

---

# Laborprotokoll

## Fernwartungssysteme

---

Systemtechnik Labor  
5BHIT 2017/18

Martin Wölfer

Note:  
Betreuer: UMAA

Version 0.1  
Begonnen am 19. Oktober 2017  
Beendet am 23. November 2017

# Inhaltsverzeichnis

<b>1</b>	<b>Aufgabenstellung</b>	<b>1</b>
1.1	Übungen- Remote Shell . . . . .	1
1.1.1	SSH-Server Public key Authentifizierung . . . . .	1
1.1.2	Konfiguration eines SSH-Tunnels für den Internet-Zugriff über einen remote Server . . . . .	1
1.2	Remote Network . . . . .	1
1.2.1	Installation und Konfiguration eines VPN Gateway (StrongSwan) . . . . .	1
1.2.2	Installation und Konfiguration einer Certificate Authority . . . . .	1
<b>2</b>	<b>Ergebnisse</b>	<b>2</b>
2.1	Public Key Authentifizierung . . . . .	2
2.1.1	ssh-key generieren . . . . .	2
2.1.2	Key auf Server übertragen . . . . .	2
2.1.3	Auf server zugreifen . . . . .	3
2.2	SSH-Tunnel Zugriff über Remote-Server . . . . .	4
2.2.1	Server vorbereiten . . . . .	4
2.2.2	Client vorbereiten . . . . .	5
2.2.3	Mit Client zu Server verbinden . . . . .	5
2.3	ipsec . . . . .	7

# 1 Aufgabenstellung

## 1.1 Übungen- Remote Shell

### 1.1.1 SSH-Server Public key Authentifizierung

Es soll ein SSH-Zugang für eine Authentifizierung mittels Public-Key-Verfahren konfiguriert werden. Dazu soll am Client ein Schlüsselpaar erstellt, der öffentliche Teil der Schlüssel auf den Server übertragen und anschließend der Server für die Schlüssel-Authentifizierung eingerichtet werden. Anschließend soll der Benutzer ohne Login-Passwort am Server anmelden können.

[https://www.thomas-krenn.com/de/wiki/OpenSSH\\_Public\\_Key\\_Authentifizierung\\_unter\\_Ubuntu](https://www.thomas-krenn.com/de/wiki/OpenSSH_Public_Key_Authentifizierung_unter_Ubuntu)

### 1.1.2 Konfiguration eines SSH-Tunnels für den Internet-Zugriff über einen remote Server

Es soll ein sicherer Fernzugriff via TightVNC zum einem Linux-Server ermöglicht werden. Dazu soll SSH-Tunnel eingesetzt werden.

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-vnc-on-ubuntu-16-04>

<https://www.theurbanpenguin.com/creating-an-ssh-tunnel-with-putty-to-secure-vnc/>

## 1.2 Remote Network

### 1.2.1 Installation und Konfiguration eines VPN Gateway (StrongSwan)

- Konfiguration IPsec Site-to-Site VPN
- Konfiguration IPsec End-to-Site VPN („Roadwarrior“)

### 1.2.2 Installation und Konfiguration einer Certificate Authority

- Aktivierung der Authentisierung über Zertifikate

<https://console.kim.sg/strongswan-ipsec-vpn-with-pre-shared-key-and-certificates/>

## 2 Ergebnisse

### 2.1 Public Key Authentifizierung

Client: 10.0.2.7

Server: 10.0.2.6

#### 2.1.1 ssh-key generieren

Das Schlüsselpaar wird generiert mit `ssh-keygen -b 4096`. Es wird anschließend nach einem Verzeichnis gefragt, wo der **private** und **öffentliche** Schlüssel gespeichert wird.

Der private Schlüssel hat die Bezeichnung `key_rsa` und der öffentliche `key_rsa.pub`.

```
bitte@vojnwoelf:~$ ssh-keygen -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bitte/.ssh/id_rsa): /home/bitte/.ssh/
key_rsa
Created directory '/home/bitte/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bitte/.ssh/key_rsa.
Your public key has been saved in /home/bitte/.ssh/key_rsa.pub.
The key fingerprint is:
SHA256:r6VXdT791Cs8TM5//WJBv9aeOG6IrGMGyFD+SWoTNT bitte@vojnwoelf
The key's randomart image is:
+---[RSA 4096]----+
|      E.          |
|    . o.          |
|   o . .          |
| . o .           o .|
| o * . S      o +o|
| * + . . . . *   |
| . . . . +.B  oB  |
|   + *.. X++*    |
|    o.=. o+*==   |
+-----[SHA256]-----+
```

Abbildung 1: ssh-key wird generiert

#### 2.1.2 Key auf Server übertragen

Um den Key auf den Server zu übertragen, müssen Client und Server zuerst in einem Netzwerk sein. Dies wurde umgesetzt mit einem NAT-Netzwerk.

