

# **Teil III**

## **Fortgeschrittene Konfiguration und Fehlersuche**



# 9 Fortgeschrittene Konfiguration

Dieses Kapitel beschäftigt sich mit der fortgeschrittenen Konfiguration von VPN Lösungen mit Linux. Hierbei werden Probleme und Lösungen besprochen, die über das bisher Gesagte hinaus gehen. Die Abhandlung dieser Probleme erfolgt teilweise ausführlicher, bisweilen aber auch recht knapp. Dies hängt auch mit der nur experimentellen Unterstützung einiger Lösungen durch FreeS/WAN und `racoon` zusammen. In den Fällen, in denen die Darstellung knapp gehalten werden musste, sollen Verweise auf weiterführende Literatur im Internet den Leser über neueste Entwicklungen informieren. In vielen Fällen ist die Unterstützung für `racoon` auch noch nicht so weit wie in dem Fall von FreeS/WAN. Dann wird lediglich auf FreeS/WAN eingegangen.

## 9.1 Aufbau einer Verbindung mit dynamischen IP Adressen auf beiden Seiten.

Der Aufbau einer VPN Verbindung, bei der beide Seiten dynamische IP Adresse verwenden, stellt ein Problem dar, da zunächst keine der beiden Seiten die IP Adresse des Partners kennt.

Als Lösung für dieses Problem kann ein Dienst wie Dyn-DNS (<http://www.dyndns.org>) genutzt werden.<sup>1</sup> Dieser Dienst ermöglicht es einem Rechner mit dynamischer IP Adresse einen festen DNS-Namen zu erhalten. Der Rechner ist dann immer unter demselben Namen erreichbar, obwohl er immer über eine andere IP Adresse verfügt. Hierzu meldet sich der Rechner bei jeder Einwahl bei dem Dyn-DNS-Dienst an und übergibt seine aktuelle IP Adresse. Der Dyn-DNS-Server trägt diese IP Adresse ein und ermöglicht so die Auflösung des DNS-Namen auf wechselnde IP Adressen.

Wenn der Tunnel aufgebaut werden soll, muss nur sichergestellt werden, dass die Seite, die den Tunnel aufbaut, zunächst die IP Adresse der Gegenseite durch eine DNS-Auflösung ermittelt. Die Gegenseite muss im Vorfeld ihre IP Adresse bei dem Dyn-DNS-Dienst eingetragen haben.

---

1. Ein Überblick über weitere ähnliche Dienste ist auf <http://www.technopagan.org/dynamic/>

Um die IP Adresse unter Linux bei Dyn-DNS einzutragen, stehen verschiedene Skripte und Clients zur Verfügung. Die beiden bekanntesten Clients sind *addns.pl* (<http://www.funtaff.com/software/addns.pl/>) und *ddclient* (<http://members.rogers.com/ddclient/pub/ddclient.tar.gz>). Die Konfiguration ist meist sehr einfach und gut erklärt. Daher soll hier nur exemplarisch die Konfigurationsdatei von *addns.pl* vorgestellt werden.

```
# /etc/addns.conf
[main]

{
    use_proxy = no
}

[vpngateway]
{
    detect_method = "iface"
    update_host = vpngateway.dyndns.org
    iface = "ppp0"
    username = "username"
    password = "password"
    server_port = 80
    server_host = members.dyndns.org
}
```

*Listing 9.1 Konfigurationsdatei /etc/addns.conf*

Anschließend muss sichergestellt werden, dass das *addns.pl* Skript bei jeder Einwahl automatisch aufgerufen wird. Die Konfiguration kann auch mit dem Befehl *addns.pl -config* erfolgen.

Nun muss der Partner nur so konfiguriert werden, dass bei dem Aufbau des Tunnels automatisch der Dyn-DNS-Name zur IP Adresse aufgelöst wird.

Bei FreeS/WAN ist diese Auflösung bereits implementiert. Hierzu muss lediglich anstelle der IP Adresse der DNS-Name für *right|left* angegeben werden:

```
conn dyndns
    left=%defaultroute
    leftid=@vpnclient
    lefttrasigkey=0s...
    right=vpngateway.dyndns.org
    rightid=@vpnservers
```

```

rightsubnet=192.168.0.0/24
rightrsasigkey=0s...
auto=start

```

*Listing 9.2 Auszug aus der FreeS/WAN /etc/ipsec.conf Datei*

Bei `racoon` und `isakmpd` ist die DNS-Namensauflösung leider nicht eingebaut. Sie verlangen die Angabe von IP Adressen in ihrer Konfigurationsdatei. Hier soll das Vorgehen am Beispiel von `racoon` verdeutlicht werden. Es ist erforderlich, vor dem Start von `racoon` die Konfigurationsdatei aus einem Template zu erzeugen. Hierzu muss der DNS Name aufgelöst werden und an den entsprechenden Stellen die IP Adresse eingefügt werden.

Ein mögliches Skript sieht folgendermaßen aus:

```

#!/bin/bash
REM_IP=`dig +short vpngateway.dnydns.org | tail -1`
MY_IP=`/sbin/ifconfig eth0 | grep "inet Adresse" | cut -d: -f2 | cut -d' ' -f1`

cat << EOF > ipsec.conf
#!/etc/setkey -f
spdf flush;
spdadd $MY_IP 10.0.2.0/24 any -P out ipsec
        esp/tunnel/$MY_IP-$REM_IP/require;

spdadd 10.0.2.0/24 $MY_IP any -P in ipsec
        esp/tunnel/$REM_IP-$MY_IP/require;
EOF

cat << 2EOF > racoon.conf
...
...
...
2EOF

```

*Listing 9.3 Automatische Erzeugung der Datei /etc/racoon.conf und /etc/ipsec.conf*

Anschließend können die Befehle `setkey` und `racoon` aufgerufen werden.

## 9.2 Advanced Routing

Der Linux Kernel bietet seit der Version 2.2 das sogenannte Advanced oder Policy Routing. Hierbei ist es möglich maximal 65.535 verschiedene Routing Tabellen zu definieren. Die Verwendung dieser Routing Tabellen wird dann durch Regeln gesteuert. So besteht die Möglichkeit in Abhängigkeit eines Quell- oder Zielports unterschiedliche Routen zu benutzen.

Diese Funktionalität wird mit dem Befehl `ip` verwaltet. Die Befehle `arp`, `ifconfig` und `route` existieren nur noch aus Kompatibilitätsgründen. Der Befehl `ip` verwaltet den ARP-Cache, die Netzwerkkarten und IP Adressen, Routen und Regeln.

Eine sehr gute Dokumentation und Einführung in die Befehle `ip` und `tc` (traffic control) ist das Linux Advanced Routing and Traffic Control Howto (<http://lartc.org>).

Hier soll nur auf einen Fall eingegangen werden, der mit diesem Werkzeug gelöst werden kann.

### 9.2.1 Gateway Routing

Das erste Beispiel für den Einsatz von Advanced Routing ist ein klassisches Problem. Wenn ein Tunnel zwischen zwei Netzwerken über VPN Gateways definiert wurde, so können die Rechner aus den Netzwerken sich gegenseitig erreichen. Jedoch ist es nicht möglich ein VPN Gateway zu erreichen oder eine Kommunikation zwischen den VPN Gateways zu erlauben. Hierfür werden weitere Tunnel benötigt, da die VPN Gateways bei der Kommunikation nach außen ihre externe im Tunnel nicht erlaubte IP Adresse verwenden. Mit dem Advanced Routing kann eine Route definiert werden, die für eine Verbindung mit dem anderen VPN Netzwerk die interne IP Adresse verwendet. Hierzu kann die Route mit dem folgenden Befehl definiert werden:

```
ip route add $net dev ipsec0 src $ip
```

Die Variablen `$net` und `$ip` sind durch das andere Netzwerk (zum Beispiel 192.168.1.0/24) und durch die interne IP Adresse (zum Beispiel 192.168.0.254) zu ersetzen. Möglicherweise muss eine vorhandene Route in das entsprechende Netzwerk vorher gelöscht werden. Bei der Verwendung von FreeS/WAN wird sinnvollerweise die Datei `_updown` entsprechend angepasst, da FreeS/WAN selbst die Routen beim Start und Stop eines Tunnels setzt.

Wurde die Route definiert, so können die Gateways sich gegenseitig auf den internen IP Adressen erreichen und auch Verbindungen in die jeweils gegenüberliegenden Netzwerke aufbauen.

## 9.3 Quality of Service

Quality of Service (QoS) bezeichnet die Fähigkeit ausgewählten Netzwerkverkehr besser zu transportieren. Hierfür gibt es verschiedene Technologien und Methoden, die stark von den zu transportierenden Informationen (zum Beispiel Voice-over-IP, VoIP) und den Netzwerkmedien abhängen.

Im Zusammenhang mit VPNs besteht häufig der Bedarf, den VPN-Verkehr insgesamt im Gegensatz zu dem restlichen unverschlüsselten Verkehr zu priorisieren oder zu beschränken.

Dieses Kapitel kann nicht eine ausführliche Einführung in die Thematik und alle Möglichkeiten geben. Es soll lediglich dem Leser als ein erster Start in die QoS-Thematik in Kombination mit IPsec-VPNs dienen. Hierzu wird in diesem Kapitel das Werkzeug `tcng` von Werner Almesberger vorgestellt. Dieses Werkzeug (<http://tcng.sf.net>) bietet die Möglichkeit relativ einfach eine Bandbreitenregulierung durchzuführen. Üblicherweise wird zur QoS Administration der Linux Befehl `tc` verwendet. Dieser Befehl besitzt jedoch eine sehr kryptische und komplizierte Handhabung. Mit dem Werkzeug `tcng` wird eine zusätzliche Abstraktionsebene eingeführt, die die Administration vereinfacht.

### 9.3.1 Installation von `tcng`

Das Paket `tcng` besteht aus zwei Komponenten: `tcc` und `tcsim`. Der Compiler `tcc` wandelt die einfache `tcng`-Sprache in die `tc`-Syntax um. Der `tcsim` Simulator kann diese Konfiguration testen. Für die Übersetzung des `tcsim` ist es erforderlich, dass sowohl der Linux Kernel Quelltext als auch der Quelltext des `iproute2` Paketes auf dem System vorhanden ist. Dann erfolgt die Übersetzung mit:

```
./configure -k kernel-dir -i iproute2-dir
make
make install
```

Wenn der Simulator nicht benötigt wird, kann `tcc` mit dem folgenden Befehl übersetzt werden:

```
./configure --no-tcsim
make tcc
make install
```

Der Autor hält aber auch unter <http://www.spenneberg.org/tc> RPM Pakete bereit.

## 9.3.2 Anwendung von tcng

Um nun den ausgehenden VPN Verkehr zu priorisieren kann mit dem `tc` Befehl ein `tcng` Skript in die `tc` Sprache übersetzt werden. Ein sehr einfaches Skript, das die notwendigen Befehle enthält, ist im Folgenden abgedruckt.

```
/*
 * Ein einfaches tcng-Skript
 *
 * Ralf Spenneberg
 *
 * Priotisiert den VPN-Verkehr
 *
 */

#include "fields.tc"
#include "ports.tc"

#define INTERFACE eth0

dev INTERFACE {
    egress {

        class ( <$vpn> )    if ip_proto == 51 || ip_proto == 50;
        class ( <$ike> )   if udp_sport == 500 || udp_dport == 500 ;
        class ( <$other> ) if 1 ;

        htb () {
            class ( rate 2Mbps, ceil 2Mbps ) {
                $ike = class ( rate 64kbps, ceil 128kbps ) { sfq; } ;
                $vpn = class ( rate 512kbps, ceil 2Mbps ) { sfq; } ;
                $other = class ( rate 128kbps, ceil 2Mbps ) { sfq; } ;
            }
        }
    }
}
```

*Listing 9.4 Dieses tcng-Skript priotisiert den VPN-Verkehr*

Dieses `tcng`-Skript definiert für das Interface `eth0` einen Hierarchical Token-Bucket Filter (HTB). Der HTB ist ein leicht verständlicher und schneller Ersatz für den klassischen Class based Queue Scheduler (CBQ). Der HTB befindet sich ab der Version 2.4.20 im Linux Kernel. Eine Einführung in die Funktion des HTB ist auf der Homepage des Entwicklers Martin Devara zu finden (<http://luxik.cdi.cz/~devik/qos/htb/>).

Hier werden zunächst die verschiedenen Pakete, die über das Interface `eth0` versendet werden sollen, in verschiedene Klassen eingeteilt. Alle ESP und

AH Pakete werden der Klasse `$vpn`, alle IKE-Pakete der Klasse `$ike` und alle anderen Pakete der Klasse `$other` zugewiesen.

Nun wird der HTB mit einer maximalen Datenrate von 2MBit/s erzeugt. Diese Datenrate wird dann unter den drei Klassen aufgeteilt. Dabei erhält die Klasse `$ike` mindestens 64kBit/s, die Klasse `$vpn` mindestens 512kBit/s und die Klasse `$other` mindestens 128kBit/s. Benötigen diese Klassen mehr Bandbreite, so dürfen sie bis zur Angabe `ceil` weitere Bandbreite »leihen«.

Die interne Verwendung eines Stochastic Fair Queueing Moduls (`sfq`) führt zu einer gleichmäßigen Verteilung der Pakete in diesen Klassen.

