

corewire

Linux

SSH Grundlagen

# Folien-Hinweis

- Space, Page down: Nächste Folie
- Page up: Vorherige Folie
- ESC, o: Übersicht

[Zur Kapitelübersicht](#)

# Was ist SSH?

# SSH - Secure Shell

## Sichere Fernzugriff und Dateienübertragung

- **Verschlüsselte Kommunikation** über unsichere Netzwerke
- **Ersatz für Telnet, rlogin, ftp** und ähnliche unverschlüsselte Protokolle
- **Standard-Port 22 (TCP)**
- **Client-Server-Architektur**

# SSH - Secure Shell

## Hauptfunktionen:

- **Remote Shell Access** - Fernzugriff auf die Kommandozeile
- **Dateiübertragung** - scp, sftp
- **Port-Weiterleitung** - Tunnel für andere Services
- **X11-Weiterleitung** - Grafische Anwendungen über SSH

# SSH-Client

- Auf dem lokalen System installiert
- Verbindet sich zu SSH-Servern
- Verschiedene Implementierungen: OpenSSH, PuTTY, etc.

```
# SSH-Client verwenden  
ssh user@hostname
```

# SSH-Server

- Läuft auf dem Zielsystem
- OpenSSH-Server - Standard auf Linux
- Konfiguration in `/etc/ssh/sshd_config`

```
# SSH-Server Status prüfen  
systemctl status ssh  
systemctl status sshd
```

# Exkurs: Systemd und Dienste



# Dienste

- **Hintergrundprozesse** - Keine direkte Benutzerinteraktion
- **Starten automatisch beim Booten** oder manuell
- **Beispiel: SSH-Dienst** - sshd

# Systemd

- Init-System und Service-Manager für Linux
- **Unit-Dateien** definieren Dienste (z.B. ssh.service)
- **Befehle:** systemctl, journalctl, etc.

# SSH-Grundlagen

# Erste SSH-Verbindung

## Einfache SSH-Verbindung:

```
# Grundlegende Syntax
ssh username@hostname
ssh username@192.168.1.100

# Mit spezifischem Port
ssh -p 2222 username@hostname
```

## Bei der ersten Verbindung:

```
The authenticity of host 'example.com (192.168.1.10)' can't be established.
ED25519 key fingerprint is SHA256:abc123def456...
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

# Host Key Fingerprints

- Eindeutige Identifikation des SSH-Servers
- Schutz vor Man-in-the-Middle-Attacken
- Gespeichert in `~/ .ssh/known_hosts`

# Host Key Management

```
# Host-Schlüssel anzeigen  
ssh-keyscan example.com  
  
# Fingerprint eines bekannten Hosts anzeigen  
ssh-keygen -l -f ~/.ssh/known_hosts  
  
# Host aus known_hosts entfernen  
ssh-keygen -R example.com  
  
# Host-Schlüssel manuell hinzufügen  
ssh-keyscan example.com >> ~/.ssh/known_hosts
```

# SSH-Konfigurationsdatei

## Client-Konfiguration `~/.ssh/config`:

```
# Beispiel SSH-Config
Host webserver
    HostName web.example.com
    User deploy
    Port 2222
    IdentityFile ~/.ssh/webserver_key

Host *.internal
    User admin
    ProxyJump bastion.example.com

Host *
    ServerAliveInterval 60
    ServerAliveCountMax 3
```

## Verwendung:

```
# Statt: ssh -p 2222 deploy@web.example.com
ssh webserver
```

# Verbindungsoptionen

## Wichtige SSH-Optionen:

```
# Verbose-Modus (Debug-Informationen)
ssh -v user@host
ssh -vvv user@host # Sehr detailliert

# X11-Weiterleitung (GUI-Apps)
ssh -X user@host
ssh -Y user@host # Trusted X11-Weiterleitung

# Befehl direkt ausführen
ssh user@host 'ls -la /var/log'

# Port-Weiterleitung
ssh -L 8080:localhost:80 user@host # Local forward
ssh -R 9000:localhost:3000 user@host # Remote forward
```



# SSH-Authentifizierung

# Passwort-Authentifizierung

- Standardmethode, einfach aber unsicher
- Anfällig für Brute-Force-Angriffe
- Nicht empfohlen für produktive Systeme

# SSH-Key-Authentifizierung

- **Höhere Sicherheit** - Keine Brute-Force-Angriffe möglich
- **Komfort** - Keine Passwort-Eingabe bei jeder Verbindung
- **Automation** - Scripts und automatische Verbindungen
- **Zentrale Verwaltung** - Schlüssel können einfach widerrufen werden

# Public-Key-Kryptographie

- **Private Key** - Bleibt auf dem Client, niemals weitergeben
- **Public Key** - Wird auf dem Server gespeichert
- **Mathematische Beziehung** - Nur zusammengehörige Schlüssel funktionieren

# SSH-Keys erstellen

## SSH-Schlüsselpaar generieren:

```
# Standard RSA-Schlüssel (nicht mehr empfohlen)
ssh-keygen -t rsa -b 4096 -a 100

# Empfohlen: Ed25519-Schlüssel
ssh-keygen -t ed25519 -a 100
```

## Interaktive Erstellung:

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_ed25519
Your public key has been saved in /home/user/.ssh/id_ed25519.pub
```

# SSH-Key-Typen

Verschiedene Schlüsselalgorithmen:

| Typ     | Sicherheit | Schlüsselgröße | Empfehlung       |
|---------|------------|----------------|------------------|
| RSA     | Gut        | 2048-4096 bit  | Veraltet         |
| Ed25519 | Sehr hoch  | 256 bit        | <b>Empfohlen</b> |
| ECDSA   | Unklar     | 256-521 bit    | Nein             |
| DSA     | Schwach    | 1024 bit       | Nein             |

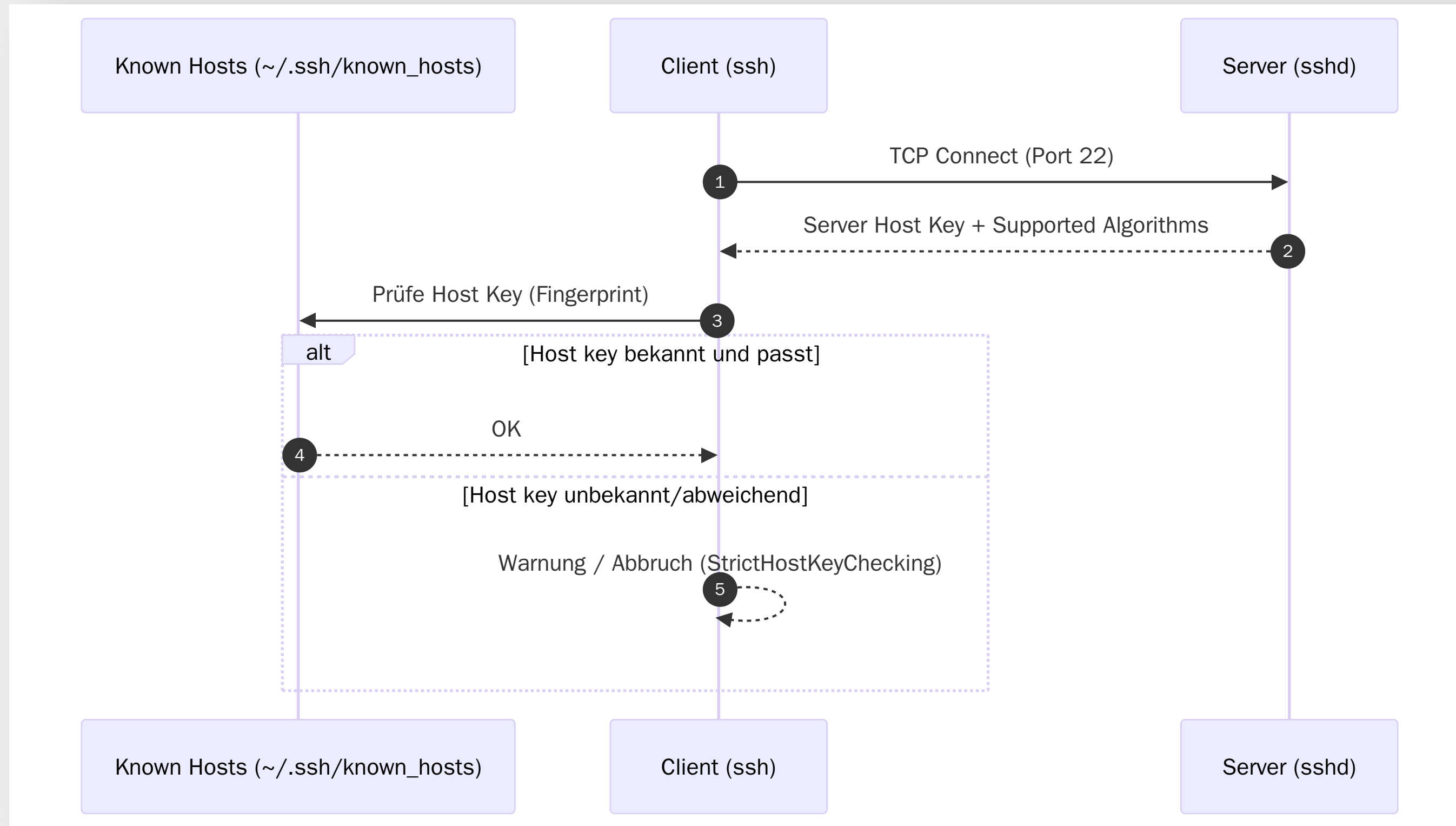
# Public Key auf Server kopieren

```
# Manuell: Public Key zum Server hinzufügen
cat ~/.ssh/id_ed25519.pub | ssh user@hostname 'cat >> ~/.ssh/authorized_keys'

# Public Key automatisch kopieren
ssh-copy-id user@hostname

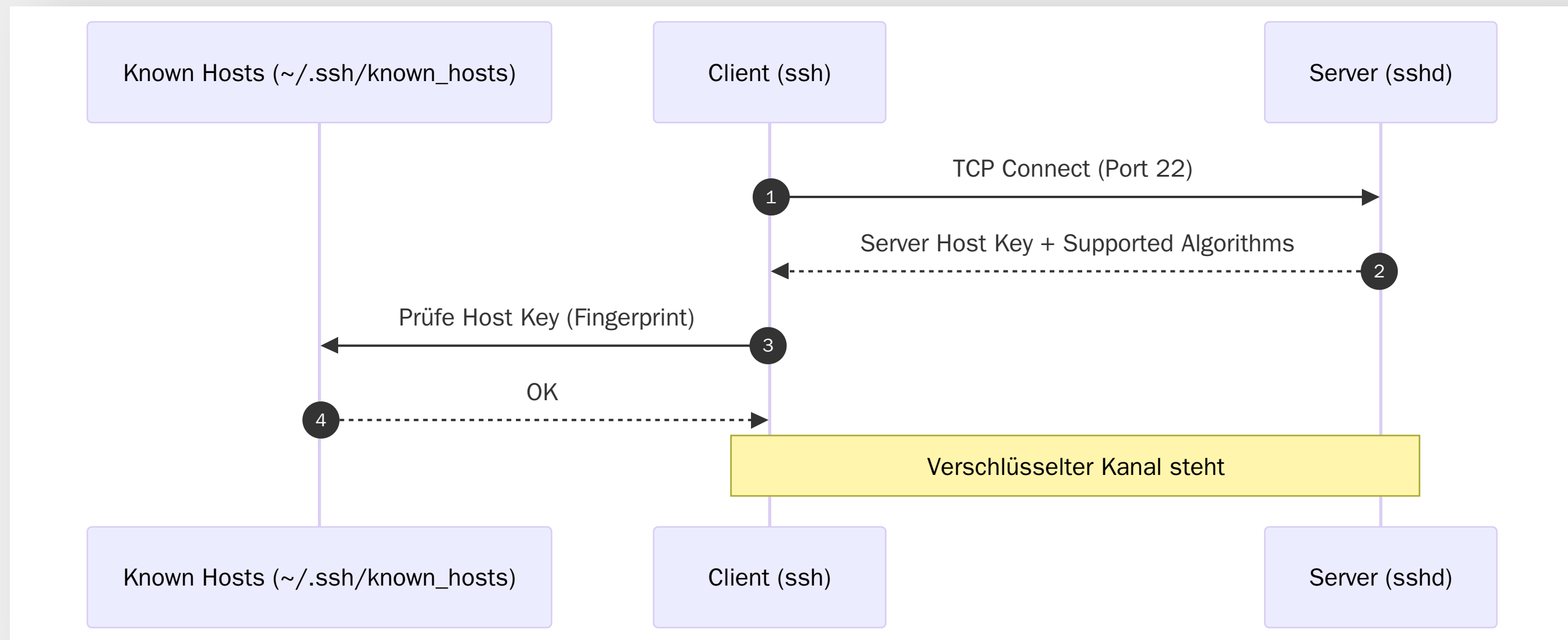
# Spezifischen Schlüssel kopieren
ssh-copy-id -i ~/.ssh/id_ed25519.pub user@hostname
```

