen:

  SSH_AUTH_SOCK=/tmp/ssh-qsiis28019/agent.28019; export SSH_AUTH_SOCK;
  SSH_AGENT_PID=28020; export SSH_AGENT_PID;
  echo Agent pid 28020;

Diese Ausgabe auf Stdout muss abgefangen und als Befehl ausgefÃ¼hrt werden,
daher die Kommando-Substitution $(...) bzw. `...` und Auswertung per "eval".

Nach dem Start des SSH-Agenten werden SSH-SchlÃ¼ssel an ihn Ã¼bergeben durch:

  Pageant-Icon          # Windows, Passphrase 1x eingeben (in graf. Dialog)
  ssh-add               # Linux, Passphrase 1x eingeben (auf Kommandozeile)
  ssh-add < /dev/null   # Linux, Passphrase 1x eingeben (in graf. Dialog)

Dabei wird wenn notwendig 1x die Passphrase zum EntschlÃ¼sseln abgerufen.

  ssh-add                     # Alle verfÃ¼gbaren SchlÃ¼ssel merken (max. 3)
```

```
  ssh-add ~/.ssh/id_rsa       # Einen RSA-SchlÃ¼ssel merken
  ssh-add ~/.ssh/id_dsa       # Einen DSA-SchlÃ¼ssel merken
  ssh-add ~/.ssh/id_ecdsa     # Einen ECDSA-SchlÃ¼ssel merken
  ssh-add ~/.ssh/id_ed25519   # Einen ED25519-SchlÃ¼ssel merken
  ssh-add ~/.ssh/identity     # Einen RSA-SchlÃ¼ssel merken

TIPP: Folgenden Alias "sshadd" definieren, der die beiden notwendigen
Befehle kombiniert und ihre etwas komplexe Syntax verpackt:

  alias sshadd="eval $(ssh-agent -s); ssh-add"               # oder
  alias sshadd="eval $(ssh-agent -s); ssh-add < /dev/null"   #


5.9) Bei SSH-Anmeldung Kommando ausfÃ¼hren/Optionen setzen
---------------------------------------------------------

In Datei "~/.ssh/authorized_keys" kÃ¶nnen am Zeilenanfang jedes SchlÃ¼ssels VOR
dem SchlÃ¼ssel-Typ "ssh-rsa/ssh-dsa/ssh-ecdsa/ssh-ed25519/ssh" folgende Optionen
angegeben werden, die dann bei einer Anmeldung Ã¼ber diesen SchlÃ¼ssel gelten:

  +--------------------------+-------------------------------------------------+
  | Option                   | Bedeutung                                       |
  +--------------------------+-------------------------------------------------+
  | command="CMD...;..."     | Kommando ausfÃ¼hren (mehrere durch ";" trennen) |
  | environment="VAR=VALUE"  | Umgebungsvariable setzen (durch "," trennen)    |
  | from="HOSTLIST"          | Auf bestimmte Remote-Rechner beschrÃ¤nken (",")  |
  | no-agent-forwarding      | SSH-Agenten NICHT weitergeben                   |
  | no-port-forwarding       | Ports NICHT weiterleiten (-L/-R)                |
  | no-pty                   | Kein Terminal zuweisen                          |
  | no-user-rc               | "~/.ssh/rc" NICHT ausfÃ¼hren                     |
  | no-X11-forwarding        | X11 NICHT weiterleiten                          |
  | permitopen="HOST:PORT"   | Lokale Weiterleitung beschrÃ¤nken                |
  | tunnel="N"               | Tunnel N erzwingen (sonst nÃ¤chstes freies)      |
  +--------------------------+-------------------------------------------------+

ACHTUNG: Alle Optionen in eine Zeile schreiben, Werte mit Leerzeichen in "..."
oder '...' einschlieÃM-^_en, mehrere Angaben durch "," trennen (keine Leerzeichen
zur Trennung verwenden), mehrere Kommandos durch ";" trennen".

