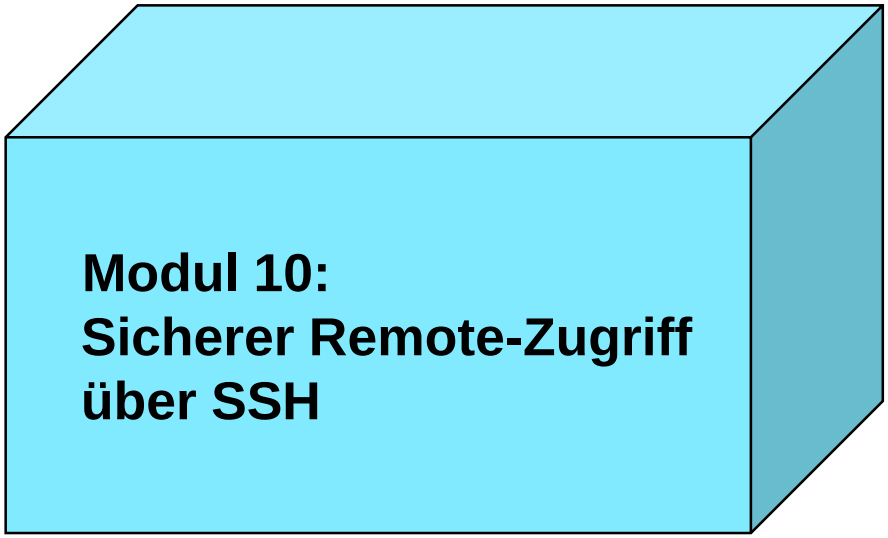
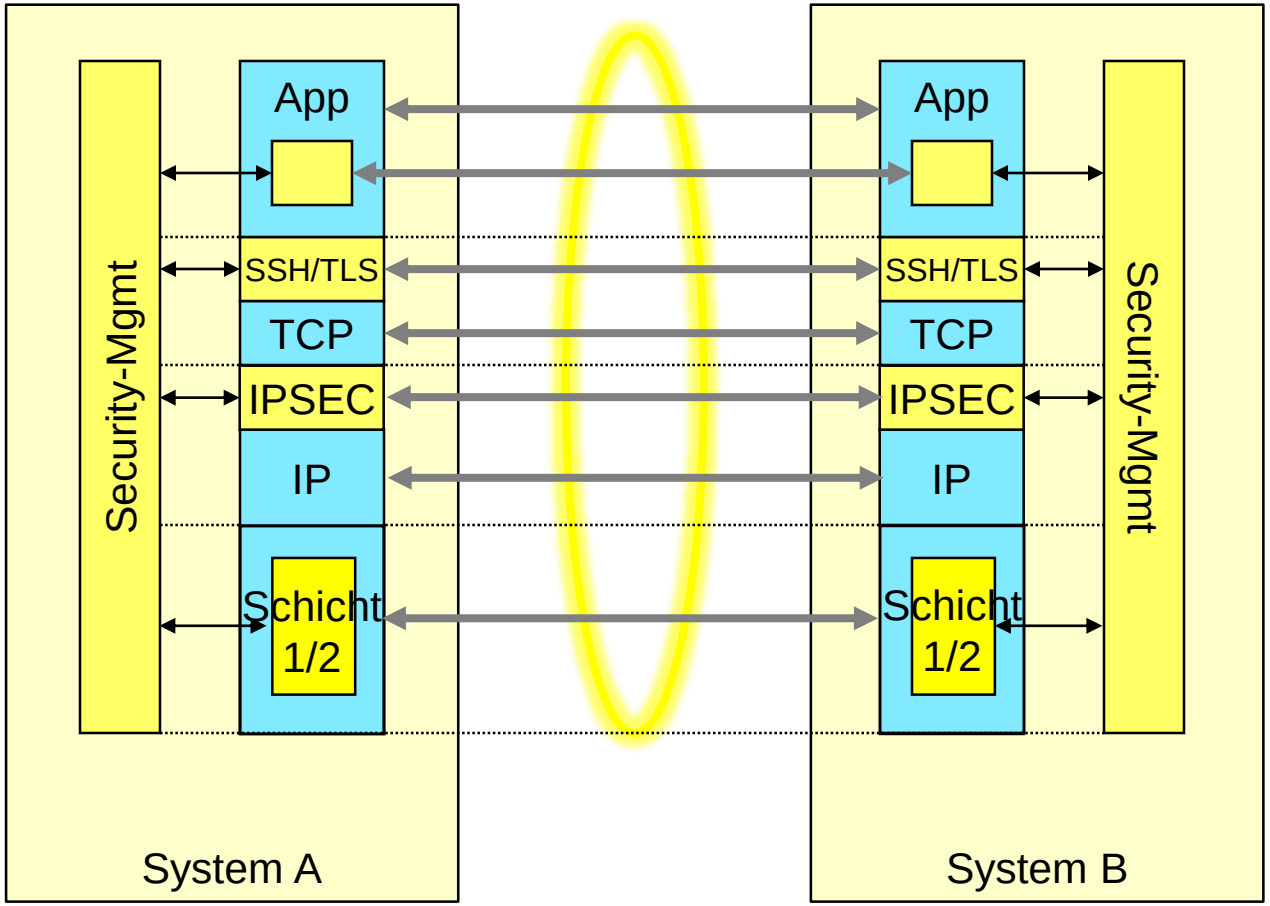