Das Skript kann nun mit dem Befehl `tcc` übersetzt werden:

```
# tcc -r vpn.tcng > vpn.tc
# cat vpn.tc
tc qdisc del dev eth0 root

# ===== Device eth0
=====

tc qdisc add dev eth0 handle 1:0 root dsmark indices 4 default_index 0
tc qdisc add dev eth0 handle 2:0 parent 1:0 htb
tc class add dev eth0 parent 2:0 classid 2:1 htb rate 250000bps ceil
250000bps
tc class add dev eth0 parent 2:1 classid 2:2 htb rate 8000bps ceil
16000bps
tc qdisc add dev eth0 handle 3:0 parent 2:2 sfq
tc class add dev eth0 parent 2:1 classid 2:3 htb rate 64000bps ceil
250000bps
tc qdisc add dev eth0 handle 4:0 parent 2:3 sfq
tc class add dev eth0 parent 2:1 classid 2:4 htb rate 16000bps ceil
250000bps
tc qdisc add dev eth0 handle 5:0 parent 2:4 sfq
tc filter add dev eth0 parent 2:0 protocol all prio 1 tcindex mask 0x3
shift 0
tc filter add dev eth0 parent 2:0 protocol all prio 1 handle 3 tcindex
classid 2:4
tc filter add dev eth0 parent 2:0 protocol all prio 1 handle 2 tcindex
classid 2:2
tc filter add dev eth0 parent 2:0 protocol all prio 1 handle 1 tcindex
classid 2:3
tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match u8 0x33
0xff at
9 classid 1:1
tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match u8 0x32
0xff at
9 classid 1:1
tc filter add dev eth0 parent 1:0 protocol all prio 1 handle 1:0:0 u32
```

```
divisor 1tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match
u8 0x11 0xff at
9 offset at 0 mask 0f00 shift 6 eat link 1:0:0
tc filter add dev eth0 parent 1:0 protocol all prio 1 handle 1:0:1 u32 ht
1:0:0
match u16 0x1f4 0xffff at 0 classid 1:2
tc filter add dev eth0 parent 1:0 protocol all prio 1 handle 2:0:0 u32
divisor 1tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match
u8 0x11 0xff at
9 offset at 0 mask 0f00 shift 6 eat link 2:0:0
tc filter add dev eth0 parent 1:0 protocol all prio 1 handle 2:0:1 u32 ht
2:0:0
match u16 0x1f4 0xffff at 2 classid 1:2
tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match u32 0x0
0x0 at 0 classid 1:3
```

Dieses Skript kann nun bei der Initialisierung des Netzwerks gestartet werden und die Bandbreitenregelung aktivieren. So ist die Verwendbarkeit des VPNs bei gleichzeitigem Zugriff auf andere Protokolle gewährleistet. Ein Protokoll wie FTP kann nicht die Verbindung komplett auslasten.

**TIPP**

Klassischerweise ist es nur möglich ausgehenden Verkehr mit QoS zu verwalten. Eingehende Pakete können nicht verwaltet werden. Dies versucht das Intermediate Queueing Device (IMQ) zu ändern. Hierbei handelt es sich um ein virtuelles Netzwerkgerät, auf dem anschließend Egress Filter für Ingress Bandbreitenkontrolle definiert werden können. Weitere Informationen sind auf der IMQ Homepage verfügbar (<http://trash.net/~kaber/imq/>).

### 9.3.3 Alternativen

Wenn die Anwendung des `tcng` Paketes zu aufwändig erscheint, so existieren noch einige Alternativen. So liefert die Red Hat Linux Distribution das RPM-Paket `shapecfg` mit. Dieses erlaubt die einfache Implementierung und Anwendung eines CBQ-Filters. Des weiteren existieren einige Init Skripte zur Anwendung von HTB (<http://sourceforge.net/projects/htbinit>) und CBQ (<https://sourceforge.net/projects/cbqinit>). Mit `ktctool` (<http://fr.ee/ktctool/>) und `htbgui` (<http://www.jarod.mpn.pl/htbgui.html>) stehen auch zwei grafische Werkzeuge zur Verfügung, die sich jedoch erst am Anfang ihrer Entwicklung befinden.

## 9.4 Nicht-IP-Tunnel

Das IPsec Protokoll unterstützt keine Routing Protokolle wie das Enhanced Interior Gateway Routing Protokoll (EIGRP) oder Open Shortest Path First (OSPF). Auch Protokolle, die nicht auf IP aufbauen, wie Appletalk und Novells Internetwork Packet Exchange (IPX) können nicht von IPsec transportiert werden. Hierfür muss auf der Basis der IPsec Verbindung ein weiterer Tunnel aufgebaut werden, der diese Protokolle transportieren kann. Dafür kann entweder ein Generic Routing Encapsulation Tunnel (GRE) oder ein Layer Two Tunnel Protocol (L2TP) Tunnel verwendet werden. Der L2TP Tunnel verwendet ein PPP Protokoll im Tunnel. Das Aufsetzen eines GRE Tunnels ist im Vergleich zu einem L2TP Tunnel unter Linux sehr einfach und wird daher zuerst besprochen. Der GRE Tunnel ist auch der Standard-tunnel von Cisco-Geräten.

### 9.4.1 GRE

Die Konfiguration eines GRE Tunnels ist sehr einfach. Hierfür ist lediglich die GRE Unterstützung im Kernel erforderlich. Ob ein Kernel mit GRE Unterstützung eingesetzt wird, kann sehr einfach mit dem Befehl `modprobe ip_gre` kontrolliert werden. Wenn dieser Befehl ohne Fehlermeldung das `ip_gre`-Modul lädt, so ist eine modulare Unterstützung vorhanden. Wurde das Modul fest in den Kernel kompiliert, funktioniert diese Methode jedoch leider nicht. Hier hilft nur ein Test oder ein Blick in die Kernelkonfigurationsdatei.

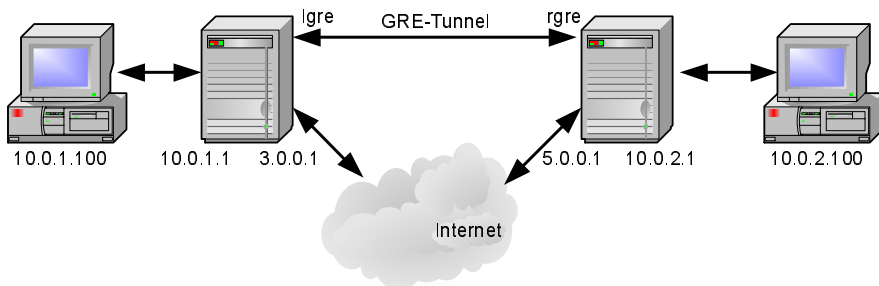


Abbildung 9.1 Ein GRE Tunnel zwischen dem linken und dem rechten Netzwerk

Die Abbildung 9.1 zeigt eine Beispielkonfiguration in der zwei Netzwerke (links und rechts) über ein drittes Netzwerk verbunden sind. Zwischen diesen beiden Netzwerken soll nun ein GRE Tunnel die Kommunikation ermöglichen. Ob die beiden Gateways über einen IPsec Tunnel kommunizieren, ist hierbei unerheblich.

Um nun einen GRE Tunnel auf dem linken Gateway zu starten, müssen die folgende Befehle ausgeführt werden:

```
DEV=lgre
LOCAL_IP=3.0.0.1
REMOTE_IP=5.0.0.1
GRE_IP=172.16.255.1 # Beliebige IP Adresse
REMOTE_NET=172.16.2.0/24
modprobe ip_gre
ip tunnel add $DEV mode gre remote $REMOTE_IP local $LOCAL_IP ttl 255
ip link set $DEV up multicast on
ip addr add GRE_IP dev $DEV
ip route add $REMOTE_NET dev $DEV
```

Auf dem rechten Gateway müssen analog die entsprechenden Befehle ausgeführt werden. Am einfachsten werden auch diese Befehle, wie bei dem Advanced Routing bereits angesprochen, in den Startskripten des VPNs eingebaut.

## 9.4.2 L2TP

Das L2TP Protokoll kann als eine Weiterentwicklung des PPTP-Protokolls verstanden werden. Es bietet von Haus aus keine Verschlüsselung. Jedoch setzen die modernen Microsoft Betriebssysteme (Windows 2000, Windows XP und Windows Server 2003) dieses Protokoll ein, um auf der Basis einer IPsec Verbindung einen sicheren Tunnel aufzubauen. Für ältere Betriebssysteme bietet Microsoft einen MSL2TP/IPsec Client zum kostenlosen Download (<http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/l2tpclient.asp>).

Hier soll die Konfiguration des L2TP-Daemons auf der Linux Seite beschrieben werden. Bevor mit der Konfiguration begonnen wird, sollte eine funktionsfähige IPsec Installation erfolgen. Hierbei ist es sinnvoll, wenn die Linux IPsec Implementierung X.509 Zertifikate unterstützt und im Falle von FreeS/WAN der Delete SA Patch hinzugefügt wurde.

Die Linux IPsec Konfiguration soll hier am Beispiel der FreeS/WAN Konfiguration vorgestellt werden. Um zunächst die Konfiguration und die Fehlersuche zu vereinfachen, ist es sinnvoll, dass zur Authentifizierung PSKs verwendet werden.

Um die in Abbildung 9.2 dargestellte Konfiguration zu erzeugen ist zunächst ein Host Host IPsec Tunnel erforderlich. Das L2TP Protokoll erlaubt dann anschließend den Zugriff auf das interne Netz.

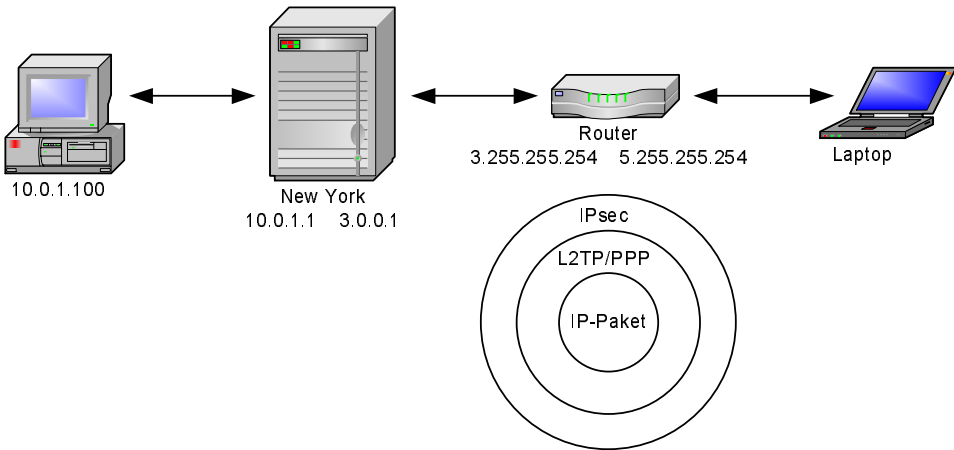


Abbildung 9.2 L2TP Verbindung mit einem Windows Client

Die entsprechende Verbindung in der FreeS/WAN Konfigurationsdatei `/etc/ipsec.conf` sieht folgendermaßen aus:

```
conn L2TP-PSK-WIN
    authby=secret
    pfs=no
    #
    left=3.0.0.1
    leftnexthop=3.255.255.254
    #
    # Der originale Win2k und XP Client
    leftprotoport=17/0
    # MSL2TP, SSH Sentinel, SoftRemote und Win2k/XP mit NAT-T Update
    # leftprotoport=17/1701
    #
    right=%any
    rightprotoport=17/1701
    auto=add
    keyingtries=3
```

Wie in der Verbindung erkannt werden kann, verhalten sich die Windows Clients leicht unterschiedlich in Bezug auf die Einschränkung des Tunnels. Damit der Windows Client erfolgreich den Tunnel aufbaut, muss dieser auf der linken Seite entweder auf das Protokoll 17 (UDP) und einen beliebigen Port oder auf das Protokoll 17 (UDP) und den Port 1701 (L2TP) eingeschränkt werden. Alle original ausgelieferten Windows 2000 und Windows XP Installationen verwenden die erste Variante. Der MSL2TP Client für Windows 9x und Windows NT als auch Windows 2000 und XP Clients mit dem NAT Traversal Update (<http://support.microsoft.com:80/support/kb/articles/>

*q818/0/43.asp*) verwenden die zweite Variante. Um die Interoperabilität mit Windows sicherzustellen ist zusätzlich auch noch die Deaktivierung der Perfect Forward Secrecy auf der Linux Seite erforderlich. Windows unterstützt die PFS per Default nicht, jedoch ist es bei einem MSL2TP Client möglich, PFS über die Registrierung zu aktivieren. Hierzu ist folgender Eintrag erforderlich:

```
[HKEY_LOCAL_MACHINE\Software\IRE\SafeNet\Soft-PK\ACL\1]
"USEPFS"=dword:00000000          (change to dword:00000001)
"P2GROUPDESC"=dword:00000001    (change to dword:00000002)
```

Nachdem nun noch der Preshared Key (PSK) in der Datei `/etc/ipsec.secrets` definiert wurde, kann auf dem Linux System IPsec gestartet werden.

Nun muss der Windows XP Client konfiguriert werden. Hier sollen nur kurz die wesentlichen Schritte aufgeführt werden. Jacco de Leeuw hat auf seiner Homepage eine Einleitung mit Screenshots hinterlegt (<http://www.jacco2.dds.nl/networking/win2000xp-freeswan.html>).