Beispiele:

  command="/bin/ls > /tmp/ls.txt"        ssh-rsa   KEY...   # Nur Kmdo ausfÃ¼hren + sofort wieder ab
melden
  command="/usr/bin/ssh tom@192.168.1.1" ssh-dsa   KEY...   # Auf weiteren Rechner springen
  from="192.168.1.1"                     ssh-ecdsa KEY...   # Nur von diesem Rechner Anmeldung erlau
bt
  from="192.168.1.1",command="/usr/bin/ssh tom@192.168.1.1" ssh KEY...   # Auf weiteren Rechner spri
ngen

Der Eintrag "command=CMD..." sorgt dafÃ¼r, dass kein echter Login mehr erfolgt,
sondern das Kommando "CMD..." ausgefÃ¼hrt und danach die SSH-Verbindung wieder
beendet wird (sofern eine schlÃ¼sselbasierte Anmeldung erfolgreich stattfindet).
Bei einer passwortbasierten Anmeldung wird kein Kommando ausgefÃ¼hrt. Sinn:

  * Kommando/Skript per reiner Anmeldung ausfÃ¼hren (freischalten)
  * SSH-Verbindung nach auÃM-^_en Ã¼ber Proxy-Server (keine End-zu-End-Verbindung)
  * SSH-Datenfluss Ã¼berwachen

HINWEIS: Die Dokumentation dazu findet man in "man sshd" (Daemon!)


6) Port-Forwarding (Tunneling von Protokollen)
----------------------------------------------
Durchschleusen von anderen Protokollen durch einen SSH-Tunnel. Insbesondere
KLARTEXT-Protokolle (z.B. TELNET, FTP, POP3, IMAP, SMTP, SNMP, ...) kÃ¶nnen auf
diese Weise nachtrÃ¤glich per verschlÃ¼sselten "Wrapper" abgesichert werden.

Dazu muss folgende "sshd"-Einstellung in "/etc/ssh/sshd_config" gelten:

  AllowTcpForwarding  yes

Es gibt zwei "Richtungen" des "Tunneling" gesteuert durch das SSH-Kommando auf
dem CLIENT, je nachdem ob anschlieÃM-^_end eine Verbindung durch den Tunnel
ausgehend vom lokalen Rechner CLIENT (-L/-D) oder ausgehend vom remote Rechner
SERVER (-R) aufgebaut werden soll:

        local                    remote
        CLIENT                   SERVER
     +----------+              +----------+
     |          | <==========> |          |
     +----------+              +----------+
         1111                      99         # Ports
```

```
  1. Verbindung CLIENT --> SERVER Ã¶ffnen,
     lokal am CLIENT auf Port 1111 lauschen,
     Daten weitergeben an SERVER auf Port 99
     (d.h. CLIENT ist Client, SERVER ist Server).
     ERGEBNIS: Tunnel von CLIENT:1111 --> SERVER:99
     ACHTUNG: "localhost" oder "127.0.0.1" statt CLIENT-IP nach -L!

       CLIENT: ssh -L 1111:localhost:99 SERVER   # Tunnel aufbauen (-L=local)
       CLIENT: telnet localhost 1111             # Tunnel CLIENT --> SERVER nutzen

  2. Verbindung SERVER --> CLIENT Ã¶ffnen,
     remote am SERVER auf Port 99 lauschen,
     Daten weitergeben an CLIENT auf Port 1111
     (d.h. SERVER ist Client, CLIENT ist Server!).
     ERGEBNIS: Tunnel von SERVER:99 --> CLIENT:1111
     ACHTUNG: CLIENT-IP statt "localhost" nach -R!

       CLIENT: ssh -R 99:CLIENT:1111 SERVER   # Tunnel aufbauen (-R=remote)
       SERVER: telnet localhost 99            # Tunnel SERVER --> CLIENT nutzen

Beispiele:

A) Lokal von Port 8080 auf Port 80 eines remote SERVER tunneln
   (keiner kann Kommunikation mitlesen, -L=lokal, lange laufendes Kommando
   starten, das keine Rechenzeit verbraucht und es in den Hintergrund schieben):

     ssh -L 8080:localhost:80 USER@SERVER "sleep 100000" &   # Nur wenn ohne Pw. mÃ¶gl.
     ssh -L 8080:localhost:80 USER@SERVER "sleep 100000"     # Auch mit Passwortabfrage
     Strg-Z
     bg

