SSH-Schlüsselgenerierung

ITS-Net-Lin

Sebastian Meisel

11. Dezember 2024

1 Einführung

SSH (Secure Shell) ermöglicht eine sichere Kommunikation zwischen Rechnern. Eine besonders sichere Methode der Authentifizierung ist die Nutzung von Schlüsselpaaren. Das Zusammenspiel von asymmetrischer und symmetrischer Verschlüsselung beim Aufbau einer SSH-Verbindung ist ein wesentlicher Bestandteil der sicheren Kommunikation.

1.1 Asymmetrische Verschlüsselung: Sicherer Schlüsselaustausch

Asymmetrische Verschlüsselung nutzt ein Schlüsselpaar, das Sie in dieser Datei zu erstellen lernen:

Öffentlicher Schlüssel (Public Key) Kann frei verteilt werden.

Privater Schlüssel (Private Key) Muss geheim bleiben.

Ablauf in SSH:

1. Der Client sendet eine Verbindungsanfrage an den Server.
2. Der Server antwortet mit seinem öffentlichen Schlüssel.
3. Der Client generiert einen zufälligen symmetrischen Sitzungsschlüssel.
4. Der Client verschlüsselt diesen Sitzungsschlüssel mit dem öffentlichen Schlüssel des Servers (asymmetrisch) und sendet ihn zurück.
5. Der Server entschlüsselt den Sitzungsschlüssel mit seinem privaten Schlüssel.

Der Austausch des Sitzungsschlüssels wird durch asymmetrische Verschlüsselung geschützt, da nur der Server den Sitzungsschlüssel entschlüsseln kann.

1.2 Symmetrische Verschlüsselung: Schnelle und effiziente Datenübertragung

Symmetrische Verschlüsselung verwendet denselben Schlüssel für Ver- und Entschlüsselung.
Ablauf in SSH:

1. Nach dem Austausch des Sitzungsschlüssels wird dieser für die symmetrische Verschlüsselung der weiteren Kommunikation verwendet.
2. Alle Daten (z. B. Befehle, Dateiinhalte) werden mit dem Sitzungsschlüssel verschlüsselt, bevor sie über die Verbindung übertragen werden.
3. Der Empfänger entschlüsselt die Daten mit demselben Sitzungsschlüssel.

Symmetrische Verschlüsselung ist wesentlich schneller und effizienter als asymmetrische Verschlüsselung, was sie ideal für den Datentransfer macht.

2 Schlüsselpaar erstellen

Mit dem modernen Algorithmus `ed25519`, der eine hohe Sicherheit bei geringer Schlüssellänge bietet, erstellt man ein Schlüsselpaar wie folgt:

```
ssh-keygen -t ed25519 -f debian
```

Parameter:

- `-t ed25519`: Verwendet den ed25519-Algorithmus.
- `-f <servername>`: Man sollte für jeden Server einen separaten Schlüssel erstellen.
- Der Befehl erzeugt zwei Dateien:
  1. `debian`: Der private Schlüssel (geheim halten!)
  2. `debian.pub`: Der öffentliche Schlüssel (wird auf dem Server gespeichert).

3 Optional: Byte-Länge anpassen

Bei Bedarf kann die Schlüssellänge angepasst werden:

- Beispiel mit 4096 Bits:

```
    ssh-keygen -t ed25519 -b 4096 -f debian
```

4 Schlüssel auf den Server übertragen

Der öffentliche Schlüssel muss auf dem Server in der Datei `~/.ssh/authorized_keys` eingetragen werden.

4.1 Methode 1: Übertragung mit `scp`

1. Öffentlichen Schlüssel übertragen:

```
    scp ~/.ssh/<server>.pub <benutzername>@<server>:~/
```

   Ersetzen Sie `<server>` durch den Servernamen, den Sie in `./ssh/config` definiert haben und `<benutzername>` durch Ihren Linuxbenutzernamen.

2. Auf dem Server den Schlüssel eintragen:

```
    cat ~/<server>.pub >> ~/.ssh/authorized_keys
    rm ~/<server>.pub
    chmod 600 ~/.ssh/authorized_keys
```

   Ersetzen Sie `<server>` durch den Servernamen, den Sie in `./ssh/config` definiert haben. Der letzte Befehl, schränkt die Rechte zum Lesen und Schreiben der Datei `~/.ssh/authorized_keys` so ein, dass nur Sie diese Rechte haben.

4.2 Methode 2: Direkte Eintragung mit `ssh-copy-id`

Alternativ kann der Schlüssel direkt eingetragen werden, allerdings steht dieser Befehl unter Windows nicht zur Verfügung :

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub benutzername@serveradresse
```

5 Zugriff testen

Nach der Einrichtung kann der Zugriff mit dem neuen Schlüssel getestet werden:

```
ssh -i ~/.ssh/<server> <benutzername>@<server>
```

5.1 Schlüssel in der Datei `~/.ssh/config` eintragen

Um die Nutzung von SSH-Schlüsseln zu vereinfachen, kann man Server-spezifische Konfigurationen in der Datei `~/.ssh/config` speichern. Damit wird die Verwendung von SSH-Schlüsseln automatisiert, ohne dass sie bei jedem Verbindungsaufbau manuell angegeben werden müssen.

5.1.1 Beispielkonfiguration

Hier ein Beispiel für einen Eintrag in `~/.ssh/config`:

```
Host Debian debian deb
    HostName       debian
    User           benutzername
    IdentityFile   ~/.ssh/debian
```

Erklärung der Optionen:

`Host Debian debian deb` Ein oder mehrere Aliasname(n), die beim SSH-Befehl verwendet werden (z. B. `ssh deb`).

`HostName debian` Die Adresse des Servers - die muss natürlich angepasst werden. Neben einem Hostnamen, kann auch eine IP-Adressen eingetragen werden.

`User benutzername` Der Benutzername, mit dem man sich auf dem Server anmeldet - auch dieser muss angepasst werden.

`IdentityFile ~/.ssh/debian` Pfad zum privaten Schlüssel - muss angepasst werden.

Vorteile der `~/.ssh/config`:

Einfachheit Der Verbindungsaufbau ist nun deutlich komfortabler:

```
ssh deb
```

Dies ist dasselbe, wie (ohne `.ssh/config`):

```
ssh -i ~/.ssh/debian benutzername@debian
```

Flexibilität Man kann mehrere Server mit unterschiedlichen Schlüsseln und Einstellungen verwalten.

5.1.2 Konfiguration testen

Nach der Einrichtung kann die Konfiguration getestet werden:

```
ssh debian
```

Wenn keine Fehlermeldungen auftreten, ist die Konfiguration erfolgreich eingerichtet.