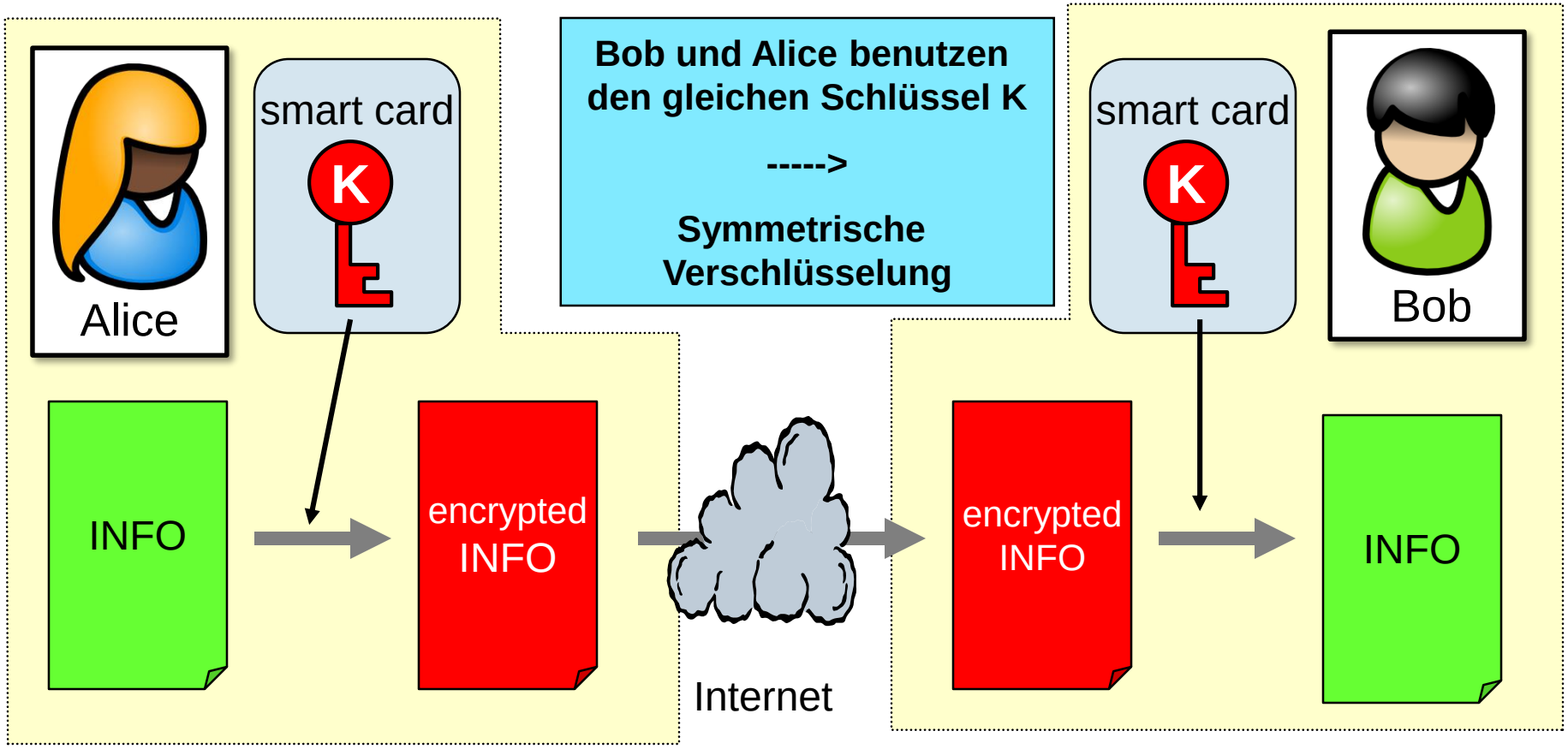
Sicherheit im Internet-Protokollstack



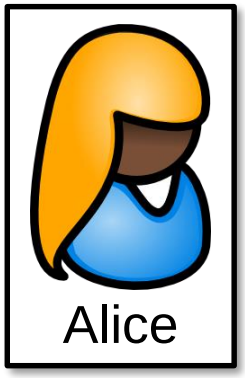
Ebenen der Sicherheit

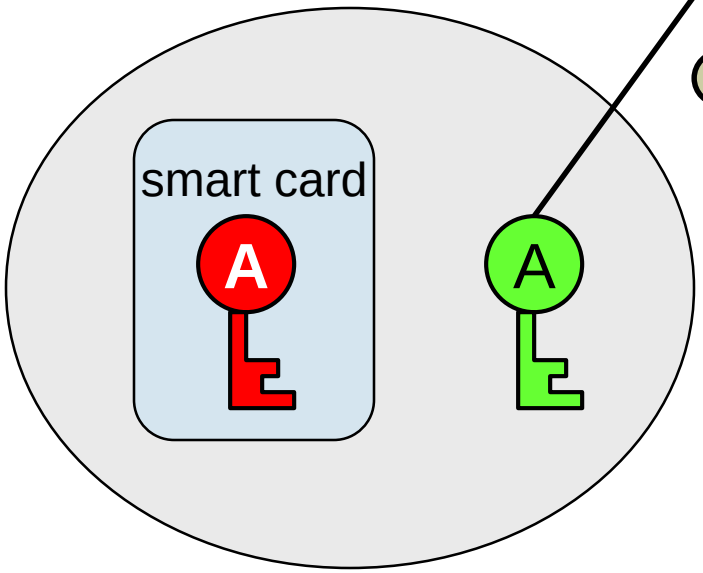
- **Sicherheit auf Applikationsebene, z.B. eCash, PGP. ("ich vertraue der Applikation")**