   Im Browser auf CLIENT den lokalen Rechner auf Port 8080 ansprechen,
   die HTTP-Daten werden verschlÃ¼sselt an/vom SERVER Port 80 Ã¼bertragen:

     http://localhost:8080   # <-> SERVER:80

B) POP3-Protokoll lokal von Port 5000 auf remote SERVER Port 110 tunneln
   (keiner kann Kommunikation mitlesen):

     ssh -L 5000:localhost:110 SERVER   # Tunnel aufbauen
     fetchmail                          # GÃ¼ltige ".fetchmailrc" muss existieren
                                        # und auf localhost:5000 zugreifen

C) VerschlÃ¼sselte Telnet-Verbindung lokal von Port 5000 auf remote SERVER
   Port 23 (telnet) aufbauen (vorher telnet-Daemon/Server auf SERVER installieren
   und aktivieren sowie "xinetd" rekonfigurieren und neu starten):

     ssh -L 5000:localhost:23 SERVER "sleep 100000"   # Tunnel aufbauen
     telnet localhost 5000                            # Telnet-Verb. aufbauen

D) Von Port 3306 lokal auf Port 3306 remote tunneln (MySQL-Server):

     ssh -L 3306:localhost:3306 SERVER          # Tunnel aufbauen
     mysql -uUSER -p --protocol tcp localhost   # TCP-Verbindung erzwingen!

E) Port 80 auf Zielrechner SERVER per Zwischenrechner ZWHOST vom CLIENT aus
   auf Port 10080 verbinden (2 Tunnels!) und somit das HTTP-Protokoll (Port 80)
   vom SERVER zum Port 10080 auf dem CLIENT Ã¼ber eine Zwischenstation ZWHOST
   (z.B. Firewall) tunneln:

        CLIENT                   ZWHOST                    SERVER
     +----------+             +----------+              +----------+
     |          | <==========> |          |  <========== |          |
     +----------+             +----------+              +----------+
        10080                    (3080)                     80

   Variante 1):
     ssh -l root -L 10080:SERVER:80 ZWHOST          # Auf ZWHOST ausgefÃ¼hrt

   Variante 2):
     ssh -l root -L 10080:localhost:3080 ZWHOST \   # Auf CLIENT ausgefÃ¼hrt
            "ssh -L  3080:localhost:80   SERVER"    # Auf ZWHOST ausgefÃ¼hrt

   Ablauf: Anmelden am SSH-Port von Rechner ZWHOST
           Tunneln von Port 10080 am eigene Rechner LOCALHOST
           zu Port 3080 auf mittlerem Rechner ZWHOST (-L = "localhost" = dort)
           Anmelden am SSH-Port von Rechner SERVER
           Tunneln von Port 3080 am Rechner ZWHOST
           zu Port 80 auf Endrechner SERVER (-L = "localhost" = dort)

   HINWEIS: Damit obiger Befehl nicht stÃ¤ndig einzutippen ist, sollte er per
```

```
   Alias oder Funktion in einer der persÃ¶nlichen Shell-Konfigurations-Dateien
   oder per Shell-Skript in persÃ¶nlichen BinÃ¤r-Verz. realisiert werden:

     # Alias
     alias srvhttp='ssh -l root -L 10080:localhost:3080 ZWHOST \
                           "ssh -L  3080:localhost:80   SERVER"'

     # Funktion
     srvhttp()
     {
         ssh -l root -L 10080:localhost:3080 ZWHOST \
                "ssh -L  3080:localhost:80   SERVER"
     }

     # Shell-Skript "srvhttp.sh" (ausfÃ¼hrbar machen: chmod a+x srvhttp.sh):
     +-------------------------------------------+
     |#!/bin/sh                                  |
     |ssh -l root -L 10080:localhost:3080 ZWHOST \|
     |       "ssh -L  3080:localhost:80   SERVER" |
     +-------------------------------------------+

F) Konfiguration einer "Putty-Session" am Windows-Client 192.168.0.200, die
   Ã¼ber den Zwischenrechner 192.168.0.201 auf den Web-Server 192.168.0.250
   Web-Zugriff durchfÃ¼hren soll:

     192.168.0.200            192.168.0.201              192.168.0.250
     User: Admin              User: schulung             User: kurs1
     +-----------+             +-----------+              +-----------+
     |   WIN     | <==========> |  ZWHOST   |  <==========> |  SERVER   |
     +-----------+             +-----------+              +-----------+
          80                       3080                       80

   In "Putty-Session" vorzunehmende Einstellungen:
     Session --> Host Name (or IP address):              # Passwort/Passphrase
       schulung@192.168.0.201
     Connection --> SSH --> Tunnels --> Forwarded Ports: # Source port: 80
       L80 localhost:3080                                # Destination: localhost:3080
     Connection --> SSH --> Remote Command:
       ssh -l kurs1 -L 3080:localhost:80 192.168.0.250   # Passwort fÃ¼r "kurs1"!
       ssh -L 3080:localhost:80 kurs1@192.168.0.250      # (analog)

   Achtung: Wahrscheinlich ist 2x Passwort/Passphrase einzugeben:
   1) Erst bei Verbindung Client --> Zwischenrechner fÃ¼r User "schulung".
   2) Dann bei Verbindung Zwischenrechner --> Server fÃ¼r User "kurs1".


7) Ã¬bersicht
------------

7.1) Kommandos
--------------
ssh, slogin                     Verbindung zu fremdem Rechner aufnehmen
                                (sicherer Ersatz von "telnet", "rsh", "rlogin")
scp                             Dateien von/zu fremden Rechner transferieren
                                (sicherer Ersatz von "rcp")
sftp                            Dateien von/zu fremden Rechner transferieren
                                (sicherer Ersatz von "ftp")
ssh-keygen                      SchlÃ¼sselpaar erzeugen
ssh-keyscan                     Ã¬ffentlichen SchlÃ¼ssel von Rechnern holen
ssh-add                         Privaten SchlÃ¼ssel beim "ssh-agent" registrieren
ssh-agent                       Privaten SchlÃ¼ssel verwalten
                                (automatische Beantwortung von "challenges")

7.2) Server-Konfigurations-Dateien
----------------------------------
/etc/ssh/sshd_config            Zentrale Server-Konfiguration
                                (es gibt keine benutzerspezifische!)

/etc/ssh/ssh_host_rsa_key         Server Identity (Private Key, Protocol V2 = RSA)
/etc/ssh/ssh_host_rsa_key.pub     Server Identity (Public  Key, Protocol V2 = RSA)
/etc/ssh/ssh_host_dsa_key         Server Identity (Private Key, Protocol V2 = DSA)
/etc/ssh/ssh_host_dsa_key.pub     Server Identity (Public  Key, Protocol V2 = DSA)
/etc/ssh/ssh_host_ecdsa_key       Server Identity (Private Key, Protocol V2 = ECDSA)
/etc/ssh/ssh_host_ecdsa_key.pub   Server Identity (Public  Key, Protocol V2 = ECDSA)
/etc/ssh/ssh_host_ed25519         Server Identity (Private Key, Protocol V2 = ED25519)
/etc/ssh/ssh_host_ed25519_key.pub Server Identity (Public  Key, Protocol V2 = ED25519)
/etc/ssh/ssh_host_key             Server Identity (Private Key, Protocol V1 = RSA)
/etc/ssh/ssh_host_key.pub         Server Identity (Public  Key, Protocol V1 = RSA)

/etc/ssh/ssh_known_hosts        Public Keys der bekannten Remote-Server (fest)
```

```
7.3) Client-Konfigurations-Dateien
----------------------------------
/etc/ssh/ssh_config             Zentrale Client-Konfiguration
~/.ssh/config                   Benutzerspezifische Client-Konfiguration