Hierzu wird zunächst unter START->PROGRAMME->ZUBEHÖR->KOMMUNIKATION->NETZWERK- UND DFÜ-VERBINDUNGEN der Wizard gestartet. Dort wird dann NEUE VERBINDUNG ERSTELLEN ausgewählt und anschließend der Knopf WEITER bestätigt. Nach Auswahl der Option VERBINDUNG MIT EINEM PRIVATEN NETZWERK ÜBER DAS INTERNET HERSTELLEN wird diese mit WEITER erneut bestätigt. Nun kann die vorherige Einwahl über eine weitere DFÜ Verbindung aktiviert werden, wenn das System nicht eine dauerhafte Internetverbindung besitzt. Nach der Bestätigung mit WEITER muss die IP Adresse oder der DNS Name des Linux Servers angegeben werden und auch diese IP Adresse mit WEITER bestätigt werden. Nach der Beschränkung auf die eigene Person oder die Freigabe für alle Benutzer muss abschließend noch ein Name für die Verbindung gewählt werden und die Konfiguration wird mit FERTIG STELLEN abgeschlossen. Im folgenden Fenster (siehe Abbildung 9.3) müssen nun die EIGENSCHAFTEN ausgewählt werden. Auf der Registerkarte SICHERHEIT die Option ERWEITERT aktivieren und anschließend die EINSTELLUNGEN auswählen. Hier muss die Verwendung des CHAP Protokolls aktiviert werden. Zusätzlich muss in der Auswahlbox die Datenverschlüsselung auf OPTIONAL gestellt werden. Damit wird die Verschlüsselung im PPP Protokoll deaktiviert. Sie ist nicht erforderlich, da bereits das IPsec Protokoll die Verschlüsselung durchführt. Diese Einstellungen können nun mit OK bestätigt werden. In der Registerkarte SICHERHEIT können nun auch die IPSEC EINSTELLUNGEN konfiguriert werden. Dort wird die Verwendung des Kennwortes (PSK) akti-

viert. Windows 2000 unterstützt nicht die Anmeldung mit PSKs. Hier sind Zertifikate erforderlich. Anschließend wird auf der Registerkarte NETZWERK das L2TP/IPsec Protokoll ausgewählt. Nach Bestätigung mit OK befindet man sich wieder in der in Abbildung 9.3 gezeigten Anmeldebox.



Abbildung 9.3 VPN Anmeldung

Wird nun die Anmeldung mit einem beliebigen Benutzer und Kennwort gestartet, so sollte die IPsec Verbindung bereits erfolgreich aufgebaut werden. Dies kann auf der Linux Seite anhand der Protokollmeldungen verfolgt werden:

```
Pluto[yyy]: "L2TP-PSK-WIN" #1: responding to Main Mode
Pluto[yyy]: "L2TP-PSK-WIN" #1: Peer ID is ID_IPV4_ADDR: 'x.y.z.a'
Pluto[yyy]: "L2TP-PSK-WIN" #1: STATE_MAIN_R3: sent MR3, ISAKMP SA
    established
Pluto[yyy]: "L2TP-PSK-WIN" #1: responding to Quick Mode
Pluto[yyy]: "L2TP-PSK-WIN" #1: STATE_QUICK_R2: IPsec SA established
```

Ist dieser Teil der Verbindung bereits erfolgreich, so kann der L2TP Daemon installiert werden. Hier ist im Moment der `l2tpd` von [www.l2tpd.org](http://www.l2tpd.org) zu empfehlen. Jacco de Leeuw pflegt RPMs (<http://www.jacco2.dds.nl/networking/freeswan-l2tp.html>) für die verschiedenen Distributionen. Die Debian Distribution enthält bereits den `l2tpd`. Nach der Installation kann der `l2tpd` konfiguriert werden.

Das RPM Paket enthält eine funktionsfähige Konfigurationsdatei, die nur leicht angepasst werden muss. Sie ist im Folgenden dargestellt:

```
[global]
; listen-addr = 192.168.1.98

[lns default]
ip range = 192.168.1.128-192.168.1.254
local ip = 192.168.1.99
require chap = yes
refuse pap = yes
require authentication = yes
name = LinuxVPNserver
ppp debug = yes
pppoptfile = /etc/ppp/options.l2tpd
length bit = yes
```

### Listing 9.5 L2TP Daemon Konfigurationsdatei

Dabei haben die einzelnen Parameter die folgenden Bedeutungen: Mit der Angabe `listen-addr` kann der `l2tpd` auf eine bestimmte IP Adresse gebunden werden. Die Angabe `local ip` definiert die lokale Endadresse der L2TP Tunnel. Die Angabe `ip range` definiert die IP Adressen, die dynamisch an die Clients verteilt werden. Mit der Angabe `require chap` wird das CHAP Protokoll aktiviert. Dann müssen die Benutzer und die Kennwörter in der Datei `/etc/ppp/chap-secrets` gespeichert werden. Die Angabe `require authentication` weist den `l2tpd` an, den `pppd` so aufzurufen, dass dieser eine erneute Authentifizierung des Benutzers durchführt. Das `pppoptfile` gibt die Möglichkeit spezifische Optionen für den `pppd` zu setzen.

Nun müssen noch die Benutzer in der Datei `/etc/ppp/chap-secrets` definiert werden. Diese Datei enthält für jeden Benutzer zwei Zeilen mit jeweils vier Spalten:

```
#client  server  kennwort  IP Adresse
ralf     *        "g3h3im"  192.168.1.128/25
*        ralf    "g3h3im"  192.168.1.128/25
```

Nun sollte die Verbindung von der Windowsseite erneut aufgebaut werden und erfolgreich einen Tunnel erstellen können. Wenn anschließend Probleme bei der Nutzung der Verbindung auftreten, so handelt es sich häufig um MTU-Probleme. Sie machen sich bemerkbar, wenn der Zugriff auf kleine Datenmengen und auch ein Ping unproblematisch sind, die Übertragung größerer Datenmengen aber scheinbar hängt. Hier ist es sinnvoll in der Konfigurationsdatei des `pppd` (`/etc/ppp/options.l2tpd`) die Zeilen `mtu 1400` und `mrui 1400` einzutragen. Damit wird die MTU entsprechend verkleinert. Bleibt das Problem bestehen, so kann mit noch kleineren Werten experimentiert werden.

## 9.5 NAT Traversal

Zum Zeitpunkt der Drucklegung ist lediglich bei FreeS/WAN die Unterstützung für NAT Traversal implementiert.

Die Unterstützung des NAT Traversal bei FreeS/WAN verlangt einen zusätzlichen Patch. Dieser Patch ist in dem Patch SuperFreeS/WAN enthalten oder kann auch einzeln angewendet werden. Die Anwendung des Patches ist bei der Installation von FreeS/WAN beschrieben.

Nach der Installation des Patches ist die Anwendung von NAT Traversal sehr einfach. Die Unterstützung ist vollkommen transparent und kann auch aktiviert werden, wenn die Gegenseite NAT Traversal nicht unterstützt. Dann kann natürlich auch kein NAT Traversal durchgeführt werden. Eine Aktivierung führt jedoch nicht zu Fehlermeldungen.

Um das NAT Traversal einzuschalten, muss der Parameter `nat_traversal` in der Konfigurationsdatei gesetzt werden.

```
nat_traversal=yes
```

### *Listing 9.6 Aktivierung des NAT Traversal*

Der NAT Traversal Patch deaktiviert den Transport Modus. Im Tunnel Modus kommt das Problem hinzu, dass nicht jede beliebige IP Adresse im Tunnel akzeptiert werden darf. Die IP Adresse, die im Tunnel verwendet wird, muss daher in der Konfigurationsdatei spezifiziert werden. Alternativ kann auch DHCP-over-IPsec (siehe nächstes Kapitel) eingesetzt werden.

Die Definition der IP Adresse erfolgt entweder mit der Direktive `rightsubnet` oder `rightsubnetwithin`.<sup>2</sup>

```
rightsubnet=192.168.0.0/16
```

Mit diesen Parametern können die IP Adressen explizit angegeben werden. Außerdem besteht die Möglichkeit, die IP Adressen virtuell zu verwalten. Hierzu existieren die Werte `vhost` und `vnet` für den Parameter `left|rightsubnet`. Diesen kann einer der folgenden Werte zugewiesen werden:

- **%no** Akzeptiere auch die öffentliche IP im Tunnel.
- **%dhcp** Akzeptiere die DHCP SA (nicht implementiert).
- **%ike** Akzeptiere die IKE Config Mode IP (nicht implementiert).
- **%priv** Akzeptiere jede IP Adresse aus der systemweiten Liste privater IP Adressen (siehe unten).

2. `rightsubnetwithin` ist nur verfügbar, wenn auch der X.509-Patch angewendet wurde.

- **%v4:x** Akzeptiere die IPv4-Adressen aus der Liste x.
- **%v6:x** Akzeptiere die IPv6-Adressen aus der Liste x.
- **%a11** Akzeptiere jede beliebige IP Adresse (zu Testzwecken, unsicher)

Um sowohl Clients, die kein NAT Traversal benötigen, als auch Clients aus privaten Netzwerken (RFC 1918) zu unterstützen wird die folgende Zeile verwendet:

```
rightsubnet=vhost:%no,%priv
```

Die Angabe `%priv` bezieht sich hierbei auf eine systemweite Liste privater IP Adressen. Diese Liste sollte in dem `config setup` Bereich der Konfigurationsdatei definiert werden. Diese Liste sollte alle RFC 1918 Netzwerke aus dem eigenen enthalten:

```
virtual_private=%v4:10.0.0.0/8,%v4:172.16.0.0/12,%v4:  
192.168.0.0/16,%v4:!192.168.0.0/24
```

## 9.6 DHCP-over-IPsec

In vielen Fällen, bei denen ein entfernter Rechner über ein VPN mit einem LAN in Verbindung tritt, ist es von Vorteil, wenn anschließend der entfernte Rechner eine IP Adresse aus dem internen Netzwerk erhält und sich scheinbar in dem LAN befindet. Dies kann erreicht werden, indem der Client eine virtuelle IP Adresse per DHCP erhält.

Die Verwendung von DHCP bietet folgende Vorteile:

- Die Verwendung von DHCP erleichtert den Einsatz und die Integration in große Netze, da diese meist sowieso bereits mit Hilfe des DHCP Protokolls verwaltet werden.
- Zusätzlich bietet das DHCP Protokoll die Möglichkeit den Adress Pool anhand bestimmter Eigenschaften des Clients zu verwalten. So können bestimmten Clients bestimmte IP Adressen zugewiesen werden.
- Die Speicherung der DHCP Daten und der DHCP Datenbank auf dem DHCP Server erleichtert die Konfiguration einer Hochverfügbarkeitslösung, da diese Informationen nicht über die VPN Gateways verteilt werden muss.
- Das DHCP Protokoll verfügt bereits über Verfahren zur Neukonfiguration der IP Adressen. Diese Funktion muss daher nicht in dem IKE Protokoll implementiert werden.

- Sämtliche DHCP Funktionen können ohne weitere Änderung des IKE-Protokolls genutzt werden. Dadurch sind keine Einschränkungen der Sicherheit oder eine zusätzliche Komplexität zu erwarten. Alternativ wird von einigen Herstellern (insbesondere Microsoft) das L2TP Protokoll eingesetzt. Es baut einen Tunnel auf der vorhandenen IPsec Verbindung auf, der dann andere IP Adressen verwenden kann (siehe Abschnitt 9.4.2, »L2TP«).

Die Abbildung 9.4 zeigt eine typische Anwendung. Hierbei baut der externe Client einen IPsec Tunnel auf. Durch den IPsec Tunnel verbindet er sich mit einer virtuellen IP Adresse, die er zuvor von dem DHCP Relay auf dem VPN Gateway erhalten hat. Anschließend befindet er sich scheinbar im internen Netzwerk. Hierzu benötigt der externe Rechner zwei IP Adressen und üblicherweise zwei Netzwerkkarten. Eine physikalische Netzwerkkarte mit echter IP Adresse wird verwendet, um den Tunnel zum VPN Gateway aufzubauen. Diese IP Adresse wird auch als IP Adresse im äußeren IP Header verwendet. Die zweite virtuelle Netzwerkkarte verwendet die virtuelle IP Adresse, die über DHCP zugeteilt wurde. Diese IP Adresse wird im inneren, verschlüsselten IP Header des Tunnels verwendet.

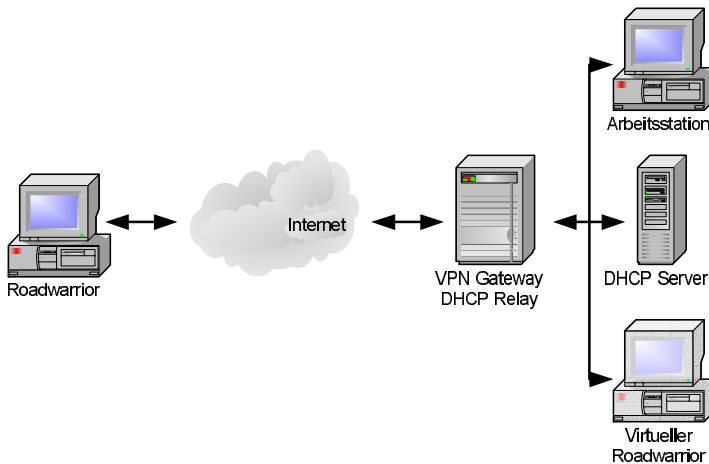


Abbildung 9.4 Der Client fordert zunächst per DHCP eine IP Adresse an, um später mit dieser Adresse über den Tunnel zu kommunizieren

#### ACHTUNG

FreeS/WAN ist nur nach Anwendung des X.509 Patches in der Lage diese Funktion zu unterstützen.

