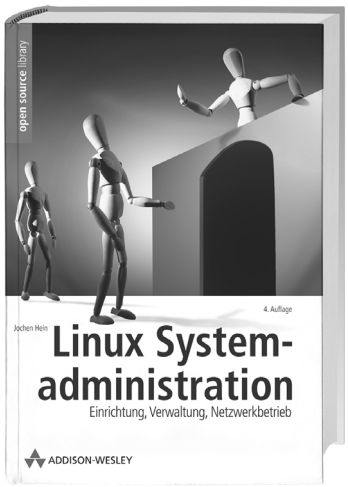
VPN mit Linux

open source library

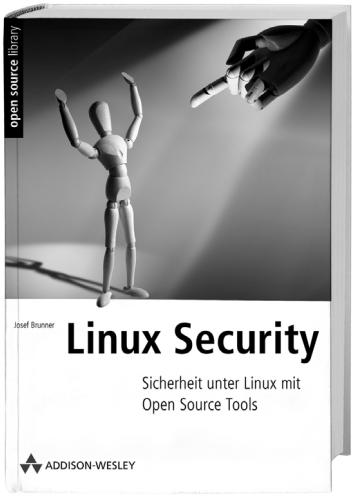
Open Source Software wird gegenüber kommerziellen Lösungen immer wichtiger. Addison-Wesley trägt dieser Entwicklung Rechnung mit den Büchern der **Open Source Library**. Administratoren, Entwickler und User erhalten hier professionelles Know-how, um freie Software effizient einzusetzen. Behandelt werden sowohl Themen wie Betriebssysteme, Netzwerke und Sicherheit als auch Programmierung.

Eine Auswahl aus unserem Programm:



Linux für den fortgeschrittenen Systemadministrator, der lernt, wie man Linux bei verteilten Netzumgebungen einsetzt. Die vierte Auflage wurde durchgehend überarbeitet, aktualisiert und erweitert. Wesentliche Neuerungen betreffen Linux-Standards, XML-Tools, Internet-Zugang mit DSL, VPNs, BIND9/dnssec.

Linux-Systemadministration
Jochen Hein
643 Seiten
EUR 49,95 [D], sFr 77,50
ISBN 3-8273-1992-7



Sicherheit ist ein Problem aller Betriebssysteme, und meist ist es teuer, eine Installation wirklich sicher zu machen. In diesem Buch zeigt der Autor, dass dies auch ohne einen größeren finanziellen Aufwand möglich ist. Hier erfahren Sie, wie Sie Linux mit Hilfe von Open-Source-Tools wie z.B. LIDS, Snort, NMap, Webmin oder Nessus sicher machen.

Linux Security
Josef Brunner
518 Seiten
EUR 49,95 [D], 51,40 [A]
ISBN 3-8273-1999-4

Ralf Spenneberg

VPN mit Linux

Grundlagen und Anwendung Virtueller Privater Netzwerke mit Open Source-Tools

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Umwelthinweis:
Dieses Produkt wurde auf chlorfrei gebleichtem Papier gedruckt.

06 05 04

ISBN 3-8273-2114-X

ein Imprint der Pearson Education Deutschland GmbH
Martin-Kollar-Straße 10–12, D-81829 München/Germany

Einbandgestaltung: Marco Lindenbeck (mlindenbeck@webwo.de)
Fachliche Redaktion: Wilhelm Dolle, Berlin
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Korrektorat: Bettina Blaß, Köln
Herstellung: Philipp Burkart, pburkart@pearson.de
Satz: reemers publishing services gmbh, Krefeld, www.reemers.de
Druck: Bercker, Kevelaer
Printed in Germany

Für Claudia

Inhaltsverzeichnis

Vorwort

Dieses Buch versucht den Aufbau von Virtuellen Privaten Netzwerken (VPN) mit Linux zu beschreiben. Dabei sollen dem Anwender die verschiedenen unter Linux zur Verfügung stehenden Werkzeuge und Vorgehensweisen nahe gebracht werden. Um einen Einsatz in heterogenen Netzwerken zu ermöglichen, beschränkt sich das Buch jedoch auf die anerkannte und seit Jahren im Einsatz befindliche Protokollfamilie IPsec. Es wird vorausgesetzt, dass Sie mit Linux oder einem anderen kommerziellen UNIX Derivat grundsätzlich vertraut sind. Außerdem werden grundlegende Netzwerkkenntnisse erwartet.