~/.ssh/id_rsa                   User Identity (Private Key, Protocol 2 = RSA)
~/.ssh/id_rsa.pub               User Identity (Public  Key, Protocol 2 = RSA)
~/.ssh/id_dsa                   User Identity (Private Key, Protocol 2 = DSA)
~/.ssh/id_dsa.pub               User Identity (Public  Key, Protocol 2 = DSA)
~/.ssh/id_ecdsa                 User Identity (Private Key, Protocol 2 = ECDSA)
~/.ssh/id_ecdsa.pub             User Identity (Public  Key, Protocol 2 = ECDSA)
~/.ssh/id_ed25519               User Identity (Private Key, Protocol 2 = EC25519)
~/.ssh/id_ed25519.pub           User Identity (Public  Key, Protocol 2 = EC25519)
~/.ssh/identity                 User Identity (Private Key, Protocol 1 = RSA)
~/.ssh/identity.pub             User Identity (Public  Key, Protocol 1 = RSA)

~/.ssh/known_hosts              Public Keys der bekannten Remote-Server (updatebar)
~/.ssh/authorized_keys          Public Keys der bekannten Remote-Benutzer


7.4) Login-Shell Konfigurations-Dateien
---------------------------------------
/etc/ssh/sshrc            Zentrale Kommandos fÃ¼r Login-Shells
~/.ssh/rc                 Benutzerabh. Kommandos fÃ¼r Login-Shell
~/.ssh/environment        Environment-Variablen fÃ¼r Login-Shell


8) Tipps
--------
* Ã¶ffentlichen SchlÃ¼ssel per Kommandoaufruf vom Remote-Rechner holen:

    ssh-keyscan -t rsa HOST        # V2: RSA-SchlÃ¼ssel von HOST holen
    ssh-keyscan -t dsa HOST        # V2: DSA-SchlÃ¼ssel von HOST holen
    ssh-keyscan -t ecdsa HOST      # V2: ECDSA-SchlÃ¼ssel von HOST holen
    ssh-keyscan -t ed25519 HOST    # V2: ED25519-SchlÃ¼ssel von HOST holen
    ssh-keyscan HOST               # V1: RSA-SchlÃ¼ssel von HOST holen

* Alle Rechner in Datei "ssh_hosts" finden, die einen neuen oder anderen
  Ã¶ffentlichen RSA/DSA/ECDSA/ED25519-SchlÃ¼ssel haben als in "ssh_known_hosts"
  hinterlegt (beide Dateien mÃ¼ssen nicht sortiert sein):

     ssh-keyscan -t rsa1,rsa,dsa,ecdsa,ed25519 -f ssh_hosts |
         sort -u - ssh_known_hosts |
         diff - <(sort ssh_known_hosts)

* Ã¶nderungen an der SSH-Server-Konfigurations-Datei "/etc/ssh/sshd_config"
  wirken sich erst dann aus, wenn der SSH-Daemon neu gestartet wird.

* Ã¶nderungen an der SSH-Client-Konfigurations-Datei "/etc/ssh/ssh_config"
  bzw. "~/.ssh/config" wirken sich automatisch auf DANACH neu hergestellte
  Verbindungen aus, NICHT aber auf bereits aktive Verbindungen.

* Der eigene Benutzername ist Standard bei der SSH-Anmeldung, sofern man keinen
  anderen angibt (per -l USER oder USER@HOST). D.h. man kann sich mit einem
  anderen Benutzernamen auf dem Remote-Rechner anmelden, sofern man dessen
  Passwort weiÃ¶_ oder dessen privaten Key besitzt.

* BesitzverhÃ¤ltnisse + Zugriffsrechte der SSH-Konfigurations-Dateien und
  ihrer Verz.pfade mÃ¼ssen einige Bedingungen erfÃ¼llen, damit SSH eine
  schlÃ¼sselbasierte Anmeldung erlaubt (nur falls "StrictModes yes" in
  "/etc/ssh/sshd_config" gesetzt ist):
  + Benutzer USER der SSH-Anmeldung MUSS Besitzer folgender Verz./Dateien sein:
      "/home/USER"
      "/home/USER/.ssh"
      "/home/USER/.ssh/authorized_keys"
  + FÃ¼r obige Verz./Dateien darf NICHT gesetzt sein:
      "w" fÃ¼r "Besitzer-Gruppe" + "Andere"    -->   drwxr-xr-x   -rwxr--r--
  + FÃ¼r priv. SchlÃ¼sseldateien darf NICHT gesetzt sein:
      "rw" fÃ¼r "Besitzer-Gruppe" + "Andere"   -->   -rw-------

  Typische Fehlermeldungen in "/var/log/auth.log" (bzw. "/var/log/messages")
  auf der Server-Seite bei falschen Zugriffsrechten auf Konfigurations-Dateien:

    Authentication refused: bad ownership or modes for directory /home/USER1
    Authentication refused: bad ownership or modes for directory /home/USER1/.ssh

* Der "ssh-agent" legt die Passphrase bzw. den privaten SchlÃ¼ssel NIE auf Datei
  ab, sondern behÃ¤lt sie im Speicher, bis der Benutzer sich wieder abmeldet.