## 9.6.1 Installation und Konfiguration des DHCP-Relays

Üblicherweise wird das VPN Gateway nicht gleichzeitig auch den DHCP Server beherbergen. Der DHCP Server befindet sich meist auf einem anderen Rechner und versorgt auch das restliche interne Netzwerk mit der Konfiguration der IP Adressen. Aus diesem Grund muss auf dem VPN Gateway ein DHCP Relay installiert werden, das die Anfragen des Clients entgegennimmt und an den DHCP Server weiterreicht. Mario Strasser hat einen derartigen DHCP Relay geschrieben und unter <http://www.strongsec.com/freeswan/dhcrelay/index.htm> zur Verfügung gestellt. Unter dieser URL findet der Leser auch ein DHCP Relay Howto, welches die Installation und Konfiguration beschreibt.

Die Software `dhcrelay-0.3.1` befindet sich als Quelltext und als RPM Paket auf der CD.

Die Installation der Software aus den Quellen ist sehr einfach und beschränkt sich auf den Aufruf der folgenden Befehle:

```
# cd /usr/local/src
# tar xvzf /<path>/dhcrelay-<version>.tar.gz
# cd dhcrelay-<version>
# ./configure
# make
# make install
```

Anschließend kann das DHCP Relay mit dem Skript `/etc/init.d/dhcrelay` gestartet und gestoppt werden. Hierbei ist es erforderlich, FreeS/WAN vor dem DHCP Relay zu starten. Nach einem Neustart von FreeS/WAN muss auch das DHCP Relay neugestartet werden, damit es sich neu an die VPN Netzwerkkarten `ipsecX` binden kann.

### ACHTUNG

Es ist sinnvoll, den Start des DHCP Relays in dem FreeS/WAN Startskript aufzunehmen und hier auch dafür zu sorgen, dass bei einem Neustart von FreeS/WAN auch das DHCP-Relay neugestartet wird.

Damit das DHCP-Relay automatisch nach einem Neustart des Rechners gestartet wird, kann das Skript mit den Befehlen `insserv` (SuSE), `update-rc.d` (Debian) oder `chkconfig` (RedHat) dauerhaft aktiviert werden.

Nun muss die Konfiguration des DHCP-Relay angepasst werden. Die Konfigurationsdatei `(/usr/local)/etc/dhcrelay.conf` ist erfreulicherweise sehr übersichtlich.

```
# DHCP-Relay configuration file
# $Id: dhcprelay.conf,v 1.1.1.1 2002/08/20 08:42:03 sri Exp $

# Logfile
LOGFILE="/var/log/dhcprelay.log"

# IPsec devices (comma separated list including NO spaces)
DEVICES="ipsec0"

# Device over which the DHCP-Server can be reached
SERVERDEVICE="eth0"

# Hostname or IP Address of the DHCP-Server
DHCPSEVER="192.168.7.5"
```

*Listing 9.7 Die Konfigurationsdatei /etc/dhcprelay.conf gibt den DHCP Server an*

Der Eintrag `DEVICES` definiert die Netzwerkkarten, auf denen das DHCP Relay DHCP Anfragen entgegennehmen soll. Der Eintrag `SERVERDEVICE` definiert die Netzwerkkarte, über die das DHCP Relay den DHCP Server (`DHCPSEVER`) erreicht.

#### TIPP

Wenn aus administrativen Gründen doch der DHCP Server auch auf dem VPN Gateway installiert werden soll, so kann die Kommunikation zwischen dem DHCP Relay und dem DHCP Server über das Loopback Interface erfolgen. Hierzu ist jedoch die Konfiguration des DHCP Servers ebenfalls anzupassen.

Der DHCP Server muss nun auch IP Adressen an das DHCP Relay ausgeben. Hierbei kann es sich um dieselben IP Adressen handeln, die er auch an normale interne Clients verteilt, oder um einen eigenen Adresspool. Das DHCP Relay erlaubt eine Unterscheidung dieser Anfragen, da es bei jeder Anfrage die Option `agent.circuit-id` setzt. Dieser Wert enthält anschließend den Namen der Netzwerkkarte, über die das DHCP Relay die Anfrage erhielt (meist `ipsec0`).

Eine typische DHCP Konfigurationsdatei für den Internet Software Consortium (ISC) DHCP Server ist im Folgenden abgebildet:

```
# Konfigurationsdatei für dhcpd-3.x
ddns-update-style none;
```

```
# Zugriff über DHCP-Relay
class "vpn-clients" {
    match if option agent.circuit-id = "ipsec0";
}

# Von dem DHCP-Server zu verwaltendes Netzwerk
subnet 192.168.0.0 netmask 255.255.254.0 {
    option domain-name "spenneberg.com";
    option domain-name-servers 192.168.0.15, 217.160.128.61;
    option routers 192.168.0.254;

    # interne clients
    pool {
        deny members of "vpn-clients";
        range 192.168.0.20 192.168.0.253;
        default-lease-time 72000;
        max-lease-time 144000;
    }
    # vpn clients
    pool {
        allow members of "vpn-clients";
        range 192.168.1.1 192.168.1.50;
        default-lease-time 3600;
        max-lease-time 7200;
    }
}
```

Wenn sowohl die VPN Clients als auch die internen Clients dasselbe Netzwerk verwenden, ist es erforderlich auf dem VPN Gateway Proxy ARP zu aktivieren. Ist dies nicht der Fall, so arbeitet das VPN Gateway als Router und leitet nur Pakete weiter, die speziell an das Gateway geschickt werden. Da jedoch sowohl die VPN Clients als auch die internen Clients davon ausgehen, dass sie sich im selben lokalen Netzwerk befinden, versuchen sie die Pakete direkt und nicht über einen Router zuzustellen. Durch die Aktivierung von Proxy ARP wird aus dem VPN Gateway praktisch eine Bridge, die alle Pakete weiterleitet.

```
# sysctl -w net.ipv4.conf.eth0.proxy_arp=1
```

#### *Listing 9.8 Aktivierung von Proxy ARP*

## 9.6.2 Konfiguration des VPN-Gateways

Auf dem VPN Gateway muss nun die Konfiguration von FreeS/WAN angepasst werden, so dass zunächst ein Tunnel für die DHCP Anfrage aufgebaut werden kann. Die folgenden Zeilen erlauben einen derartigen Tunnel:

```
config setup
    interfaces          = %defaultroute
    klipsdebug          = none
    plutodebug         = none
    plutoload          = %search
    plutostart        = %search
    uniqueids          = yes

conn %default
    authby              = rsasig
    left                = %defaultroute
    leftcert            = certs/newyork_cert.pem
    leftid              = \}/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
    OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net\{
    leftsubnet         = 192.168.0.0/23
    right               = %any
    rightrsasigkey     = %cert
    auto                = add

# VPN-Gateway ist left, Roadwarrior ist right
conn dhcp
    rekey               = no
    keylife             = 30s
    rekeymargin        = 15s
    leftsubnet         = 0.0.0.0/0
    leftprotoport      = udp/bootps
    rightprotoport     = udp/bootpc

conn roadwarrior
    rightsubnetwithin  = 192.168.1.0/24
```

*Listing 9.9 Konfiguration VPN Gateway Teil 1*

In der angegebenen Konfiguration muss als `leftsubnet` der Wert `0.0.0.0/0` eingetragen werden, da der Client die DHCP Anfrage an die Broadcast Adresse sendet. Der Tunnel wird nur für wenige Sekunden benötigt und muss lediglich UDP Pakete zwischen den Ports `bootpc` und `bootps` erlauben.

Anschließend muss das VPN Gateway einen Tunnel erlauben, der die Verwendung der dynamischen DHCP IP Adresse zulässt. Dies kann mit der Direktive `rightsubnetwithin` erreicht werden. Diese Funktion steht ähnlich dem Protokollselektor und Portselektor nur nach Anwendung des X.509 Patches für FreeS/WAN zur Verfügung.

**ACHTUNG**

Bei einigen Clients kommt es bei dieser Konfiguration zu Problemen. Diese Clients (zum Beispiel SSH Sentinel) verwenden den aufgebauten VPN-Kanal auch um ihre IP Adresse bei dem DHCP Server zu erneuern. Diese Clients bauen hierfür nicht einen neuen DHCP Tunnel auf. Das führt zu Problemen, da die Anfragen unter gewissen Umständen an Broadcast Adressen gesendet werden. In diesem Fall ist der Roadwarrior folgendermaßen zu konfigurieren:

```
conn roadwarrior
    leftsubnet      = 0.0.0.0/0
    rightsubnetwithin = 192.168.1.0/24
```

*Listing 9.10 Abweichende Roadwariorkonfiguration bei Einsatz von SSH Sentinel*

### 9.6.3 Konfiguration des VPN-Clients

Der gebräuchlichste Client für DHCP-over-IPsec ist SSH Sentinel. SSH Sentinel (<http://www.ssh.com>) ist ein kommerzieller Client und kostet etwa 180 Euro.

Die Konfiguration des SSH Sentinel wird im Kapitel 7, »Aufbau heterogener Virtueller Privater Netze« beschrieben. Die Abbildungen 9.5 und 9.6 zeigen daher nur die Aktivierung der Funktion DHCP-over-IPsec im SSH Sentinel.

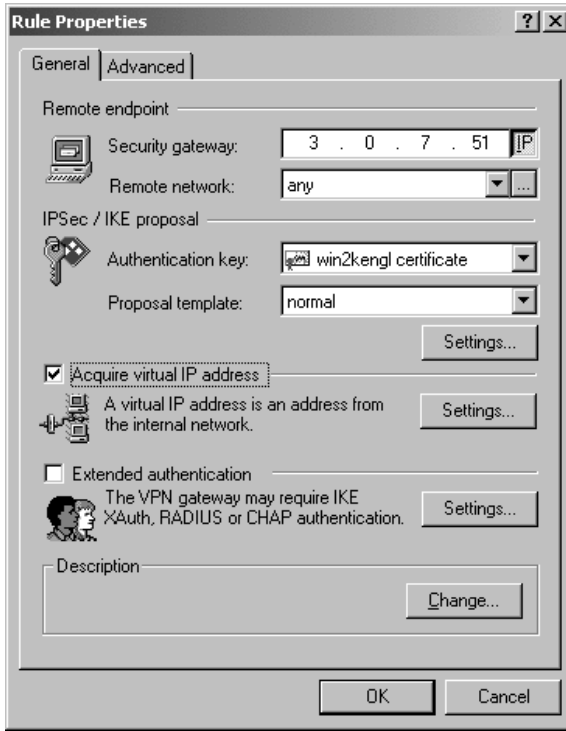


Abbildung 9.5 Aktivierung der virtuellen Adresse im SSH Sentinel

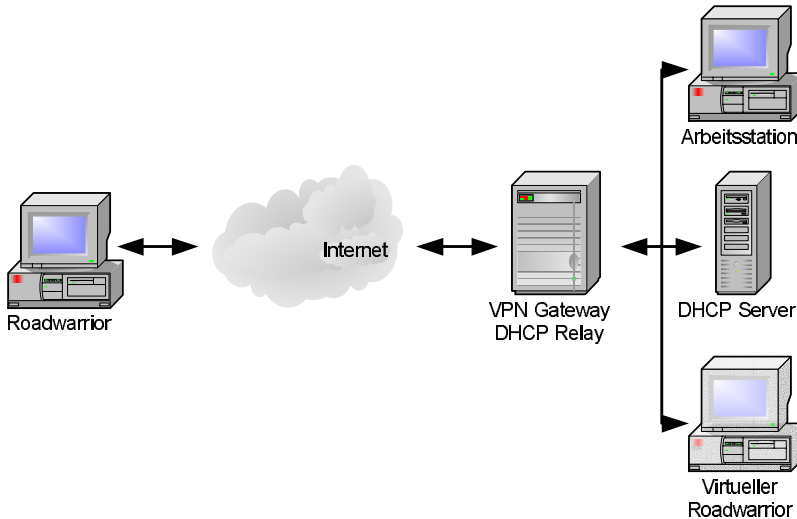


Abbildung 9.6 Auswahl des DHCP-over-IPsec Protokolls

## 9.7 Opportunistische Verschlüsselung

Das Ziel des FreeS/WAN-Projektes ist die Sicherung des Internets durch eine Verschlüsselung des Netzwerkverkehrs. Die Verwendung eines VPNs erfordert aber immer die vorherige Konfiguration durch den Administrator. Hierzu muss der Administrator auf allen Systemen die entsprechende Software installieren und die Schlüssel der Systeme untereinander austauschen. Anschließend können die Systeme die vordefinierten Tunnel nutzen. Die opportunistische Verschlüsselung ermöglicht es den beteiligten Systemen, selbst herauszufinden, ob eine verschlüsselte Übertragung der Daten möglich ist. Die hierfür erforderlichen Schlüssel müssen lediglich im DNS Server hinterlegt werden.