Der öffentliche Schlüssel wurde auf den Server kopiert mit:

```
ssh-copy-id -i .ssh/key_rsa.pub bitte@10.0.2.6
```

Es musste lediglich das Passwort des Benutzers am Server angegeben werden.

```
bitte@vojnwoelf:~$ sudo ssh-copy-id -i .ssh/key_rsa.pub bitte@10.0.2.6
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/key_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
bitte@10.0.2.6's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'bitte@10.0.2.6'"
and check to make sure that only the key(s) you wanted were added.
```

Abbildung 2: Key wird auf Remote-Server übertragen

### 2.1.3 Auf server zugreifen

Es kann sich nun am Server eingeloggt werden mit:

```
ssh bitte@10.0.2.6
```

Wichtig: Es wird nur mehr nach der Passphrase gefragt und nicht nach dem öffentlichen Schlüssel.

```
bitte@vojnwoelf:~$ ssh bitte@10.0.2.6
The authenticity of host '10.0.2.6 (10.0.2.6)' can't be established.
ECDSA key fingerprint is SHA256:FbqbumSGMG0/QbpwTSM8eqFkkUniIK600CpA7/wVAXg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.6' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-97-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

558 packages can be updated.
300 updates are security updates.
```

Abbildung 3: Es wird auf den Server zugegriffen

Auch zu beachten ist, nach dem ersten verbinden durch ssh wird auch nicht mehr nach der Passphrase gefragt.

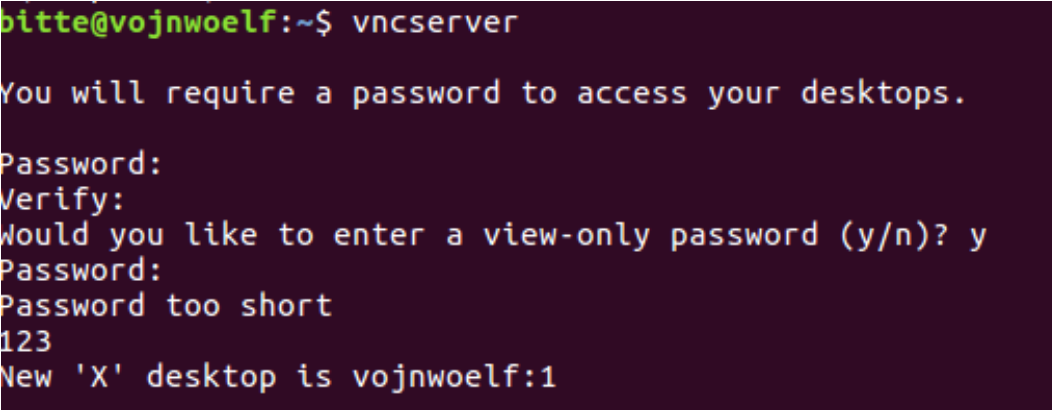
## 2.2 SSH-Tunnel Zugriff über Remote-Server

### 2.2.1 Server vorbereiten

Um auf den Server über eine grafische Oberfläche zuzugreifen muss zuerst ein Desktop Environment und ein VNC-Server installiert werden.

```
sudo apt-get install xfce4 xfce4-goodies tightvncserver
```

Nach der Installation wird der VNC-Server konfiguriert mit dem Befehl `vncserver`:

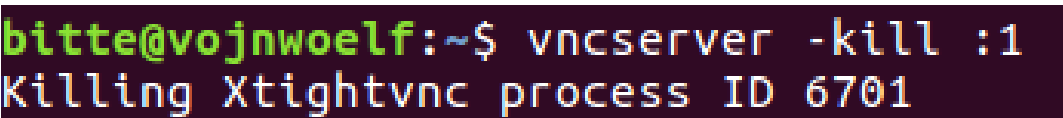


```
bitte@vojnwoelf:~$ vncserver
You will require a password to access your desktops.
Password:
Verify:
Would you like to enter a view-only password (y/n)? y
Password:
Password too short
123
New 'X' desktop is vojnwoelf:1
```

Abbildung 4: Der VNC-Server wird konfiguriert

Tatsächlich wird nur nach einem Passwort für Remote Control (Tastatur und Maus) und View-Only definiert.