* Der "ssh-agent" ist nur fÃ¼r Programme aktiv, die im gleichen Terminal wie er
  gestartet werden (die gleiche Shell als Elternprozess haben).
```

```
* StandardmÃ¤Ã-^_ig erfolgt keine X11-Umlenkung mehr. Dazu ist die Option "-X/-Y"
  beim Aufruf von "ssh" anzugeben oder "ForwardX11" und "TrustedForwardX11" in
  "/etc/ssh/ssh_config" bzw. "~/.ssh/config" auf "yes" zu setzen.

* Zum testen von Verbindungsproblemen:
    ssh -v     # Meldungen zum Verbindungsaufbau
    ssh -vv    # Mehr Meldungen zum Verbindungsaufbau
    ssh -vvv   # Noch mehr Meldungen zum Verbindungsaufbau

* Woher stammt der HOST-Key unter "/etc/ssh"?
  Wird beim Installieren eines Rechners automatisch eingerichtet.
  --> Kopien eines Rechners als virtuelle Maschine haben gleichen HOST-Key!

* Unbedingt Passphrase != Passwort(e) wÃ¤hlen
  (hÃ¶here Sicherheit; weniger Verwirrung: was mache ich eigentlich?).

* Privater SchlÃ¼ssel:
  + DIREKT auf Rechner erzeugen, wo er eingesetzt werden soll
  + NIE weitergeben/kopieren/...
  + Ã-^\BERSCHLÃ-^\SSELN mit Passphrase falls Rechner allgemein zugÃ¤nglich ist
  + SICHER verwahren: Bei Verlust sind alle mit dem Ã¶ffentlichen SchlÃ¼ssel
    verschlÃ¼sselten Daten verloren (z.B. ausgedruckt in Safe legen)

* Typischer FEHLER: Neues SchlÃ¼sselpaar erzeugen, dabei wird altes Ã¼berschrieben
  --> SchlÃ¼sselpaar ausdrucken und sicher verwahren

* In Datei "authorized_keys" gilt:
  + Jedes einzelne Zeichen wichtig
  + Anzahl der Leerzeichen wichtig
  + Vorne EinschrÃ¤nkungen fÃ¼r die Verbindung eintragbar (bis 1. Leerzeichen)
  + Text hinten (USER@HOST) irrelevant (bel. Kommentar nach letztem Leerz.)
  + Pro Zeile ein vollstÃ¤ndiger (langer) Eintrag
  + Leerzeilen werden ignoriert


9) Links
--------
* http://www.ssh.fi                                                  SSH (original)
* http://www.ssh.com                                                 SSH (original)
* http://www.openssh.org                                             OpenSSH
* http://www.openssh.com                                             OpenSSH
* ftp://ftp.de.openbsd.org/pub/unix/OpenBSD/OpenSSH/portable        OpenSSH
* http://www.chiark.greenend.org.uk/~sgtatham/putty                  Putty (Windows SSH)
* ftp://ftp.freesoftware.com/pub/infozip/zlib                        Zlib
* http://www.openbsd.org                                             OpenBSD
* http://www.openssl.org                                             OpenSSL
* http://www.winscp.net                                              WinSCP
* http://www.mobatek.net                                             MobaXterm (X-Terminal fÃ¼r Windows)
* http://www.bananahut.net                                           MobaXterm Plugins
* http://www.filezilla.de                                            FileZilla
* http://kitty.9bis.com                                              KiTTY (Putty-Ersatz)
* https://launchpad.net                                              Ã-^Vffentlicher Keyserver


================================================================================
```