Die Verwendung des DNS Servers als zentraler Speicher für die öffentlichen Authentifizierungsschlüssel ist sowohl der größte Vor- als auch Nachteil der opportunistischen Verschlüsselung. Der DNS Dienst ist eine weltweit verfügbare offene Lösung zur Verteilung von Software. Leider sind aber die dort gespeicherten Informationen nur so sicher wie der Dienst und das verwendete Protokoll selbst. Da DNS in erster Linie das UDP-Protokoll verwendet, ist vor der flächendeckenden Einführung von DNSSEC, einer zusätzlichen Sicherheitsebene, die Anwendung der opportunistischen Verschlüsselung mit Vorsicht zu planen. Ein Angriff auf die Verschlüsselung selbst ist zwar nicht möglich, aber ein Angreifer kann als Man-in-the-Middle den Verkehr abhören.

Ein weiteres Problem bei der opportunistischen Verschlüsselung ist die Tatsache, dass alle Rechner, die nun einen Schlüssel im DNS hinterlegen als vertrauenswürdig eingestuft werden. Zusätzliche Firewallregeln müssen die wirklichen Freunde von den Feinden unterscheiden.

Schließlich existiert noch die Gefahr eines Denial-of-Service bei der Verwendung der opportunistischen Verschlüsselung.

### ACHTUNG

Mit der FreeS/WAN Version 2.01 ändert sich die Konfiguration und die Funktionsweise der opportunistischen Verschlüsselung. Während ältere Versionen sowohl TXT als auch KEY Ressource Records (RR) für die Speicherung der Schlüssel nutzten, verwendet die Version 2.01 nur noch TXT RRs.

### 9.7.1 Funktionsweise

Die opportunistische Verschlüsselung erlaubt die Verwendung von IPsec, ohne dass der Administrator zuvor die entsprechenden Verbindungen konfiguriert hat. Damit dies möglich ist, muss FreeS/WAN jedes nach außen gerichtete Paket abfangen und anschließend ermitteln, ob es möglich ist das Paket über eine verschlüsselte Verbindung zuzustellen. Dabei unterscheidet FreeS/WAN zwei Rollen: den Initiator, der den Tunnel aufbaut, und den Responder.

Sobald der Initiator ein Paket an den Responder versenden möchte, fängt FreeS/WAN dieses Paket ab und stellt eine Reverse DNS Anfrage für die Ziel IP Adresse. Der Responder hat zuvor seinen öffentlichen Schlüssel im DNS veröffentlicht. Erhält der Initiator in der Antwort auf seine DNS Anfrage einen öffentlichen Schlüssel, so startet er die IKE Verhandlungen der Phase 1 mit dem Responder. Sobald die Verschlüsselung verhandelt wurde überträgt der Initiator seine Identifikation (zum Beispiel FQDN) an den Responder. Der Responder ermittelt mit dieser Information im DNS den öffentlichen Schlüssel des Initiators und kann mit diesem Schlüssel den Initiator authentifizieren.

Aus diesem Protokoll ergeben sich drei verschiedene Rollen, die ein Rechner einnehmen kann. Diese werden im folgenden erklärt und ihre Konfiguration beschrieben.

### 9.7.2 OE-Initiator

Am einfachsten ist die Konfiguration eines reinen OE-Initiators. Dieser Rechner kann lediglich opportunistische Verbindungen initiieren, aber nicht passiv derartige Verbindungen entgegennehmen.

Um erfolgreich einen OE-Initiator aufbauen zu können, muss der Administrator über einen DNS-Eintrag verfügen können. Dieser Eintrag muss nicht unbedingt auf die IP Adresse des Systems verweisen, daher können auch verschiedene freie DNS-Anbieter wie <http://soa.granitecanyon.com> oder <http://www.fdns.net> genutzt werden.

Nach der Wahl eines geeigneten FQDN, dessen DNS-Eintrag geändert werden kann, wird mit dem folgenden Befehl dieser Eintrag generiert:

```
# ipsec showhostkey --txt @vpn.spenneberg.com
; RSA 2192 bits  vpn.spenneberg.com  Fri Jul 25 14:01:52 2003
      IN      TXT      "X-IPsec-Server(10)=@vpn.spenneberg.com"  "
AQNv7irLLViBZRKVUAnHrLwTMsvBeYnV52eCsdhUiTvgc6+17MzZbAHH+1B1lFX8T1K0Bs1
```

```
jBRRF1dd85g93eUEXARuJOi2LLo1X5JwGabLYtju+fhyks358rvuzmDX+V/PUDgvnWq96COWU
BM7119wv1gbtOvoHI2eDyUYZmhh7N3uKafArmEnAIGdJxe5/FEF9h6ZhDn0a7bV3C/0571rCrn
OhV9c56LXGDIAnwS7g1QY/zPkzG4o+UVJcF/" "PwsPCJSY+m3iqfLXJXSkEvBL5+m9kPQLmy
CA7ezIvnPVp+x6ptQ7V1f50DbJQvmZ9uSS6/Bbq4nEMd5/U5d/crrbni+OkZel2UGtcFvZtV
vmyoj+1J"
```

*Listing 9.11 Erzeugung des Forward DNS Eintrags*

## ACHTUNG

Momentan verwendet FreeS/WAN noch einen TXT RR. Das FreeS/WAN-Team versucht jedoch einen neuen IPSECKEY-RR in den DNS Standard aufnehmen zu lassen. Dieser wird dann den TXT RR ablösen.

Dieser Eintrag muss nun im DNS Server für die entsprechende Domäne veröffentlicht werden. Das kann dann wie folgt aussehen:

```
$TTL      86400
$ORIGIN  spenneberg.com.
@         1D IN SOA      ns1.spenneberg.net.
ralf.spenneberg.net
(
                                2002080201      ; serial (d.
                                adams)
                                3H              ; refresh
                                15M             ; retry
                                1W              ; expiry
                                1D )           ; minimum

                                1D IN NS      ns1.spenneberg.net.
                                1D IN MX      5      mail
vpn      1D IN A        217.160.128.61
                                10D IN TXT
"X-IPsec-Server(10)=@vpn.spenneberg.com" " AQNv7irLLViBZRKVUANHrLwTmsvBe
YnV52eCsdhUiTvgc6+17MZzbAHH+1B11FX8T1K0Bs1jBRRF1dd85g93eUEXARuJOi2LLo1X
5JwGabLYtju+fhyks358rvuzmDX+V/PUDgvnWq96COWUBM7119wv1gbtOvoHI2eDyUYZmhh7N3
uKafArmEnAIGdJxe5/FEF9h6ZhDn0a7bV3C/0571rCrnOhV9c56LXGDIAnwS7g1QY/zPkzG4o+
UVJcF/" "PwsPCJSY+m3iqfLXJXSkEvBL5+m9kPQLmyCA7ezIvnPVp+x6ptQ7V1f50DbJQ
vmZ9uSS6/Bbq4nEMd5/U5d/crrbni+OkZel2UGtcFvZtVvmyoj+1J"
```

*Listing 9.12 Zonendatei mit Schlüssel*

Eine erfolgreiche Veröffentlichung des Schlüssels im DNS kann mit dem Befehl `ipsec verify -host vpn.spenneberg.com` überprüft werden. Der Befehl meldet unter anderem:

```
Looking for TXT in forward map: vpn.spenneberg.com      [OK]
```

Nun muss die FreeS/WAN Konfigurationsdatei noch leicht angepasst werden, in dem die folgende Verbindung hinzugefügt wird:

```
conn my-private-or-clear
    leftid=@vpn.spenneberg.com
    also=private-or-clear
```

Zusätzlich muss eine Gruppendatei für diese neue IPsec Policy angelegt werden:

```
cp -p /etc/ipsec.d/policies/private-or-clear
    /etc/ipsec.d/policies/my-private-or-clear
```

Die neue Datei sollte den Eintrag 0.0.0.0/0 enthalten. Dieser Eintrag sollte aus der Datei `private-or-clear` entfernt werden. So versucht FreeS/WAN nun bei jeder Verbindung mit jedem Rechner (0.0.0.0/0) zunächst eine verschlüsselte Verbindung aufzubauen. Ist dies nicht möglich, so erlaubt FreeS/WAN eine Klartextverbindung.

### 9.7.3 Volle opportunistische Verschlüsselung

Wenn ein System sowohl als Initiator als auch als Responder auftreten kann, so spricht die FreeS/WAN Dokumentation von voller opportunistischer Verschlüsselung (Full OE). Hierzu ist es erforderlich, dass sowohl bei dem DNS Namen als auch bei der IP Adresse des Systems ein TXT RR mit dem öffentlichen Schlüssel des Systems eingetragen wird. Dabei ist es notwendig, dass der verwendete DNS Name bei einer DNS Auflösung tatsächlich die echte IP Adresse des Systems liefert.

Der Eintrag dieses Schlüssels ist im letzten Abschnitt bereits besprochen worden. Daher soll hier nur die Generierung des Schlüssels für den Reverse Lookup beschrieben werden.

Dazu wird der folgende Befehl verwendet:

```
# ipsec showhostkey --txt 217.160.128.61
; RSA 2192 bits    vpn.spenneberg.com    Fri Jul 25 14:01:52 2003
      IN          TXT          "X-IPsec-Server(10)=217.160.128.61"  "
AQNv7irLLViBZRKVUAnHrLwTMsVBeYnV52eCsdhUiTvgc6+17MzZbAHH+1B1lFX8T1K0Bs1jBR
RF1dd85g93eUEXARuJOi2LLo1X5JwGabLYtju+fhyks358rvuzmDX+V/PUDgVnWq96COWUBM7
119wv1gbtOvoHI2eDyUYZmhh7N3uKafArmEnAIGdJxe5/FEF9h6ZhDn0a7bV3C/0571rCrnOh
V9c56LXGDIAwS7g1QY/zPkg4o+UVJcF/"  "PwsPCJSY+m3iqfLXJXSkEvBL5+m9kPQLmyCA7
ezIvnPVp+x6ptQ7V1f50DbJQvmZ9uSS6/Bbq4nEMd5/U5d/crrbni+OkZel2UGtcFvZtVvmyo
j+1J"
```

*Listing 9.13 Erzeugung des Reverse DNS Eintrags*

Dieser Schlüssel muss nun unter dem Eintrag `61.128.160.217.in-addr.arpa` in der Reverse DNS Zone veröffentlicht werden. Die erfolgreiche Veröffentlichung im DNS kann mit dem Befehl `ipsec verify -host vpn.spenneberg.com` überprüft werden. Dabei sollte nun auch die Zeile `Looking for TXT in reverse map` ein OK ausgeben.

Eine weitere Konfiguration wie bei dem OE Initiator ist nicht erforderlich. Die Konfiguration nach der Installation genügt. Die Policy-Gruppe `private-or-clear` ist bereits vollständig konfiguriert.

## 9.7.4 Gateway OE

Schließlich besteht noch die Möglichkeit, ein Gateway zu konfigurieren, das für dahinter liegende Subnetze die opportunistische Verschlüsselung ermöglicht. Dabei fängt der Initiator ein Paket ab, das an einen Endknoten gerichtet ist, der selbst nicht in der Lage ist einen Tunnel aufzubauen. Jedoch befindet sich zwischen dem Initiator und dem Endknoten ein Gateway, das diese Aufgabe wahrnehmen und die Verbindung zwischen dem Initiator und dem Gateway mit dem IPsec Protokoll sichern kann. Der Initiator baut dann die Verbindung zum Gateway auf.

Hierfür ist es zunächst erforderlich, dass dieses Gateway als Full OE System konfiguriert wurde. Dies wurde im letzten Abschnitt beschrieben.

Nun besteht aber noch zusätzlich das Problem, dass der Initiator herausfinden muss, dass es ein Gateway gibt, welches den Endknoten schützt und welche IP Adresse und welchen Schlüssel dieses Gateway verwendet. Hierfür werden für jeden Endknoten zusätzliche Reverse DNS Einträge benötigt. Diese Einträge enthalten die IP Adresse des Gateways und den öffentlichen Schlüssel des Gateways. Dazu wird der Eintrag zunächst wie in Listing 9.13 auf dem Gateway erzeugt. Anschließend wird dieser TXT RR bei jedem Reverse DNS Eintrag von jedem zu schützenden Endknoten eingetragen:

```
77.1.0.3.in-addr.arpa. IN PTR eins.example.com.
; RSA 2192 bits  vpn.spenneberg.com  Fri Jul 25 14:01:52 2003
      IN  TXT      "X-IPsec-Server(10)=217.160.128.61" " AQNv7.../"

78.1.0.3.in-addr.arpa. IN PTR zwei.example.com.
; RSA 2192 bits  vpn.spenneberg.com  Fri Jul 25 14:01:52 2003
      IN  TXT      "X-IPsec-Server(10)=217.160.128.61" " AQNv7.../"

79.1.0.3.in-addr.arpa. IN PTR drei.example.com.
; RSA 2192 bits  vpn.spenneberg.com  Fri Jul 25 14:01:52 2003
      IN  TXT      "X-IPsec-Server(10)=217.160.128.61" " AQNv7.../"
```

Auch hier ist keine weitere Konfiguration erforderlich.