- **TLS: sichere Ende-zu-Ende-Verbindung ("ich vertraue dem sicheren Transport bis hin zu meiner Anwendung")**
- **SSH: Sicheres Einloggen in ein Remote-System**
- **IPsec sichere IP-Verbindung zwischen Systemen ("ich vertraue der Übermittlung über das Internet")**

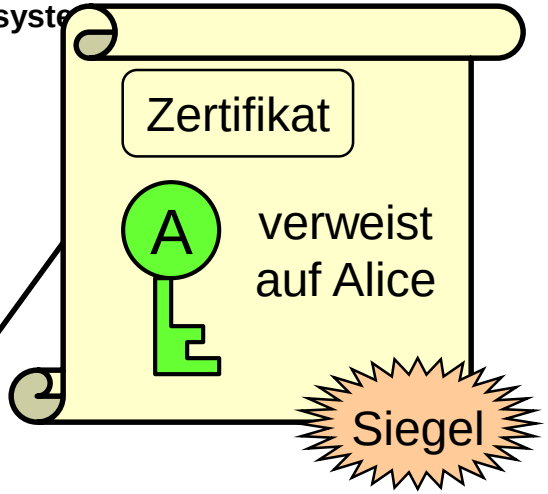
Krypto-Basic 1: Symmetrische Verschlüsselung