Nun soll noch geändert werden was passiert wenn ein VNC-Server gestartet wird. Dafür wird zuerst der service gekilled mit `vncserver -kill :1`:



```
bitte@vojnwoelf:~$ vncserver -kill :1
Killing Xtightvnc process ID 6701
```

Abbildung 5: Der VNC-Server Service wird gekilled für Bearbeitung

Anschließend wird das File geöffnet welches bestimmt welche Aktionen durchgeführt werden wenn der VNC-Server gestartet wird mit `sudo nano ~/.vnc/xstartup`

Falls Inhalt bereits vorhanden ist, wird dieser aus dem File gelöscht, und folgender Inhalt wird hinzugefügt:

```
1 #!/ bin/bash
  xrdp $HOME/.Xresources
3 startxfce4 &
```

Der Befehl `xrdb $HOME/.Xresources` kümmert sich darum, dass der User bestimmte Einstellungen zu der grafischen Oberfläche machen kann wie Farben ändern, Designs oder Schriftarten.

Der zweite Befehl `startxfce4 &` startet lediglich Xfce, was die komfortable Bearbeitung durch die grafische Oberfläche ermöglicht.

Es müssen diesem File auch noch bestimmte Privilegien zugewiesen werden:

```
sudo chmod +x ~/.vnc/xstartup
```

Der service kann nun durch `vncserver` gestartet werden, der output sollte folgendermaßen aussehen:

```
bitte@vojnwoelf:~$ vncserver
New 'X' desktop is vojnwoelf:1
Starting applications specified in /home/bitte/.vnc/xstartup
Log file is /home/bitte/.vnc/vojnwoelf:1.log
```

Abbildung 6: Der VNC-Server wird wieder gestartet

### 2.2.2 Client vorbereiten

In diesem Fall stellt die lokale Windows-Maschine den Client dar. Es werden 2 Programme benötigt, PuTTY und TightVNC Viewer.

Mit PuTTY wird der SSH-Tunnel geöffnet und TightVNC Viewer greift über diesen Tunnel auf die grafische Oberfläche des Servers zu.

### 2.2.3 Mit Client zu Server verbinden

Zuerst wird mit PuTTY der Tunnel erstellt.

Dazu wird das Programm geöffnet und als IP-Adresse wird `bitte@10.0.106.174` angegeben und es soll über SSH verbunden werden:

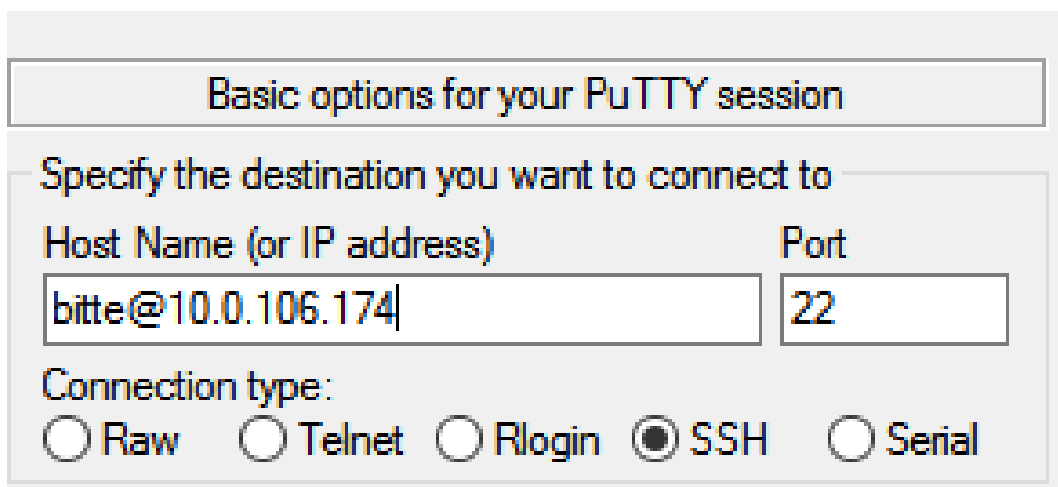


Abbildung 7: Es werden user, Ip-Adresse und Übertragungsart angegeben

Der nächste Schritt ist es nun den Tunnel zu definieren. Diese Einstellung ist zu finden unter `Connection→SSH→Tunnels`.