Das Buch ist in mehrere Teile gegliedert:

- Grundlagen
- Praktische Umsetzung
- Fortgeschrittene Konfiguration und Fehlersuche

Teil 1 *Grundlagen* beginnt mit einer allgemeinen Einführung in die Technik und Grundlagen Virtueller Privater Netzwerke. Anschließend werden die verschiedenen Verschlüsselungsalgorithmen, die hier zum Einsatz kommen erklärt. Ihre Kenntnis ist bei der Anwendung eines VPN nicht zwingend notwendig, jedoch erlaubt ein gewisses Grundwissen eine Einschätzung ihrer Sicherheit.

Schließlich werden die eingesetzten Protokolle der IPsec Familie betrachtet und analysiert. Auch dieses Wissen ist nicht zwingend erforderlich für einen Betrieb eines VPNs. Bei einer Fehlersuche ist dieses Hintergrundwissen jedoch sehr nützlich, wenn nicht sogar obligatorisch.

Sollte das entsprechende Grundwissen bereits vorhanden sein, oder Sie ungeduldig mit dem Aufbau eines VPNs beginnen wollen, so können Sie direkt mit Teil 2 des Buches beginnen. Für das Verständnis, die Planung und die Fehlersuche empfiehlt es sich jedoch zu einem späteren Zeitpunkt Teil 1 nachzulesen.

Teil 2 *Praktische Umsetzung* beschreibt den Aufbau einfacher Virtueller Privater Netzwerke mit den momentan zur Verfügung stehenden Implementierungen für den Linux Kernel 2.4 und 2.6, FreeS/WAN, `racoon` und `isakmpd`. Hierbei werden die Konfigurationsparameter beschrieben und einfache VPN Lösungen aufgebaut.

Teil 3 *Fortgeschrittene Konfiguration und Fehlersuche* beschreibt besondere Eigenheiten der Implementierungen und stellt zusätzliche Funktionalitäten vor, die für die eine oder andere Implementierung einzigartig sind. Hier werden Funktionen besprochen, wie NAT Traversal, DHCP-over-IPsec und opportunistische Verschlüsselung. Auch die Unterstützung von Hardware Kryptoprozessoren zur Beschleunigung der Verschlüsselung wird hier angesprochen.

In Teil 3 werden auch die wichtigsten Werkzeuge zur Fehlersuche vorgestellt. Außerdem enthält dieser Teil auch die häufigsten Fehlermeldungen und ihre wahrscheinlichsten Ursachen.

Bei der Strukturierung des Buches habe ich versucht, die Aufgabenstellungen bei Aufbau eines VPNs in Schritt für Schritt Anleitungen durch zusprechen. Sicherlich wird dabei nicht jedes Problem geklärt. Um diesem Missstand Rechnung zu tragen, behandelt Teil 3 komplizierte und besondere Fragestellungen. Hierbei ist es jedoch nur möglich Einblicke in bestimmte Probleme zu geben. Sie sollen als Anregung aufgefasst werden und den Lösungsweg aufzeigen.

Beim Schreiben des Buches waren behilflich: Fridtjof Busse, Wilhelm Dolle und Thomas Walpuski. Diesen Personen möchte ich hierfür danken.

Alles in allem hoffe ich, dass mir ein Buch gelungen ist, mit dem Sie in der Lage sind, den Einsatz von Virtuellen Privaten Netzwerken auf der Basis von Linux abzuwägen und umzusetzen. Schließlich bleibt mir nur, Ihnen dabei Spaß und viel Erfolg zu wünschen.

Kontakt für Rückfragen und Anmerkungen

Virtuelle Private Netzwerke sind ein sehr aktuelles Thema. Dies führt dazu, dass die entsprechenden Technologien und Produkte ständig weiterentwickelt werden. Die in diesem Buch besprochenen Themen sind dabei sicherlich keine Ausnahme. Wenn Sie also zum Inhalt dieses Buches Updates, Korrekturen oder einfach Anregungen loswerden möchten, können Sie mich unter *ralf.spenneberg@mut.de* erreichen. Sofern es das Volumen zulässt, bin ich auch gerne bereit Fragen zum Thema zu beantworten. Unabhängig davon werde ich versuchen unter *http://www.spenneberg.com/* Updates und Korrekturen zum Buch zu veröffentlichen.