## 9.7.5 Test der opportunistischen Verschlüsselung

Das FreeS/WAN Team stellt zwei Server zur Verfügung, die für einen Test der opportunistischen Verschlüsselung genutzt werden können. Hierzu genügt es mit einem Web Browser vom Initiator eine Verbindung nach <http://oetest.freeswan.org> oder <http://oetest.freeswan.nl> aufzubauen. Wenn die opportunistische Verschlüsselung funktioniert, so erscheint der folgende Text:

```
You seem to be connecting from: 217.160.128.61 which DNS says is:
vpn.spenneberg.com
```

```
Status E-route
OE   enabled   16   192.139.46.73/32   ->   217.160.128.61/32   =>
tun0x2097@217.160.128.61
OE   enabled   176  192.139.46.77/32   ->   217.160.128.61/32   =>
tun0x208a@217.160.128.61
```

*Listing 9.14 Erfolgreiche opportunistische Verschlüsselung*

Beim Aufbau und dem Test der opportunistischen Verbindungen ist es wichtig, dass kompatible FreeS/WAN Versionen eingesetzt werden. Bisher unterstützen lediglich FreeS/WAN Versionen ab 1.91 diese Funktion. Jedoch wurde mit der Version 2.01 eine Änderung des Protokolls eingeführt. Dadurch können ältere Versionen nur noch als Initiator eine Verbindung zu FreeS/WAN 2.01 aufbauen. Ein Aufbau in umgekehrter Richtung ist nicht möglich.

Da erst sehr wenige Erfahrungen mit der opportunistischen Verschlüsselung von FreeS/WAN 2.01 gemacht wurden, sei hier nur auf die OE-Troubleshooting Tipps des FreeS/WAN Projektes verwiesen, die immer erweitert werden. Sie sind unter [http://www.freeswan.org/freeswan\\_trees/freeswan-2.01/doc/quickstart.html#oe.trouble](http://www.freeswan.org/freeswan_trees/freeswan-2.01/doc/quickstart.html#oe.trouble) zu finden.

## 9.7.6 Policy-Gruppen

FreeS/WAN 2.0 hat mit den Policy Groups ein neues Konfigurationswerkzeug eingeführt. Hierbei handelt es sich um die Möglichkeit OE Verbindungen sehr einfach mit geringem Aufwand zu konfigurieren. Die Policy Gruppen funktionieren im Moment nur in Kombination mit der opportunistischen Verschlüsselung.

FreeS/WAN enthält bereits die folgenden eingebauten Policy Gruppen: `private`, `private-or-clear`, `clear-or-private`, `clear` und `block`.

Um diese Gruppen zu nutzen, ist es lediglich erforderlich, die entsprechenden IP Adressen oder IP Netzwerke in CIDR Notation in den Gruppdateien `/etc/ipsec.d/policies/*` einzutragen. Anschließend werden diese Dateien von FreeS/WAN durch den Aufruf von `ipsec auto -rereadgroups` wieder eingelesen. Hier ein Beispiel:

```
[root@vpn root]# cd /etc/ipsec.d/policies
[root@vpn policies]# cat private
3.0.2.18/32          # Mein POP3 Server
3.0.2.19/32          # Mein Web Server

[root@vpn policies]# cat private-or-clear
0.0.0.0/0           # Meine Default Regel

[root@vpn policies]# cat clear
3.0.2.81/32         # Mein Web Proxy

[root@vpn policies]# cat block
www.badhost.com
```

Die eingebauten Policy-Gruppen schützen lediglich den Rechner selbst. Sie sind nicht in der Lage, Subnetze, die sich hinter dem System befinden, zu schützen. Hierfür müssen eigene Gruppen erzeugt werden. Dies ist aber relativ einfach. Dazu ist der folgende Eintrag in der FreeS/WAN-Konfigurationsdatei `ipsec.conf` hinzuzufügen:

```
conn private-net
    also=private # erbt alle Einstellungen der Gruppe
                private
    leftsubnet=192.168.2.0/24

conn private-or-clear-net
    also=private # erbt alle Einstellungen der Gruppe
                private
    leftsubnet=192.168.2.0/24

conn clear-or-private-net
    also=private # erbt alle Einstellungen der Gruppe
                private
    leftsubnet=192.168.2.0/24

conn clear-net
    also=private # erbt alle Einstellungen der Gruppe
                private
    leftsubnet=192.168.2.0/24
```

```
conn block-net
    also=private # erbt alle Einstellungen der Gruppe
                private
    leftsubnet=192.168.2.0/24
```

Nun müssen noch die einzelnen Dateien für die Gruppen im Verzeichnis `/etc/ipsec.d/policies` erzeugt werden. Dies erfolgt am einfachsten mit:

```
# cp -p /etc/ipsec.d/policies/private /etc/ipsec.d/policies/private-net
```

Dieser Befehl muss dann für jede Datei ausgeführt werden.

## 9.8 Einsatz von Hardware Kryptoprozessoren

Der Einsatz von Hardware Kryptoprozessoren erlaubt besonders auf Embedded Plattformen mit langsamen CPUs einen höheren VPN-Durchsatz. Diese Hardware Kryptoprozessoren wurden meist optimiert auf die (symmetrische) Verschlüsselung mit 3DES, Blowfish oder AES. Um sie einzusetzen sind jedoch besondere Treiber erforderlich, die bisher nicht von den Entwicklern von FreeS/WAN oder der Kernel-CryptoAPI unterstützt und gepflegt wurden. Daher sind die meisten dieser Treiber leider nur für ältere FreeS/WAN-Versionen verfügbar.

### 9.8.1 HiFn 7901

Ein Treiber für den HiFn 7901 Chip (<http://www.hifn.com/products/7901.html>) in Kombination mit FreeS/WAN 1.91 und dem Linux Kernel 2.2.18 wurde von Colubris entwickelt. Dieser Chip unterstützt 3DES und Public Keys. Zusätzlich verfügt er über einen echten Zufallszahlengenerator und kann Netzwerkverkehr mit einer Datenrate von bis zu 32MBit/s verschlüsseln. Der Treiber ist unter <http://sources.colubris.com/en/projects/FreeSWAN/> verfügbar. Auf der Seite befinden sich auch Hinweise für die Übersetzung des Treibers mit dem Linux Kernel 2.4 und FreeS/WAN 1.96.

### 9.8.2 HiFn 7811

Basierend auf dem HiFn 7901 Treiber hat Martin Gadbois einen Treiber für den HiFn 7811 Chip (<http://www.hifn.com/products/7811.html>) entwickelt (<http://sources.colubris.com/en/projects/FreeSWAN/hifn7811.tar.bz2>). Dieser Treiber ist speziell für den Linux Kernel 2.4.5. Der 7811 Chip von HiFn unterstützt die Verschlüsselung von IPsec Verkehr mit einer Bandbreite von 252

MBit/s mit 3DES und MD5. Zusätzlich unterstützt er Hardware Kompression, das PPTP und PPP Protokoll und bis zu 32000 IPsec Verbindungen gleichzeitig.

### 9.8.3 HiFn 7751

Für den alten HiFn 7751 Chip wird auf der Seite <http://hifn7751.sourceforge.net/> ein Treiber vorgehalten. Dieser Treiber unterstützt sowohl den Linux Kernel 2.2 als auch 2.4.

### 9.8.4 HiFn 7951

Der HiFn 7951 Chip bietet die gleichen Leistungsdaten, wie der HiFn 7901. Ein Beta-Treiber für diesen Chip wurde auf der Sourceforge-Seite <https://sourceforge.net/projects/hifn7951/> gepflegt. Leider hat im letzten Jahr keine wesentliche Entwicklung mehr stattgefunden. Auch dieser Patch baut wie der HiFn 7901 Treiber auf der Cryptolib von Martin Gadbois auf.

### 9.8.5 Zukunft der Hardware Kryptoprozessoren in Linux

Momentan ist die Unterstützung der Kryptoprozessoren in Linux im Gegensatz zu anderen Betriebssystemen wie OpenBSD eher spärlich (<http://www.openbsd.org/de/crypto.html#hardware>). Dies ändert sich hoffentlich, nachdem die CryptoAPI (<http://www.kernel.org>) in den Linux Kernel 2.6 aufgenommen wurde. Leider besitzt die CryptoAPI aber noch keine Unterstützung für Hardware.

## 9.9 Automatisches Laden der CRL

FreeS/WAN unterstützt seit der Version 1.1.0 des X.509 Patches das automatische Update der Certificate Revocation List (CRL) mit den Protokollen LDAP, HTTP und FTP. Hierzu muss für die Protokolle HTTP und FTP das Kommandozeilenwerkzeug cURL zur Verfügung stehen. Dieses Werkzeug wird auf der Webseite <http://curl.haxx.se> gepflegt und ist auch bereits Bestandteil vieler Linux Distributionen (wie zum Beispiel Red Hat Linux 9). Für das LDAP Protokoll muss während der Übersetzung von FreeS/WAN die OpenLDAP Bibliothek (<http://www.openldap.org>) zur Verfügung stehen. Auch diese ist in den meisten Linux Distributionen enthalten. Häufig heißt

das entsprechende Paket `openldap-devel`. Damit diese Bibliothek bei der Übersetzung von FreeS/WAN eingebaut wird, muss der Administrator vorher die Datei `programs/pluto/Makefile` anpassen und dort die entsprechenden Zeilen durch die Entfernung des Kommentarzeichens aktivieren.

```
# Uncomment this line to enable dynamic CRL fetching using
LDAP V3
#LDAP_VERSION=3
# Uncomment this line to enable dynamic CRL fetching using
LDAP V2
#LDAP_VERSION=2
```

Anschließend kann Pluto mit den Befehlen `make programs`; `make install` übersetzt und installiert werden.

Damit nun Pluto automatisch die entsprechenden CRLs im richtigen Zeitabschnitt lädt, ist es erforderlich, dass das CA Zertifikat die entsprechenden Informationen enthält. Hierbei handelt es sich um die X.509.v3 Erweiterung `crlDistributionPoint`. Ist diese Erweiterung vorhanden und gesetzt, kann anschließend mit dem Befehl `ipsec auto -listcrls` der aktuelle Zustand der CRLs überprüft werden:

```
Jul 31 08:25:51 2003, trials: 2
  issuer: 'C=DE, O=OpenSourceSecurity, CN= Root CA'
  distPts: 'http://vpn.spenneberg.com/ca/cert.crl'
           'ldap://ldap.spenneberg.com/o=OpenSourceSecurity, c=DE
           ?certificateRevocationList?base
           ?(objectClass=certificationAuthority)'
```

Um die Häufigkeit des CRL Updates zu bestimmen existiert nun eine neue Konfigurationsdirektive `crlcheckinterval` in der Datei `ipsec.conf`.

```
config setup
  crlcheckinterval = 600 # Sekunden
```

Außerdem besteht die Möglichkeit eine Richtlinie zu definieren, die das Verhalten von FreeS/WAN bestimmt, wenn keine aktuelle CRL zur Verfügung steht:

```
config setup
  strictcrlpolicy = yes
```

Wird die `strictcrlpolicy` gesetzt, so wird kein Zertifikat mehr von FreeS/WAN akzeptiert, wenn nicht eine aktuelle CRL der entsprechenden CA vorliegt.

## 9.9.1 Halbautomatisches Update der CRL

Ältere FreeS/WAN-Installationen und auch `racoon` und `isakmpd` unterstützen die automatische Aktualisierung der CRL nicht. Hier kann ein halbautomatisches System aufgebaut werden. Dazu wird ein Skript eingesetzt, das per Cronjob täglich oder auch stündlich prüft, ob eine neue CRL existiert. Ist dies der Fall, wird sie an die entsprechende Stelle kopiert und an den IKE-Daemon ein `SIGHUP` gesendet. Sowohl `racoon` als auch `isakmpd` reagieren auf das `SIGHUP` mit dem Neu-Einlesen der Konfigurationsdateien.

## 9.10 Hochverfügbarkeit

In vielen Umgebungen stellen Virtuelle Private Netzwerke zentrale Strukturen in der Netzarchitektur dar. Ein Ausfall dieser Systeme für einige Stunden oder Tage ist meist mit hohen Folgekosten verbunden. Können durch den Ausfall eines VPNs 20 Mitarbeiter einen Tag lang nicht mehr ihre Aufgaben erfüllen, so entspricht dies einem theoretischen Schaden von 20 Mann-Tagen, also etwa einem Monat.

Um einem derartigen Ausfall vorzubeugen, ist es sinnvoll die VPN-Gateways möglichst ausfallsicher zu konfigurieren. Hierzu gehört die Ausstattung mit RAID-Festplatten, redundanten Netzteilen, aber möglicherweise auch die hochverfügbare Auslegung durch die Bereitstellung eines zweiten VPN-Gateways, das bei Ausfall des ersten Gateways dessen Aufgaben übernimmt.

Ein derartiger hochverfügbarer Aufbau ist mit einer Einschränkung sehr einfach mit Open Source Werkzeugen realisierbar. Die wesentliche Einschränkung liegt in der Natur der IPsec Tunnel. Da die Schlüssel der IPsec Tunnel von den VPN Gateways ausgehandelt und nicht auf der Festplatte gespeichert werden, können aufgebaute Tunnel nicht durch den Backup Server übernommen werden. Das Backup System kann lediglich die Tunnel wieder neu aufbauen oder selbst neue Anfragen entgegennehmen. Bei Standleitungen, die mit einem VPN gesichert werden, stellt dies also keine echte Einschränkung dar. Roadwarrior müssen nach einem Ausfall der Verbindung diese neu initiieren.