Hier wird als `Source Port` ein beliebiger Port angegeben (Für das Beispiel wurde 9090 gewählt) und als `Destination` wird der VNC-Server angegeben, mit der Adresse `localhost:5901`. Es kann auf `Add` gedrückt werden und man sollte folgendes Ergebnis sehen:

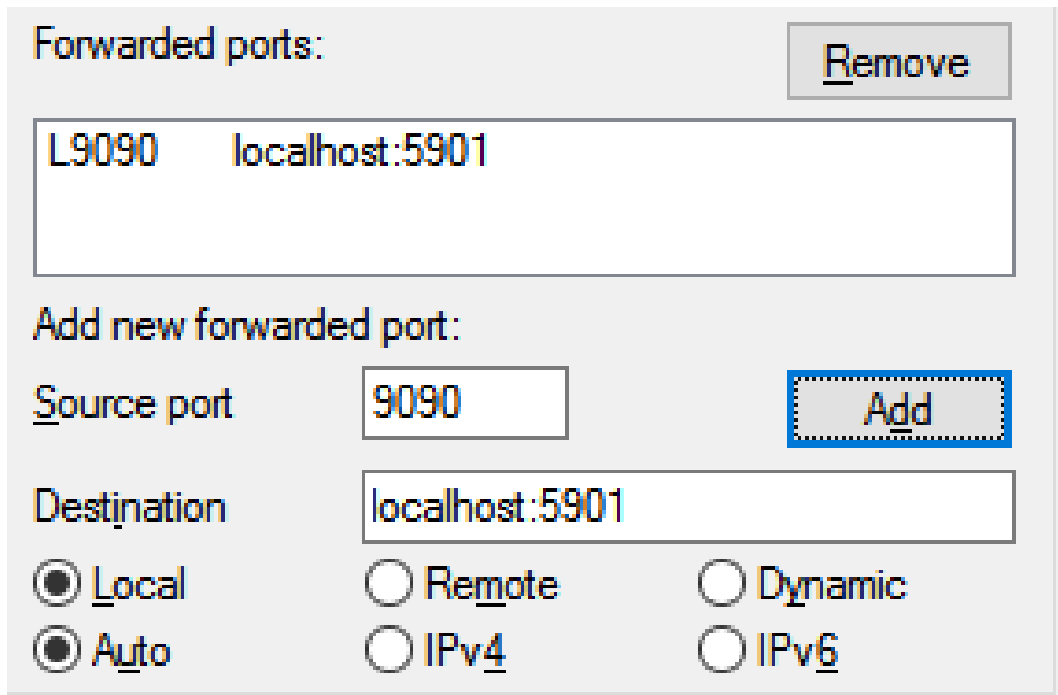


Abbildung 8: Es wurde ein Tunnel zum VNC Server hinzugefügt

Nun kann auf **Open** gedrückt werden und man sollte nach dem User-Passwort gefragt werden. Nach der Passwordeingabe sollte man die Konsole des Servers sehen:

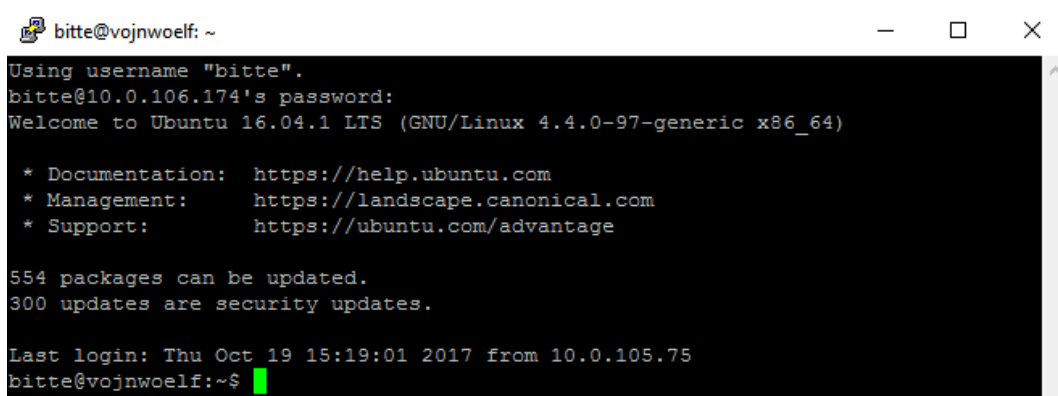


Abbildung 9: Es wurde zum Server verbunden

Das wichtige ist nun im **Event Log** zu sehen, wenn man diesen öffnet (Rechtsklick auf Statusleiste→Event Log) sollte man folgende 3 Zeilen sehen:

```
2017-10-19 15:54:23 Opening session as main channel
2017-10-19 15:54:23 Opened main channel
2017-10-19 15:54:23 Local port 9090 forwarding to localhost:5901
```