Teil I
Grundlagen

Dieser Teil des Buches behandelt die allgemeinen und theoretischen Grundlagen bei dem Aufbau von virtuellen privaten Netzwerken mit Linux. Die Informationen in diesem Teil sind für das Verständnis und die Wartung von derartigen VPNs unbedingt erforderlich. Wenn Sie dieses Wissen bereits besitzen oder Sie ungeduldig mit dem Aufbau eines VPNs beginnen wollen, so können Sie direkt zum Teil 2 vor blättern. Ich möchte Ihnen jedoch auch bei entsprechender Erfahrung empfehlen später den Teil Eins nachzulesen. Vielleicht findet sich doch noch die eine oder andere Information die für Sie interessant ist.

1 Einleitung

Virtuelle Private Netzwerke (VPN) erlauben eine sichere, stabile und preisgünstige Kommunikation über das Internet. Mit ihrer Hilfe können verteilte Unternehmensnetze verbunden werden oder Außendienstmitarbeiter auf Ressourcen und Daten in dem Unternehmensnetz zugreifen. Sie bieten eine kostengünstige und sichere Anbindung von Filialen an eine Zentrale und erlauben den Einsatz von Telearbeitsplätzen, bei denen die Angestellten von Zuhause ihre Arbeit erledigen.

Die Implementierung eines VPNs erforderte bis vor wenigen Jahren immer die Beteiligung eines Providers und den Einsatz komplizierter und kostspieliger Hardware. Seit einiger Zeit besteht jedoch die Möglichkeit derartige VPNs komplett softwarebasiert auf der Basis des Internet Protokolls IP zu implementieren. Es existieren eine ganze Reihe von kommerziellen Lösungen, die unterschiedlich gut und häufig sehr kostspielig sind.

Das Open Source Betriebssystem Linux wird in den letzten Jahren und Monaten immer häufiger als Alternative zur Senkung der Lizenzkosten bei proprietärer Software eingesetzt. Auch seine Firewallfähigkeiten sind allgemein anerkannt. Leider wissen bisher nur wenige, dass Linux auch in der Lage ist anspruchsvolle VPN Lösungen zu bieten. Dieses Buch versucht, die Möglichkeiten und Grenzen der unter Linux existierenden Technologien aufzuzeigen und eine Anleitung für die Praxis zu geben.

Um speziell den praktischen Gesichtspunkt nicht außer Acht zu lassen, werden im Rahmen dieses Buches verschiedene klassische Szenarien für den Einsatz eines VPNs besprochen und die Linux Lösungen beschrieben.

1.1 Was ist ein Virtuelles Privates Netzwerk?

Ein Virtuelles Privates Netzwerk (VPN) ermöglicht die private vertrauliche Kommunikation über ein öffentliches und eigentlich unsicheres Netz. Hierzu werden virtuelle Verbindungen genutzt, die ein Abhören und Modifizieren der Informationen unmöglich machen. Als öffentliches Transfernetz wird hier meist das Internet gewählt.

Derartige VPNs erlauben eine kostengünstige Kommunikation zwischen Unternehmensnetzen weltweit. Wenn in der Vergangenheit eine dedizierte Leitung durch einen Kommunikationsanbieter bereitgestellt werden musste,

genügt nun die Anbindung über einen lokalen Internet Service Provider (ISP). So können auch lokale Netze (Local Area Network, LAN) an vollkommen unterschiedlichen Standorten, etwa Berlin und New York, ohne dedizierte Leitung, bei der der Telefonanbieter die Vertraulichkeit durch sein eigenes Netzwerk bereitstellt, sicher und vertraulich kommunizieren.

Ein VPN ermöglicht auch den sicheren Zugang zu internen Ressourcen für Außendienstmitarbeiter die von unterschiedlichen Standorten zugreifen wollen. Bisher wurden hierzu meist Modem Pools durch die Unternehmen zur Verfügung gestellt, bei denen sich der Außendienstmitarbeiter – meist über ein Ferngespräch – einwählen konnte. Die Verbindungskosten sind jedoch hoch und garantieren dennoch nicht die Vertraulichkeit, wenn die Informationen nicht verschlüsselt übertragen wurden.