Für den Aufbau einer derartigen Hochverfügbarkeitslösung stehen unter Linux verschiedene Werkzeuge zur Verfügung. Die meisten dieser Werkzeuge arbeiten nach demselben Prinzip, so dass hier nur Heartbeat vom Linux HA Projekt beschrieben werden soll.<sup>3</sup>

3. <http://www.linux-ha.org>

Heartbeat ist die zentrale Komponente des Linux HA Projektes. Es ermöglicht die ausfallsichere Verfügbarkeit eines Dienstes. Hierzu werden zwei Rechner benötigt. Diese Rechner haben eine identische Konfiguration. Damit die Rechner in der Lage sind sich gegenseitig zu überwachen, werden sie mit geeigneten Methoden verbunden. Hierzu verwendet man ein Cross over Kabel oder ein Nullmodem Kabel. Um Probleme durch den Ausfall eines Kabels zu vermeiden, werden sinnvollerweise zwei verschiedene Varianten angewandt.

Es besteht auch die Möglichkeit das normale Netzwerk für die Überwachung zu nutzen. Dies verkompliziert jedoch die Konfiguration von Heartbeat, da nun die Überwachung authentifiziert durchgeführt werden muss, um DoS-Angriffe zu vermeiden.

Der mit Heartbeat realisierte Cluster aus Master-Node und Backup Node ist in Abbildung 9.7 dargestellt. Hierbei verfügt jeder Node zunächst über eigene IP Adressen. Gemeinsam stellen beide Nodes die Verfügbarkeit des Dienstes unter der externen IP Adresse 3.0.0.3 und der internen IP Adresse 192.168.0.3 sicher.

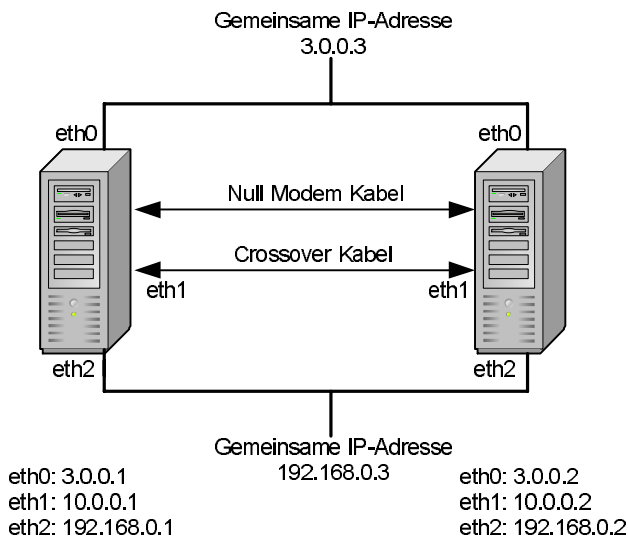


Abbildung 9.7 Der Heartbeat Cluster erlaubt den hochverfügbaren Aufbau von VPN Gateways

Die Installation und Konfiguration des Heartbeat Clusters ist sehr einfach. Zunächst müssen zwei Systeme identisch installiert und konfiguriert werden. Hierbei ist es aber wichtig, dass die Netzwerkkarten unterschiedliche IP Adressen (wie in Abbildung 9.7) erhalten. Anschließend sollte die Kommuni-

kation über das Cross Over Kabel und das Nullmodem Kabel getestet werden. Die Funktion des Cross Over Kabels kann sehr einfach mit einem Ping getestet werden. Um das Nullmodem Kabel zu testen, sollte auf einem Rechner der Befehl `cat < /dev/ttyS0` und auf dem anderen Rechner der Befehl `echo »Kabeltest« > /dev/ttyS0` eingegeben werden. Anschließend sollte auch die Gegenrichtung getestet werden. Bei Anschluss des Nullmodem Kabels an COM2 ist dementsprechend `/dev/ttyS1` zu verwenden.

Nun kann das Heartbeat-Paket von <http://www.linux-ha.org> heruntergeladen, ausgepackt und installiert werden. Für eine einfache Installation enthält das Paket auch die notwendigen Dateien um RPM-Pakete oder Debian-Pakete zu bauen. Für ein RPM-Paket ist die Datei `heartbeat.spec` verantwortlich. Mit dem Befehl `rpmbuild -bb heartbeat.spec` werden die Pakete `heartbeat`, `heartbeat-pils`, `heartbeat-ldirectord` und `heartbeat-stonith` erzeugt. Sie lassen sich nun sehr einfach mit dem Befehl `rpm -i` installieren.

Die Konfiguration von Heartbeat ist recht einfach, da die entsprechenden Dateien sehr gut kommentiert sind und dem Paket eine sehr gute Dokumentation beiliegt. Die Konfigurationsdatei in Listing 9.15 enthält aus Platzgründen diese Kommentare nicht.

```
debugfile /var/log/heartbeat-debug
logfile /var/log/heartbeat
keepalive 2
deadtime 30
initdead 120

serial /dev/ttyS0
baud 19200

udpport 694
bcast eth1

node vpn1.spenneberg.de
node vpn2.spenneberg.de
```

*Listing 9.15 Die Datei `/etc/ha.d/ha.cf` enthält die wesentlichen Einstellungen.*

Die Parameter `debugfile` und `logfile` definieren die Dateien für die Protokollierung von Ereignissen. Heartbeat sendet nun alle zwei Sekunden (`keepalive`) einen »Herzschlag« aus. Wurden 30 Sekunden keine Herzschläge empfangen, so wird der andere Knoten für tot erklärt (`deadtime`). Bei einem Neustart der Rechner dauert es möglicherweise einige Zeit, bis das Netzwerk zur Verfügung steht. Dieser Umstand soll nicht dazu führen, dass

der andere Knoten für tot erklärt wird (`initdead`). Für die Kommunikation soll die serielle Schnittstelle `/dev/ttyS0` genutzt (`serial`) und ein UDP Broadcast über `eth1` durchgeführt werden (`udp` und `bcast`). Die Namen der Knoten im HA-Cluster sind `VPN1` und `VPN2`. Diese Namen müssen mit der Ausgabe des Befehls `uname -n` übereinstimmen!

Zusätzlich wird die Datei `/etc/ha.d/authkeys` benötigt. Diese Datei definiert, wie die Heartbeat Meldungen authentifiziert werden. Da die Kommunikation nur über gesicherte Leitungen (Nullmodem- und Cross Over Kabel) erfolgt besteht hier keine Gefahr. Dennoch sollte hier ein Schlüssel gewählt werden (Listing 9.16).

```
auth 2
2 sha1 Ein geheimer Schlüssel
```

*Listing 9.16 Die Datei `/etc/ha.d/authkeys` definiert die Schlüssel für die Authentifizierung der »Herzschläge«*

Die letzte und wichtigste Datei bei der Konfiguration von Heartbeat ist die Datei `/etc/ha.d/haresources`. Diese Datei (Listing 9.17) enthält die Spezifikation der IP Adressen und Dienste, die bei einem Ausfall des ersten Knotens übernommen werden sollen.

```
vpn1.spenneberg.de 3.0.0.3 192.168.0.3 ipsec
```

*Listing 9.17 Die Datei `/etc/ha.d/haresources` definiert die ausfallsicheren Dienste*

Der in Listing 9.17 angegebene Eintrag sorgt nun dafür, dass `vpn1.spenneberg.de` zunächst der aktive Knoten ist. Bei Ausfall des Knotens übernimmt der zweite Knoten automatisch die IP Adressen 3.0.0.3 und 192.168.0.3 als `eth0:0` und `eth2:0` respektive. Zusätzlich startet der zweite Knoten den Dienst `ipsec`. Damit dies funktioniert, dürfen die hier angegebenen IP Adressen nicht im Betriebssystem konfiguriert werden. Heartbeat kümmert sich um deren Konfiguration nach seinem Start. Auch der Dienst `ipsec` darf nicht vom System automatisch gestartet werden. Ihn startet Heartbeat. Das Betriebssystem muss lediglich nach einem Reboot automatisch Heartbeat starten, damit der Cluster seine Aufgabe übernehmen kann.

Um die Funktion von Heartbeat zu testen, sollte es auf beiden Systemen gestartet werden. Anschließend sollte die IP Adresse 3.0.0.3 mit einem Ping erreichbar sein. Die Protokolldateien sollten den erfolgreichen Start von Heartbeat melden. Wird nun der Knoten `vpn1` ausgeschaltet, so sollte nach 30 Sekunden der Knoten `vpn2` dies merken und den Dienst übernehmen.

Bei der Konfiguration des VPN Gateways sind nun einige Besonderheiten zu berücksichtigen. FreeS/WAN verlangt die Angabe des externen Interfaces, das die verschlüsselten IPsec Pakete empfängt und versendet. Hierbei handelt es sich nun um `eth0:0`. Um diesem Umstand Rechnung zu tragen ist die folgende Zeile in der Datei `/etc/ipsec.conf` erforderlich:

```
interfaces="ipsec0=eth0:0"
```

Das VPN-Gateway besitzt offiziell nach außen die IP Adresse 3.0.0.3. Daher muss diese IP Adresse und nicht 3.0.0.1 oder 3.0.0.2 in den Konfigurationsdateien eingetragen werden. Im Falle von FreeS/WAN ist dies wie folgt möglich:

```
left=3.0.0.3
```

Sämtliche Rechner, die den VPN-Tunnel für ihre Kommunikation nutzen sollen, müssen als Gateway die IP Adresse 192.168.0.3 verwenden.

## 9.11 Smartcard Unterstützung

Seit der Version 1.4.0 des X.509-Patches für FreeS/WAN unterstützt dieser die Speicherung der privaten Schlüssel (Private Key) auf einer Smartcard – im Gegensatz zu `racoon` oder `isakmpd`. Dabei baut die Smartcard Unterstützung auf der OpenSC Bibliothek (<http://www.opensc.org>) auf. Sie ist in der Lage mit den verschiedensten Lesegeräten zusammenzuarbeiten. Hierzu können zwei verschiedene Treibersysteme eingesetzt werden: PCSC-lite (<http://www.linuxnet.com/middle.html>) und OpenCT. OpenCT wird ebenfalls vom OpenSC Team entwickelt und kann von dessen Homepage bezogen werden.

Diese Programme unterstützen eine ganze Reihe von Kartenlesegeräten (Towitoko, Kobil Kaan Professional, Aladdin eToken, Rainbow iKey 3000, Cryptoflex eGate und Eutron Crypto-Identity ITSEC) und Smartcards. Eine Liste der unterstützten Karten und Lesegeräte ist auf der Homepage der Software zu finden. Der Autor verwendet einen Towitoko Chipdrive Micro (<http://www.towitoko.com>) mit Schlumberger Cryptoflex 8K Karten (<http://www.smartcards.net>).

Neben FreeS/WAN können auch einige weitere Programme Smartcards für die Authentifizierung nutzen. Folgende Programme können OpenSC bisher nutzen:

- Netscape und Mozilla
- OpenSSH
- OpenSSL
- Pluggable Authentication Modules (PAM)

Dieses Kapitel beschreibt die notwendigen Programme, deren Installation und Konfiguration für den Einsatz mit FreeS/WAN.

## 9.11.1 Installation

Bevor FreeS/WAN mit der Smartcard Unterstützung übersetzt werden kann, müssen zunächst die entsprechenden Bibliotheken installiert werden. Hierbei handelt es sich um die OpenSC Bibliothek und mindestens eine der Bibliotheken OpenCT oder PCSC-Lite.

### Installation von OpenCT

Die Installation von OpenCT ist recht einfach. Hierzu wird zunächst das Quelltextarchiv von der Homepage (<http://www.opensc.org>) geladen und entpackt. Anschließend wird der Quelltext konfiguriert, übersetzt und installiert. Ein RPM-Paket kann von <http://www.spenneberg.org/VPN/SmartCards> heruntergeladen werden.

```
# ./configure
# make
# make install
```

### Installation von PCSC-Lite

Die Installation von PCSC-Lite ist genauso einfach, wie die Installation von OpenSC. Auch hierfür steht ein RPM Paket zur Verfügung, wenn die Installation nicht aus dem Quelltextarchiv erfolgen soll. Sie ist recht einfach und erfolgt mit:

```
# ./configure --enable-usb
# make
# make install
```

### Installation von OpenSC

Nun kann OpenSC installiert werden. Bei der Installation wird anschließend der Pfad zu OpenCT und PCSC-Lite angegeben. Auch diese Installation ist mit einem RPM Paket möglich. Sie kann aber auch sehr einfach aus dem Quelltextarchiv erfolgen.

```
# ./configure --with-openct=path --with-pcsclite=path
# make
# make install
```

## Installation von FreeS/WAN

Die Installation von FreeS/WAN wurde bereits in dem entsprechenden Kapitel besprochen. Für die Unterstützung von Smartcards ist der X.509-Patch ab Version 1.4 für FreeS/WAN 2 erforderlich. Leider existiert bisher noch keine Super FreeS/WAN-Version, die diesen Patch enthält. Der Super FreeS/WAN-Maintainer Ken Bancoft konzentriert sich momentan noch auf die FreeS/WAN-Versionen 1.99. Es ist aber sehr wahrscheinlich, dass sich das bei Erscheinen des Buches bereits geändert hat.

Nachdem der Patch eingespielt wurde, muss im Quelltext bei folgender Zeile das Kommentarzeichen entfernt werden:

```
#Uncomment this line to enable smartcard support
SMARTCARD=1
```

*Listing 9.18 Datei `programs/pluto/Makefile`*

Nun müssen die FreeS/WAN Programme neu übersetzt werden.

```
# make programs
# make install
```

Eine erneute Übersetzung des Kernels ist nicht erforderlich, schadet aber auch nicht.