Abbildung 10: Ein Channel mit Port forwarding zum Server wurde geöffnet

Nun kann man den **TightVNC Viewer** öffnen. Im Feld **Remote Host** gibt man nun **localhost:9090** an und wird wieder nach dem Passwort gefragt. Nach Eingabe des Passworts sollte die grafische Oberfläche des Server zu sehen sein:



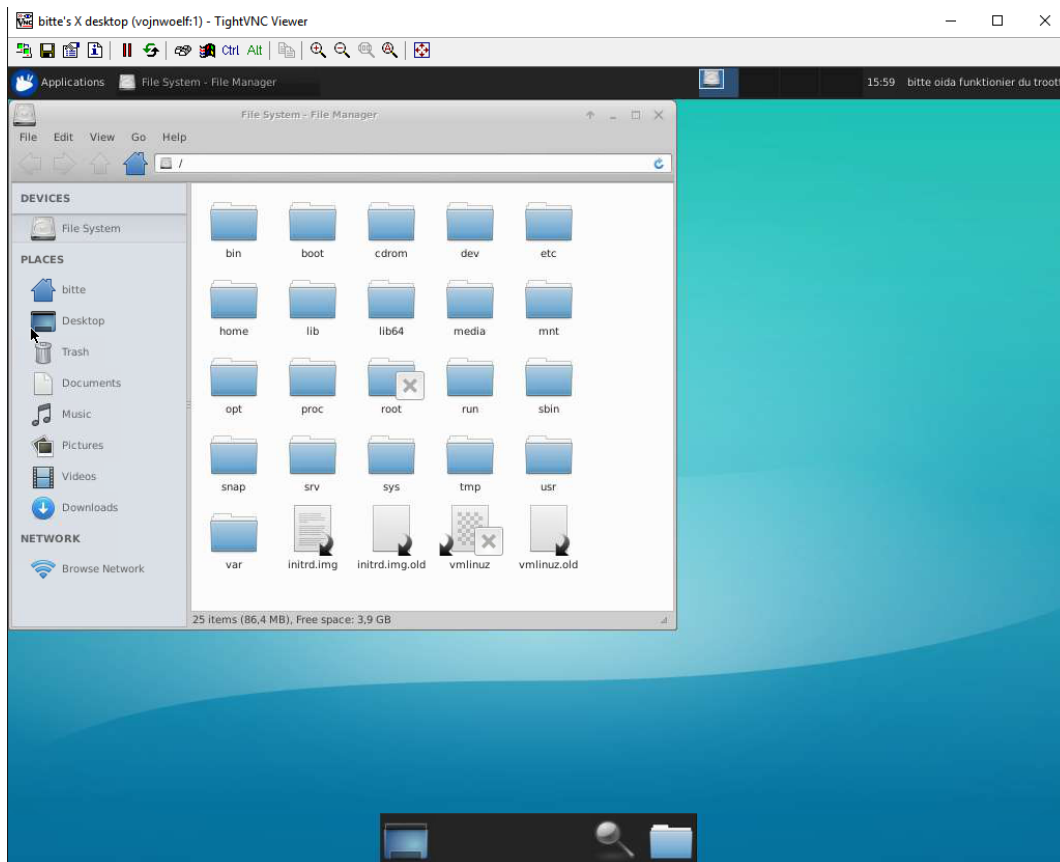


Abbildung 11: Es kann über eine grafische Oberfläche auf den Server zugegriffen werden

Es wird nun mit dem Viewer auf den verschlüsselten SSH-Tunnel zugegriffen welcher eine Verbindung zum Server hat!