Ein VPN kann hier eine Lösung bieten, da die Außendienstmitarbeiter sich nun über nationale oder internationale ISPs in das Internet einwählen können. Anschließend kann ein VPN auf der Basis dieser Internetverbindung aufgebaut werden. So kann die Vertraulichkeit und Integrität der ausgetauschten Informationen sichergestellt werden.

1.2 Aufgaben eines VPN

Ein Virtuelles Privates Netzwerk stellt ein Sicherheitsinstrument dar. Um die Aufgaben eines VPNs richtig verstehen zu können, sollen zunächst einige Risiken im Internet aufgezählt und erläutert werden.

1.2.1 Gefahren im Internet

Ein lokales Netzwerk (LAN) ist üblicherweise relativ gut geschützt. Ein Zugriff von außen ist nicht möglich. Wenn die Benutzer des Netzwerks vertrauenswürdig sind, ist mit bösartigen Angriffen nicht zu rechnen. Häufig genügen dann einfache Maßnahmen um die Sicherheit und Funktionalität des Netzwerkes zu gewährleisten. Hierbei handelt es sich um Maßnahmen zur Sicherung der Daten (Backup), zum Schutz vor Viren und vor Hardwareausfall. Es sollte jedoch nicht vergessen werden, dass allgemein davon ausgegangen wird, dass 40 bis 60 Prozent aller erfolgreichen Angriffe von Innen ausgeführt werden. Hier kann die Intrusion Detection eine Hilfe sein (siehe auch *Intrusion Detection für Linux Server*, Ralf Spenneberg, Markt+Technik Verlag 2002, ISBN 3-8272-6415-4).

Sobald dieses LAN jedoch mit dem Internet verbunden wird, kommen einige wesentliche Gefahren hinzu. Das Internet ist ein anonymes Netzwerk, das keine Schutzmechanismen bietet. Angriffe sind meist nicht verfolgbar und sehr einfach auszuführen. Sämtliche Informationen, die im LAN transportiert und gespeichert werden, sind nun diesen Angriffen ausgesetzt. Dies gilt insbesondere für die Daten, die über das Internet transportiert werden.

Im wesentlichen existieren vier verschiedene Gefahren im Internet:

- **Einbruch:** Der Einbruch in ein Netzwerk ist eine sehr große Gefahr bei der Anbindung dieses Netzwerkes an das Internet. Einbrüche sind möglich, da die eingesetzten Produkte Sicherheitslücken aufweisen oder bei der Administration und Konfiguration dieser Produkte Fehler gemacht wurden. Der Einbrecher entdeckt diese Lücken und Fehler und nutzt diese aus, um sich Zugang zum LAN zu verschaffen. Üblicherweise wird eine Firewall eingesetzt, um derartige Einbrüche zu vereiteln.

  Sobald der Einbrecher aber erfolgreich eine Sicherheitslücke ausgenutzt hat (Exploit), ist er in der Lage sämtliche Funktionen des LANs zu nutzen, zu stören und möglicherweise auch umzukonfigurieren.
- **Falsche Identität:** Eine große Gefahr im Internet besteht in seiner scheinbaren Anonymität. Es existiert nicht die Möglichkeit einwandfrei die Identität eines Kommunikationspartners zu garantieren. Dieser kann eine falsche IP Adresse (IP Spoofing), eine falsche MAC Adresse (ARP Spoofing) oder sogar einen falschen DNS Namen (DNS Spoofing) vortäuschen. So besteht grundsätzlich die Möglichkeit, dass ein Angreifer vortäuscht, den DNS Namen www.sparkasse.de zu besitzen. Mögliche Besucher der Website werden dann auf die Website des Angreifers geleitet. Existieren keine weiteren über die TCP/IP Protokolle hinausgehende Methoden zur Feststellung der Authentizität der Website, kann dieser Angriff durch den Besucher nicht erkannt werden.
- **Lauschangriff (Sniffen):** Die TCP/IP Protokolle bieten keinen Schutz vor dem Abhören der übertragenen Informationen. Alle klassischen Protokolle der TCP/IP Familie in Version 4 übertragen die Informationen im Klartext. Daher ist es die Aufgabe der Applikationsprotokolle oder des Benutzers die Daten zu verschlüsseln, bevor sie an die TCP/IP Protokolle übergeben werden. Dies ist recht umständlich und erfordert einen zusätzlichen Aufwand durch den Benutzer. Erfolgt keine Verschlüsselung, so können die Daten entlang der gesamten Verbindungsstrecke mit gelesen werden (siehe 1.1). Eine Vertraulichkeit der Daten ist nicht gewährleistet.
- **Modifikation der Daten:** Sobald ein Mitlesen der Daten möglich ist, können diese Daten auch bei deren Transport verfälscht oder zusätzliche Daten eingeschleust werden. Die klassischen Protokolle der TCP/IP Familie