# SSH-Verbindungsaufbau

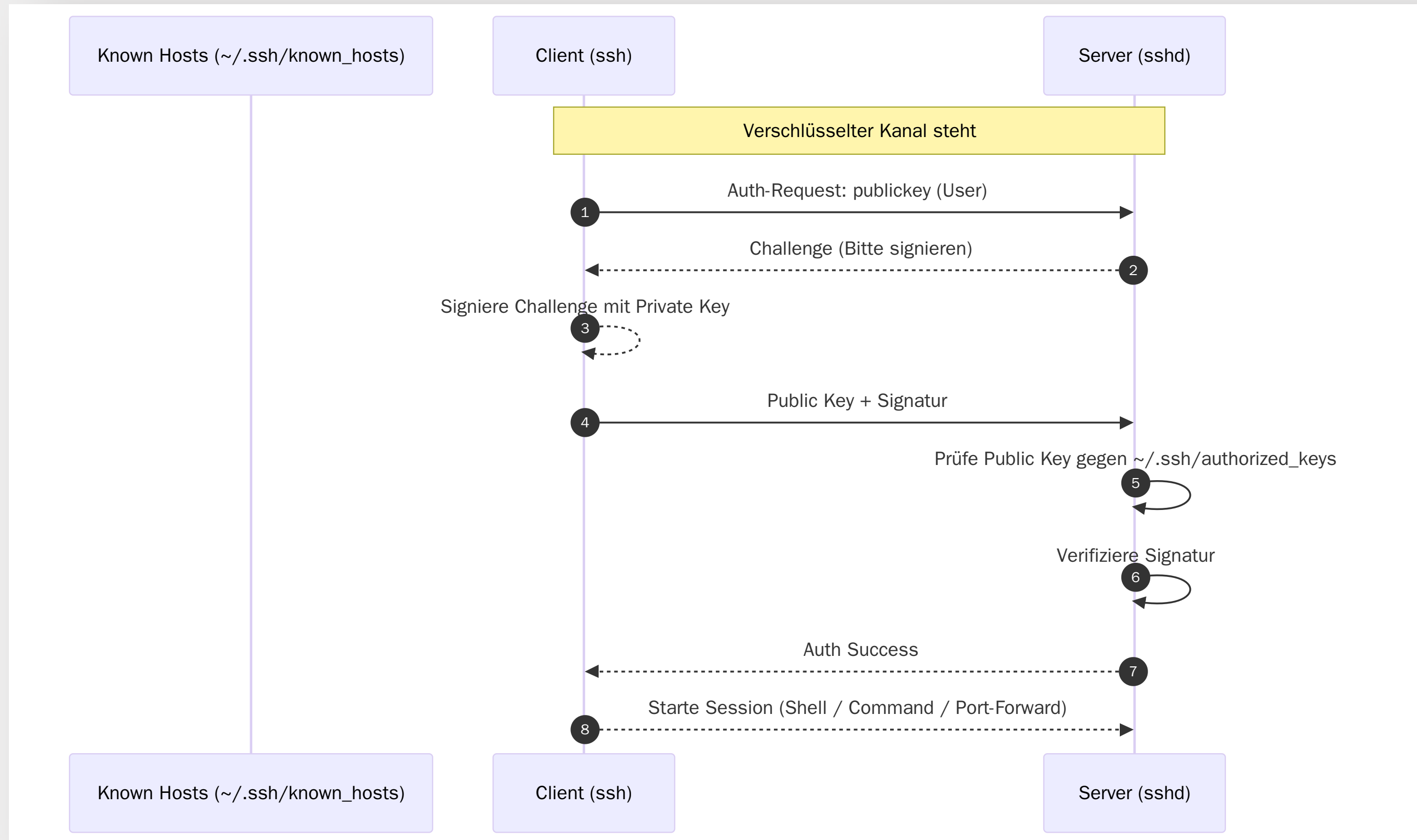




# SSH-Verbindungsaufbau



# SSH-Verbindungsaufbau



# SSH mit Jumphost

# Jumphost/Bastion Host

- **Sicherheits-Gateway** - Einziger direkter Zugang zu privaten Netzwerken
- **Vermittler-Server** - Verbindung über Zwischensystem
- **Sicherheitsschicht** - Zentrale Kontrolle und Logging

```
Internet → Jumphost → Private Server  
[Client] → [Bastion] → [Target]
```

# Jumphost Anwendungsfälle

- **Cloud-Umgebungen** - Private Subnets ohne direkten Internet-Zugang
- **Corporate Networks** - Sichere Fernwartung
- **Compliance** - Zentrale Zugriffskontrolle

# ProxyJump verwenden

## Moderne SSH ProxyJump-Option:

```
# Einfacher Proxy Jump  
ssh -J jumphost.example.com user@private-server
```

## In SSH-Config definieren:

```
# ~/.ssh/config  
Host private-server  
    HostName 192.168.10.100  
    User app-user  
    ProxyJump admin@jumphost.example.com  
  
# Jetzt einfach verwenden:  
ssh private-server
```

# SCP - Secure Copy

# Secure Copy (SCP)

- **Dateiübertragung über SSH-Verbindung**
- **Gleiche Sicherheit wie SSH**
- **Einfache Syntax ähnlich cp**
- **Authentifizierung mit SSH-Keys oder Passwort**

```
# Grundlegende Syntax
scp [Optionen] Quelle Ziel

# Von lokal zu remote
scp datei.txt user@server:/path/to/destination/

# Von remote zu lokal
scp user@server:/path/to/file.txt ./

# Zwischen zwei remote Hosts
scp user1@host1:/file user2@host2:/destination/
```



## Zur Kapitelübersicht

- Vorheriges Kapitel: [Benutzer, Gruppen und Rechte](#)
- Nächstes Kapitel: [Logs & Fehleranalyse](#)