## 2.3 ipsec

Zuerst wurde auf den beiden virtuellen Maschinen **strongswan**

```
1 sudo apt-get install strongswan
```

Zusätzlich wird auch **haveged** installiert und der Dienst gestartet:

```
1 sudo apt-get install haveged
2 sudo systemctl enable haveged
3 sudo systemctl start haveged
```

Der nächste Schritt ist es auf den beiden Maschinen den gleichen **PSK (Pre Shared Key)** zu definieren im **/etc/ipsec.secrets** file:

left

right

```
1 10.0.2.8 10.0.2.9 : PSK 'schueler'
```

```
1 10.0.2.9 10.0.2.8 : PSK 'schueler'
```

Zum Schluss müssen noch die individuellen Config Files zu bearbeiten, diese liegen jeweils in `/etc/ipsec.conf`:

left

right

```

1  conn A_TO_B
   authby=secret
3  left=10.0.2.8
   leftsubnet=10.0.2.0/24
5  right=10.0.2.9
   rightsubnet=10.0.2.0/24
7  ike=aes256-sha2_256-modp1024!
   esp=aes256-sha2_256!
9  keyingtries=0
   ikelifetime=1h
11 lifetime=8h
   dpddelay=30
13 dpdtimeout=120
   dpdaction=restart
15 auto=start

```

```

1  conn B_TO_A
   authby=secret
3  left=10.0.2.9
   leftsubnet=10.0.2.0/24
5  right=10.0.2.8
   rightsubnet=10.0.2.0/24
7  ike=aes256-sha2_256-modp1024!
   esp=aes256-sha2_256!
9  keyingtries=0
   ikelifetime=1h
11 lifetime=8h
   dpddelay=30
13 dpdtimeout=120
   dpdaction=restart
15 auto=start

```

Danach muss ipsec auf beiden Maschinen neu gestartet werden mit `sudo ipsec restart` und danach kann schon der Tunnel getestet werden mit `sudo ipsec up A_TO_B` auf Maschine A bzw. `sudo ipsec up B_TO_A` auf Maschine B. Tatsächlich wird aber der Befehl nur auf einer Maschine ausgeführt da der Tunnel zwischen den beiden Geräten aufgebaut wird.

```

bitte@vojnwoelf:~$ sudo ipsec up A_TO_B
establishing CHILD_SA A_TO_B
generating CREATE_CHILD_SA request 2 [ SA No TSi TSr ]
sending packet: from 10.0.2.8[4500] to 10.0.2.9[4500] (208 bytes)
received packet: from 10.0.2.9[4500] to 10.0.2.8[4500] (208 bytes)
parsed CREATE_CHILD_SA response 2 [ SA No TSi TSr ]
CHILD_SA A_TO_B{2} established with SPIs ceb9f824_i ce4b517a_o and TS 10.0.2.0/2
4 == 10.0.2.0/24
connection 'A TO B' established successfully

```

Abbildung 12: Es wurde erfolgreich der Tunnel zwischen den beiden Geräten aufgebaut

Um zu testen ob die Pakete mit **ESP** verschlüsselt werden, wird von einer Maschine auf die andere gepingt, während die andere `sudo tcpdump esp` ausführt.

Es sollten dauerhaft **responses** und **requeests** zu sehen sein, welche mit **ESP** verschlüsselt wurden.

```

15:56:56.183686 IP 10.0.2.9 > 10.0.2.8: ESP(spi=0xc6f1a002,seq=0x9a), length 136
15:56:57.185608 IP 10.0.2.8 > 10.0.2.9: ESP(spi=0xc100cf1f,seq=0x9b), length 136

```

Abbildung 13: Pakete werden mit ESP verschlüsselt

## Abbildungsverzeichnis

1	ssh-key wird generiert . . . . .	2
2	Key wird auf Remote-Server übertragen . . . . .	3
3	Es wird auf den Server zugegriffen . . . . .	3
4	Der VNC-Server wird konfiguriert . . . . .	4
5	Der VNC-Server Service wird gekilled für Bearbeitung . . . . .	4
6	Der VNC-Server wird wieder gestartet . . . . .	5
7	Es werden user, Ip-Adresse und Übertragungsart angegeben . . . . .	5
8	Es wurde ein Tunnel zum VNC Server hinzugefügt . . . . .	6
9	Es wurde zum Server verbunden . . . . .	6
10	Ein Channel mit Port forwarding zum Server wurde geöffnet . . . . .	6
11	Es kann über eine grafische Oberfläche auf den Server zugegriffen werden . . . . .	7
12	Es wurde erfolgreich der Tunnel zwischen den beiden Geräten aufgebaut . . . . .	8
13	Pakete werden mit ESP verschlüsselt . . . . .	8