Krypto-Basic 2: Prinzip der asymmetrischen bzw. Public-Key-Kryptographie

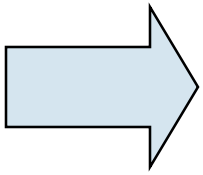






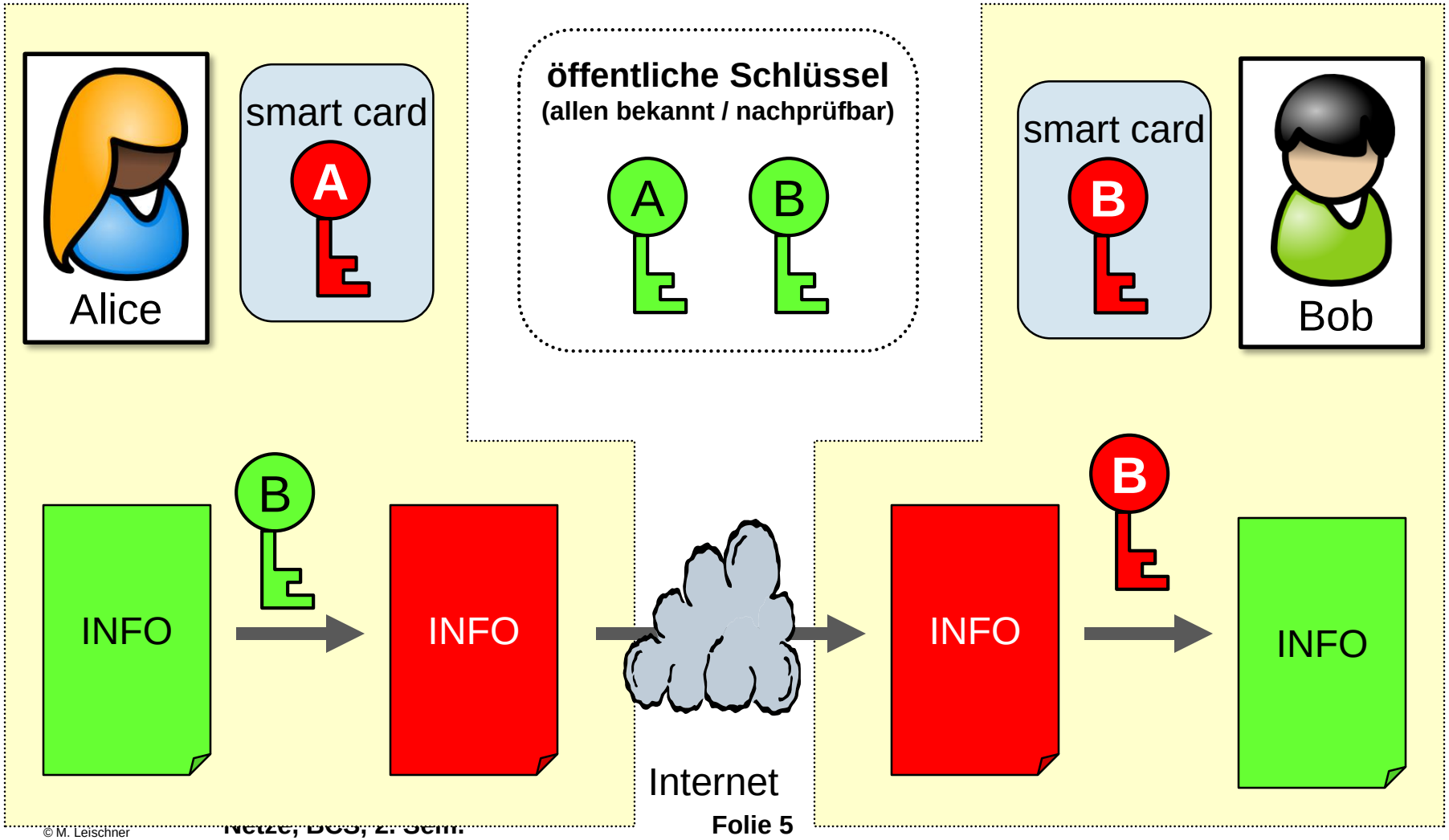
Alice hat zwei Schlüssel

- **einen secret key**
(z.B. auf einer Chipkarte)
- **einen public key**
(Dieser ist nicht geheim, kann **und sollte** in einem öffentlichen Verzeichnis gespeichert werden