bieten keinerlei Integritätsschutz der transportierten Informationen. Ein Angreifer kann übertragene E-Mails oder Word-Dokumente bei deren Transport modifizieren oder erweitern. Diese Modifikation ist durch den Empfänger nicht zu erkennen (siehe Abbildung 1.1).

Ein VPN hat die Aufgabe in Zusammenarbeit mit anderen Maßnahmen einen Schutz vor diesen Gefahren zu bieten.

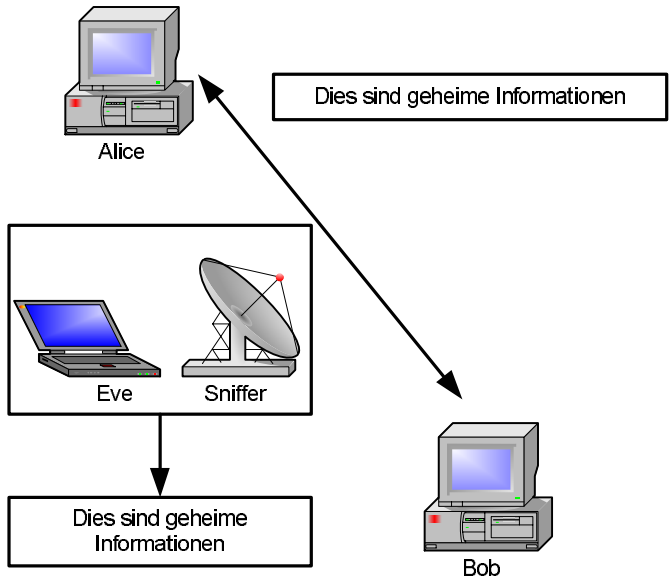


Abbildung 1.1 Tauschen Alice und Bob geheime Informationen unverschlüsselt aus, kann Eve mit einem Sniffer diesen Austausch mitlesen

1.2.2 Schutz durch eine Firewall

Der Schutz vor einem Einbruch wird üblicherweise durch eine Firewall gewährleistet. Die Aufgabe dieser Firewall ist es den Netzwerkverkehr über die Firewall zu untersuchen und zu kontrollieren.

Der Begriff Firewall ist vermutlich grundsätzlich bekannt. Hier soll aber dennoch kurz der Begriff und die verwendeten Technologien erläutert werden. In der Autoindustrie kennzeichnet das Wort Firewall die Wand, die den Motorraum von den Insassen trennt. Sie stellt einen Schutz vor einem möglichen Motorbrand dar, und muss in der Lage sein, diesem zu widerstehen.

Übertragen auf Computernetze stellt eine Firewall ein trennendes Gerät zwischen mindestens zwei Netzen dar. Sie unterbindet den ungehinderten Austausch von Informationen. Lediglich ausgewählte Daten dürfen transportiert

werden. Damit die Firewall diese Funktion erfüllen kann, darf keine zusätzliche Verbindung zwischen den beiden Netzen existieren, die eine Umgehung der Firewall erlauben würde. Die Firewall muss die einzige Verbindung sein.

Es gibt nun verschiedene Techniken eine Firewall zu implementieren. Die beiden am weitesten verbreiteten Techniken sind der Paketfilter und der Filter auf der Schicht des Applikationsprotokolls, häufig auch als Proxy bezeichnet. In vielen Fällen setzen Firewallsysteme beide Techniken ein. Beispiele für Open Source Produkte, die diese Techniken implementieren, sind *ipchains* und *iptables (netfilter)* als Paketfilter und *squid* und *httpf* als Proxy. Beide Ansätze unterscheiden sich stark in ihrer Performanz und in ihren Filtermöglichkeiten.