### 9.11.2 Konfiguration des Lesegerätes und der Karte

Zunächst ist es erforderlich, dass der Kartenleser konfiguriert wird. Dies erfolgt in der Datei `/etc/openct.conf`. Für einen Towitoko-Kartenleser an einer seriellen Schnittstelle ist dort folgender Eintrag erforderlich:

```
reader towitoko {
    driver = towitoko;
    device = /dev/ttyS0;
};
```

Um Hotplug-fähige USB-Lesegeräte zu unterstützen sind etwas mehr Schritte erforderlich. Zunächst muss die Datei `/etc/hotplug/usb.usermap` um die folgenden Zeilen erweitert werden (diese Zeilen befinden sich auch in der Dokumentation).

```

openct 0x0003 0x0973 0x0001 0x0000 0x0001 0xff 0x00 0x00 0xff 0x00
0x00 0x0000 0000
openct 0x0003 0x0529 0x050c 0x0000 0x0001 0xff 0x00 0x00 0xff 0x00
0x00 0x0000 0000
openct 0x0003 0x0529 0x0514 0x0000 0x0001 0xff 0x00 0x00 0xff 0x00
0x00 0x0000 0000
openct 0x0003 0x073d 0x0005 0x0000 0x0001 0xff 0x00 0x00 0xff 0x00
0x00 0x0000 0000
openct 0x0003 0x04b9 0x1300 0x0000 0x0001 0xff 0x00 0x00 0xff 0x00
0x00 0x0000 0000
openct 0x0003 0x076b 0x0596 0x0000 0x0001 0xff 0x00 0x00 0xff 0x00
0x00 0x0000 0000

```

Anschließend muss eine symbolische Verknüpfung `/etc/hotplug/usb/openct` auf die Datei `hotplug.openct` erzeugt werden.

```

mkdir /etc/hotplug/usb
ln -s /path/to/openct/sbin/hotplug.openct /etc/hotplug/usb/openct

```

Nun sollte beim Einstecken des USB Lesegerätes automatisch der Befehl `openct-control init` aufgerufen werden.

Dieser Befehl muss beim seriellen Lesegerät von Hand gestartet werden. Damit dies bei jedem Systemstart geschieht, enthält das Paket ein SysV Init Skript `/etc/init.d/openct`, das diese Aufgabe übernimmt.

Wurde die Software richtig eingerichtet und installiert, so sollte der Befehl `openct-control funktionieren`.

```

# openct-control init
# openct-control status
No.    Name                               Info

=====
  0    Towitoko Chipdrive Micro          slot0: empty
# ps -ax | grep ifdhandler
7087  ttyS0    S          0:00 /usr/sbin/ifdhandler -r0 towitoko /dev/ttyS0
# openct-control shutdown

```

Sobald eine Karte eingelegt wird, kann auf sie zugegriffen werden:

```

# openct-control status
No.    Name                               Info

=====
  0    Towitoko Chipdrive Micro          slot0: card present

```

**# opensc-explorer**

```
OpenSC Explorer version 0.8.0
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
OpenSC [3F00]> quit
```

Nun kann auf der Karte eine PKCS#15 Struktur für die Speicherung von RSA Schlüsseln und Zertifikaten angelegt werden. Hierzu wird der Befehl `pkcs15-init` verwendet. Sie erzeugt zunächst die Speicherstruktur, in der anschließend die Schlüssel gespeichert werden.

**# pkcs15-init --create-pkcs15**

```
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
About to create PKCS #15 meta structure.
New security officer (SO) PIN required (press return for no PIN).
Please enter PIN: <enter>
Transport key (External authentication key #1) required.
Please enter key in hexadecimal notation (e.g. 00:11:22:aa:bb:cc),
or press return to accept default.
```

To use the default transport keys without being prompted, specify the `--use-default-transport-keys` option on the command line (or `-T` for short), or press `Ctrl-C` to abort. Please enter key [2c:15:e5:26:e9:3e:8a:19]: <enter>

Die von mir eingesetzten Karten unterstützen scheinbar keinen Security Officer. Die Eingabe einer PIN und eines PUK führen hier zu einem Fehler. Die Angabe `-no-so-pin` überspringt die Abfrage des Security Officers. Wenn für die Übertragung der Daten zur Karte immer dieselben Transportschlüssel verwendet werden sollen, muss hier keine Abfrage erfolgen. Das kann mit der Option `-use-default-transport-keys` aktiviert werden. Die ganze Zeile sieht dann so aus:

**# pkcs15-init --erase-card --use-default-transport-keys**

```
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
About to erase card.
```

**# pkcs15-init --create-pkcs15 --no-so-pin --use-default-transport-keys**

```
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
About to create PKCS #15 meta structure.
```

## ACHTUNG

Die Verwendung der Default Transportschlüssel erlaubt es jeder beliebigen Person mit Zugriff auf einen Kartenleser, die Karte zu löschen. Die Daten, die auf der Karte gespeichert wurden, können nicht gelesen werden, aber die Karte kann mit neuen Informationen gefüllt werden. Hier ist es möglicherweise sinnvoll einen anderen Transportschlüssel zu wählen. Im weiteren Kapitel wird aber der Einfachheit halber mit den Default Transport Keys gearbeitet.

Nun müssen die Schlüssel auf der Karte gespeichert werden. Zukünftige Versionen von OpenSC werden die Generierung der Schlüssel auf der Karte mit folgendem Befehl unterstützen. Aktuell ist dieser Befehl noch nicht funktionstüchtig:

```
# pkcs15-init --generate-key rsa/2048 --auth-id 1 --pin "12345678" --puk
"87654321" --label "my Pin" --store-pin --use-default-transport-keys
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
Found OpenSC Card
About to store PIN.
About to generate key.
Warning: card doesn't support on-board key generation.
Trying software generation
Updating RSA private key...
Updating RSA public key...
```

Daher muss der Schlüssel momentan manuell mit OpenSSL generiert und im PEM Format auf die Karte übertragen werden. Hierfür wird zunächst eine PIN und eine PUK auf der Karte gespeichert. Die PIN und der PUK kann auf der Kommandozeile mit den Optionen `-pin` und `-puk` angegeben werden. Aus Sicherheitsgründen sollte dies jedoch unterbleiben. Die Angaben auf der Kommandozeile werden in der Bash-History gespeichert und können bei Ausführung des Kommandos vom `ps`-Befehl angezeigt werden.

```
# pkcs15-init --auth-id 1 --store-pin --label "VPN Pin"
--use-default-transport-keys
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
Found OpenSC Card
About to store PIN.
New user PIN required.
Please enter PIN: <6-stellige PIN>
Please type again to verify: <6-stellige PIN>
```

```
Unlock code for new user PIN required (press return for no PIN).
```

```
Please enter PIN: <6-stellige PUK>
```

```
Please type again to verify: <6-stellige PUK>
```

Die PIN wird nun benötigt um auf die in der Karte gespeicherten Informationen zuzugreifen. Gibt der Benutzer mehr als dreimal die PIN falsch ein, so wird die Karte gesperrt und kann nur durch den PUK entsperrt werden.

Nun können der private RSA-Schlüssel (Private Key) und das X.509 Zertifikat auf der Karte gespeichert werden. Wenn der Schlüssel mit einer Passphrase geschützt ist, so kann diese mit der Option `-passphrase` auf der Kommandozeile angegeben werden. Erfolgt das nicht, so fragt der Befehl die Passphrase interaktiv ab (empfohlen). Sehr angenehm ist, dass der Befehl den RSA-Schlüssel auch aus der Request-Datei des OpenSSL-CA-Befehls extrahieren kann.

```
# pkcs15-init --auth-id 1 --store-private-key berlin_req.pem --id 45
--use-default-transport-keys
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
Found OpenSC Card
About to store private key.
Enter PEM pass phrase: passphrase
Updating RSA private key...
Updating RSA public key...
```

Dieser Befehl speichert den Schlüssel mit der ID 45 ab. Dies ist der Default-Wert. Durch die Angabe einer anderen ID können mehrere Schlüssel auf einer Smartcard gespeichert werden. Die Anwendungen können dann später zwischen den Schlüsseln wählen.

Um das Zertifikat auf der Smartcard zu speichern ist nun folgender Befehl erforderlich:

```
# pkcs15-init --auth-id 1 --store-certificate berlin_cert.pem --id 45
--use-default-transport-keys
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
Found OpenSC Card
About to store certificate.
```

Die abgespeicherte Struktur auf der Smartcard kann nun mit dem Befehl `pkcs15-tool` betrachtet werden:

```
# pkcs15-tool --list-certificates --list-pins --list-keys
Connecting to card in reader Towitoko Chipdrive Micro...
Using card driver: Schlumberger Multiflex/Cryptoflex
Trying to find a PKCS#15 compatible card...
```

```
Found OpenSC Card!
Card has 1 certificate(s).
```

```
X.509 Certificate [Certificate]
  Flags      : 2
  Authority: no
  Path       : 3F0050155501
  ID         : 45
```

```
Card has 1 private key(s).
```

```
Private RSA Key [Private Key]
  Com. Flags : D
  Usage      : [0x4], sign
  Access Flags: [0x0]
  ModLength  : 1024
  Key ref    : 0
  Native     : yes
  Path       : 3F0050154B010012
  Auth ID    : 01
  ID         : 45
```

```
Card has 1 PIN code(s).
```

```
PIN [VPN Pin]
  Com. Flags: 0x3
  Auth ID   : 01
  Flags     : [0x32], local, initialized, needs-padding
  Length    : min_len:4, max_len:8, stored_len:8
  Pad char  : 0x00
  Reference : 1
  Type     : 1
  Path     : 3F0050154B01
```

Diese Karte kann nun für den Aufbau eines VPNs mit FreeS/WAN verwendet werden. Die Schlüssel sind auf dieser Karte nun sicher gespeichert. Ein Auslesen des privaten Schlüssels ist nicht möglich. Um den privaten Schlüssel einzusetzen ist die mindestens 6-stellige PIN erforderlich.

### 9.11.3 Anwendung in FreeS/WAN

Damit FreeS/WAN die Smartcard nutzen kann, muss der Administrator die Konfiguration anpassen. Der private RSA Schlüssel und das X.509 Zertifikat werden vom IKE Daemon Pluto für die Authentifizierung bei der Erzeugung der ISAKMP SA benötigt. Dabei verwendet Pluto die Schlüssel nicht mehr selbst, sondern beauftragt die Smartcard, die entsprechenden Informationen zu signieren.

Hierzu ist in der FreeS/WAN Konfigurationsdatei `/etc/ipsec.conf` die Konfiguration der Verbindung wie folgt anzupassen:

```
conn smartcard
    right=3.0.0.1
    rightright=3.255.255.254
    rightid=@vpn.spenneberg.com
    rightrsasigkey=%cert
    left=5.0.0.1
    leftnextrhop=5.255.255.254
    leftid=@client.spenneberg.com
    leftcert=%smartcard
    auto=add
```

Wird das Zertifikat so angegeben, so greift Pluto auf den ersten RSA Schlüssel im ersten Smartcard Leser zu. Existieren mehrere Schlüssel oder Lesegeräte, so können diese mit der Direktive `leftcert` oder `rightcert` explizit angegeben werden:

```
leftcert=%smartcard<lesegerät nr>:<ID>
```

Der Wert `ID` bezieht sich auf die bei der Speicherung des Schlüssels angegebene ID. Die Angabe `%smartcard` entspricht also der Angabe `%smartcard0:45`.

Nun muss Pluto für den Zugriff auf die Smartcard die PIN kennen und an die Smartcard übergeben können. Hierfür existieren im Grunde zwei verschiedene Varianten:

1. Die PIN wird in Klartext in der Datei `ipsec.secrets` eingetragen. Hierfür wird die folgende Zeile der Datei hinzugefügt:

```
: PIN %smartcard "12345678"
```

Auch hier ist es wieder möglich die Nummer des Lesegerätes und die ID anzugeben. Dabei wird die gleiche Syntax verwendet wie oben. Hier besteht aber das Problem, dass die PIN auf dem System gespeichert ist. Dies ermöglicht einen automatischen Start der VPN Verbindung ohne Interaktion eines Benutzers.

2. Die PIN wird nicht auf dem System gespeichert. Wenn das System die Verbindung startet fragt es die PIN vom Benutzer ab. Hierzu ist die Angabe `%prompt` in der Datei `/etc/ipsec.secrets` erforderlich.

```
: PIN %smartcard %prompt
```

Aber auch diese Lösung ist nicht ohne Probleme. Da FreeS/WAN erst die PIN verlangt, wenn der Tunnel aufgebaut wird, ist dies bei einem VPN

Gateway, die nur Tunnel anbietet, aber selbst keine aufbaut, keine gangbare Lösung. Der Tunnel kann zu jeder Zeit von außen aufgebaut werden und dann wird die PIN benötigt. Zu diesem Zeitpunkt ist möglicherweise noch nicht einmal ein Benutzer angemeldet. Um diese PIN bereits im Vorfeld zu laden, kann der Befehl `ipsec auto -rereadsecrets` verwendet werden. Er fordert dann vom Benutzer die PIN an und hält sie im Arbeitsspeicher vor. Mit dem Befehl `ipsec auto -listcards` kann der Zustand der Karten und der PINs überprüft werden.