Um zu verschlüsseln und um zu entschlüsseln benötigt man verschiedene Schlüssel (= asymmetrisch)

Krypto-Basic 3: Asymmetrische Verschlüsselung



SSH – Secure Shell

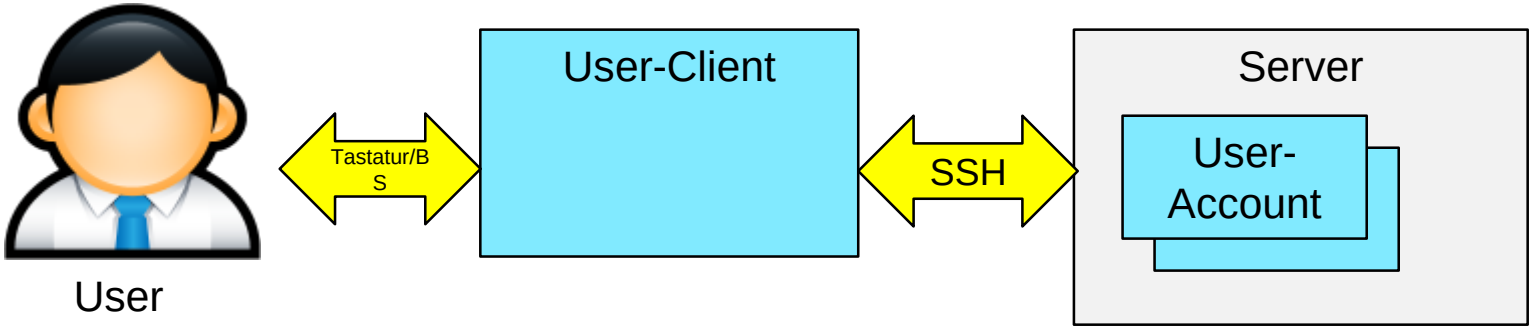
- **Haupteinsatz: Sicherer Fernzugriff auf entfernte Geräte aller Art.**
  - **Authentifizierung:**
    Nur Befugte dürfen zugreifen.
  - **Vertraulichkeit:**
    Kein Unbefugter darf Daten auf der Verbindung einsehen.
- **SSH läuft über TCP und benutzt standardmäßig den TCP-Port 22.**
- **Den Port 22 kann man in der Firewall (zu Hause) „vorsichtig“ öffnen.**
  - **Falls dahinter nur Server mit vernünftig gesichertem SSH-Zugang stehen, entsteht kein Problem.**
  - **Allerdings wird der Port 22 von Hackern Tag und Nacht angegriffen. Die Hacker hoffen, einen nicht gut gesicherten SSH-Zugang zu finden.**
  - **Daher verwenden vorsichtige Menschen für SSH statt Port 22 einen anderen Port (z.B. 22022).**
    Vorteil: höhere Sicherheit.
    Nachteil: Unorthodoxe Herangehensweise, die zu überraschenden bzw. lästigen Probleme führen kann („Ich habe zwar meinen SSH-Schlüssel, aber leider die Portnummer vergessen“).

Vorüberlegungen zu SSH

- Beteiligte sind:
  - Client
  - Server
- Reflexion: Was genau sind bei SSH die wesentlichen Sachziele bezüglich Sicherheit?
  - Vertraulichkeit?
  - Integrität?
  - Verfügbarkeit?
  - Authentizität?
  - Nachweisbarkeit (Nichtabstreitbarkeit, Non-Repudiation)?
  - Verbindlichkeit?
- Reflexion: Schlüsselmanagement bei SSH
  - Welche Schlüssel benötigen wir für SSH?
  - Wer erzeugt die Schlüssel?
  - An wen werden die Schlüssel verteilt?
  - Wie werden die Schlüssel verteilt?