Paketfilter

Wie es der Name schon sagt: Der reine Paketfilter ist in der Lage Pakete zu filtern. Dazu betrachtet er die Header der IP Pakete. Die meisten Paketfilter können den IP Header und, wenn vorhanden, auch den TCP, UDP und ICMP Header lesen und verarbeiten. Bei diesen Informationen handelt es sich um die IP Adressen, das IP Protokoll, zum Beispiel TCP, UDP, ICMP, IGMP, ESP, AH, wenn vorhanden die TCP und UDP Ports und den ICMP Code. Weitere Informationen im IP Header sind beispielsweise der Fragmentierungszustand, Länge des Paketes, TTL und TOS Werte. Virtuelle private Netzwerke auf der Basis von IPsec verwenden als IP Protokolle das Encapsulated Security Payload Protokoll (ESP) und das Authentication Header Protokoll (AH). Dies sind die IP Protokolle 50 und 51 respektive. Diese Protokolle verwenden jedoch keine Portnummern.

Mit Hilfe dieser Kriterien können Regeln definiert werden, die zum Beispiel nur Pakete zu einem Webserver durchlassen, wenn sie an seinen Port 80 gerichtet sind. Da ein Paketfilter normalerweise nicht in der Lage ist, den Inhalt der Pakete zu betrachten, kann er jedoch nicht feststellen, ob diese Pakete tatsächlich eine HTTP-Anfrage enthalten und ob das HTTP-Protokoll fehlerfrei verwendet wird. Der Paketfilter arbeitet meist im Kernel des Betriebssystems auf den Schichten 3 und 4 des OSI Modells. Er hat normalerweise keinerlei Zugriff auf die Applikationsdaten. Die zu filternden Pakete müssen nicht an eine Applikation im Userspace weitergegeben werden. Dadurch kann der Paketfilter sehr schnell arbeiten.

Es existieren zwei verschiedene Varianten eines Paketfilters: einfache zustandslose Paketfilter und zustandorientierte Paketfilter, sogenannte »Stateful Packetfilter«.

Ein zustandsloser Paketfilter (zum Beispiel ipchains[1]) ist in der Lage einzelne Pakete zu filtern. Er ist jedoch nicht in der Lage einen Zusammenhang zwischen verschiedenen Paketen herzustellen. Bei einem Paket, welches den Paketfilter von außen erreicht, ist er nicht in der Lage festzustellen, ob dieses Paket Teil einer bereits aufgebauten Verbindung ist oder eine neue Verbindung öffnet. Ein zustandsloser Paketfilter muss daher alle theoretisch möglichen Antwortpakete von außen erlauben, um eine reibungslose Kommunikation zu unterstützen.

Ein zustandsorientierter Paketfilter (zum Beispiel iptables[2]) prüft bei jeder neuen Verbindung, ob sie entsprechend den Regeln erlaubt ist. Er erzeugt dann einen Eintrag in seiner Zustandstabelle. Anschließend können weitere Pakete dieser Verbindung automatisch zugelassen werden. Es müssen nicht mehr alle denkbar möglichen Antwortpakete erlaubt werden. Der Paketfilter erlaubt nur noch diejenigen Pakete, die zu vorher aufgebauten und entsprechend den Regeln authentifizierten Verbindungen gehören. Dies erhöht die Sicherheit des Paketfilters. Damit ist ein zustandsorientierter Paketfilter gewissermaßen ein Verbindungsfilter.

Viele dieser Paketfilter unterstützen die »Stateful Inspection«. Hierbei betrachtet der Paketfilter auch den Inhalt einiger Pakete für die Verwaltung seiner Regeln. Dieses Verhalten wird benötigt, da einige Protokolle vom üblichen Standard einer IP Verbindung zwischen einem Client und Server abweichen. Normalerweise kontaktiert der Client von einem hohen Port (Port ≥ 1024) den Server auf einem privilegierten Port (Port < 1024). Über diese Verbindung werden *alle* Informationen ausgetauscht.

Der bekannteste Vertreter der Protokolle, die sich nicht an diesen Standard halten ist FTP. Der Client verbindet sich von einem hohen Port auf dem Port 21 (ftp control port) auf dem Server. Diese Verbindung wird verwendet um die Informationen zur Anmeldung und die weiteren Befehle zu übertragen.