Szenario: Beteiligte und Schlüssel

PW-S (Key-U_{pub}, Key-U_{sec}) hash(PW-S) (Key-S_{pub}, Key-S_{sec})



Vorhandenes, mögliches und sinnvolles Schlüsselmaterial:

- **Userpasswort von Serveraccount (PW-S)**
- **Hashwert von PW-S : hash(PW-S)**
- **Schlüsselpaar User (Key-U_{pub}, Key-U_{sec})**
- **Schlüsselpaar Server (Key-S_{pub}, Key-S_{sec})**

Ferner:

Einfachster Fall:

User möchte sich über SSH mit seinem UserAccount-Passwort auf den Server einloggen.

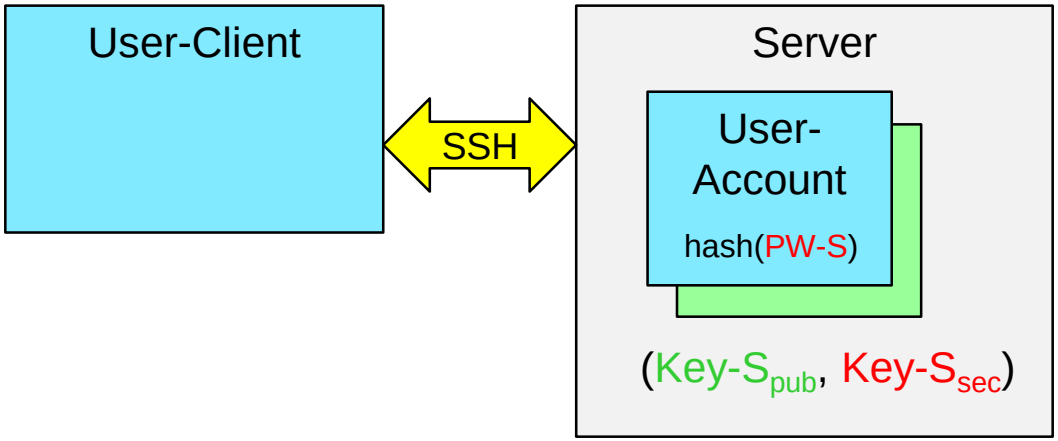
Welche Schlüssel brauchen wir?

Welche Schlüssel kommen wohin?

Szenario 1: SSH-Basisszenario

User

PW-S



Vorhandene Schlüssel:

- **Userpasswort von Serveraccount (PW-S)**
- **Schlüsselpaar Server (Key-S$_{pub}$, Key-S$_{sec}$)**
- **Schlüsselpaar User (Key-U$_{pub}$, Key-U$_{sec}$)**
  (nur notwendig, wenn User-Authentisierung mit öffentlichem Schlüssel erfolgen soll)

Hashwert von PW-S : hash(PW-S)

SSH – Protokollablauf vereinfacht (Client/Server-Szenario)

Voraussetzung: Server besitzt asymmetrisches Schlüsselpaar (Key-S_{pub}, Key-S_{sec})

- **Schritt 1 – Client:**
  - **TCP-Verbindung aufbauen**
  - **Protokollversion austauschen**
- **Schritt 2 - Server:**
  - **sendet öffentlichen Host-Key Key-S_{pub} und**
  - **Liste der unterstützte Verschlüsselungsalgorithmen**
- **Schritt 3 - Client:**
  - **akzeptiert Host-Key (bereits bekannt bzw. User fragen)**
  - **wählt Verschlüsselungsalgorithmus**
  - **generiert symmetrischen Session-Key**
  - **sendet diesen Session-Key verschlüsselt mit öffentlichem Host-Key**
- **Schritt 4 - Server:**
  entnimmt Session-Key und schaltet auf Verschlüsselung um.
- **Schritt 5 - Client:**
  authentifiziert sich in geeigneter Weise
  (das ist der entscheidender Punkt! Es könnte ja jeder kommen.)

SSH – Protokollablauf vereinfacht

- **Methoden der Clientauthentifizierung**
  - **User-Passwort. User loggt sich über SSH als user mit Passwort PW-S ein**
  - **Authentifizierung über öffentliche Schlüssel:**
    - **User erzeugt im Vorfeld ein asymmetrisches Schlüsselpaar (Key-U_{pub}, Key-U_{sec}) und hinterlegt im Vorfeld seinen öffentlichen Schlüssel Key-U_{pub} beim Server.**
    - **Der Client verschlüsselt den öffentliche Server-Schlüssel Key-S_{pub} mit dem geheimen Userschlüssel Key-U_{sec}. und sendet das Ergebnis**

      ERG = $enc_{Key\text{-}Usec}$ (Key-S_{pub})

      an den Server.
    - **Der Server besitzt den öffentlichen Schlüssel Key-U_{pub} des Users. Mit diesem entschlüsselt er das Ergebnis ERG . Kommt als bei der Entschlüsselung Key-S_{pub} heraus, weiß er, dass der Client den echten geheimen Schlüssel besitzt.**

- **Verbesserungen bei SSH2:**
  - **Aufbau einer verschlüsselten Verbindung über ein ephemerales Diffie-Hellman-Verfahren.**
    Ephemeral: Die Schlüssel gelten nur temporär und werden nach Ablauf der Verbindung weggeschmissen.
    Vorteil: Verbindungen können nachträglich nicht mehr entschlüsselt werden
    (→ Perfect Forward Secrecy (PFS))
    Nachteil: recht aufwändig.
  - **Es sind nur bessere Algorithmen zugelassen.**

SSH – praktische Anwendung

SSH über Passwort:

1. `apt install openssh-server` : SSH-Server installieren.
   Es wird eine Server-Schlüsselpaar erzeugt und die App installiert.
2. Jetzt kann man sich per ssh und mit seinem Passwort in den Server einwählen.
   Befehl hierfür: `ssh user@server`.

3. Reflexion:
   - Wie sicher ist SSH über Passwort?
   - Gibt es sinnvolle Einsatzszenarien für SSH über Passwort
   - Macht es bezüglich Sicherheit einen Unterschied, ob IPv6 oder IPv4 eingesetzt wird?

SSH – praktische Anwendung

SSH mit öffentlichem Schlüssel:

1. **Zunächst benötigt man ein Schlüsselpaar.**
   Falls nicht vorhanden, kann dieses z.B. mit `ssh-keygen -t rsa -b 4096` erzeugt werden.
   `-t rsa` : Typ des Schlüssels
   `-b 4096` : Anzahl der Bits (Schlüssellänge)
   Bei Bedarf kann der geheime Schlüssel zusätzlich mit eine Passwort geschützt werden.
2. **Der öffentliche Schlüssel wird als ASCII (`ssh-rsa AAA.....`) als erste bzw. weitere Zeile in die Server-Datei `~/.ssh/authorized_keys` des Homeverzeichnisses des Benutzers eintragen.**
3. **Der geheime Schlüssel wird auf dem Client für ssh verfügbar gemacht. (Hierzu ist der geheime Schlüssel in das Verzeichnis `~/.ssh/id_rsa` zu kopieren.**
4. **Zugriff mit öffentlichem Schlüssel testen, d.h. für `ssh user@server` wird kein Passwort mehr benötigt (obwohl das Passwort noch aktiv ist).**

SSH – praktische Anwendung

Passwortzugriff abschalten (da Sicherheitslücke):

Auf dem Server die Datei `/etc/ssh/sshd_config` wie folgt anpassen:

```
ChallengeResponseAuthentication no

PasswordAuthentication no

UsePAM yes
```

Zugriff testen. Fertig!

<u>Anmerkung</u>: Es ist tatsächlich nur die Zeile

```
PasswordAuthentication no
```

anzupassen. Alles andere ist bei der aktuellen ssh-Konfiguration schon als Standard voreingestellt.