---

1. Der Paketfilter ipchains ist auch in der Lage Pakete zu maskieren. Hierbei wird die Absender IP Adresse in den Paketen ausgetauscht. Damit die Antwortpakete später den korrekten Absendern und Ports zugeordnet werden können, muss ipchains eine Zustandstabelle pflegen und stellt in dem Moment eine Art zustandsorientierten Paketfilter dar. Dies trifft jedoch nur für die maskierten Verbindungen zu!
2. Der Paketfilter iptables ist nur dann ein zustandsorientierter Paketfilter, wenn das *ip_conntrack.o*-Modul geladen wurde. Dies erfolgt automatisch, wenn der Paketfilter ein Network Address Translation (NAT) durchführt. Zusätzlich müssen jedoch diese Funktionalitäten auch von den Regeln genutzt werden. Für die »Stateful Inspection« müssen ebenfalls weitere Module geladen werden.

Sobald der Server Daten auf den Client übertragen muss (Verzeichnisinhalt oder Datei), öffnet der Server eine Verbindung von Port 20 (ftp data port) auf einen anderen hohen Port des Clients. Dies bezeichnet man als aktives FTP, da der Server eine aktive Rolle einnimmt (siehe Abbildung 1.2). Der zu verwendende hohe Port wird zuvor von dem Client an den Server in einem sogenannten PORT Kommando übertragen. Stateful Inspection bedeutet, dass die Firewall in der Lage ist, das PORT Kommando zu erkennen und anschließend spezifisch die aktive FTP Verbindung zu erlauben. Eine zustandslose Firewall kann diesen Zusammenhang nicht herstellen und muss daher grundsätzlich Pakete von jedem beliebigen Rechner und Port 20 auf jeden hohen Port eines Clients zulassen, um aktives FTP zu unterstützen.

Die Stateful Inspection stellt die einzige Ausnahme dar, bei der ein Paketfilter intelligent auf den Inhalt des Paketes zugreift. Dies kann auch für die Applikationsprotokolle Internet Relay Chat (IRC), Point to Point Tunneling Protocol (PPTP), H.323, ICMP und andere erfolgen. Ansonsten betrachtet jedoch ein Paketfilter nur die Header der Pakete. Er ist mehr oder weniger ein intelligenter Router! Das bedeutet, dass die Verbindung bei einem Paketfilter (im Gegensatz zum Proxy) zwischen dem echten Client und dem echten Server aufgebaut wird.

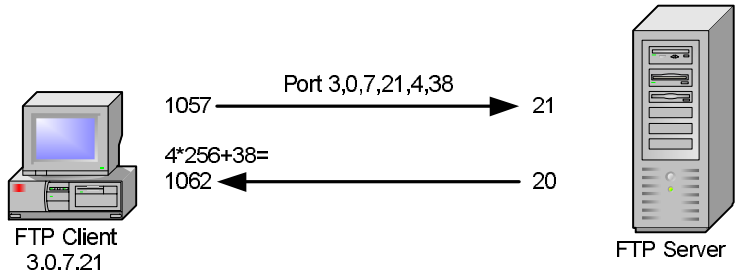


Abbildung 1.2 Aktive FTP Verbindung

Proxy-Firewall

Ein Filter auf der Schichten des 5 bis 7 des OSI Modells (Proxy) betrachtet die Pakete nicht. Ein Proxy arbeitet im Userspace und das Betriebssystem bereitet ihm die Pakete zu einem Datenstrom auf. Diesen Datenstrom kann nun der Proxy verarbeiten. Dabei ist er theoretisch in der Lage auf sämtliche Informationen des Datenstroms zuzugreifen, diese zu untersuchen und zu verändern.

Der Proxy fungiert hierbei als ein Mann in der Mitte (Man-in-the-Middle). Der Proxy nimmt an Stelle des Servers die Anfragen des Clients als Datenstrom entgegen. Er verarbeitet und filtert diese Anfrage und leitet sie anschließend als Client an den echten Server weiter. Dieser sendet seine Antwort an den Proxy, der erneut in der Lage ist, die Daten zu analysieren und zu filtern. Schließlich wird der Proxy die Daten an den echten Client zustellen.

Ein Proxy erlaubt nicht den Aufbau von Netzwerkverbindungen zwischen dem Client und dem Server. In Wirklichkeit werden zwei Netzwerkverbindungen aufgebaut: Client-Proxy und Proxy-Server. Es existiert kein Paketaustausch zwischen dem Client und dem Server!

Das größte Problem bei der Implementierung einer Firewall rein auf der Basis von Proxies stellen die Applikationsprotokolle selbst dar. Diese weisen keine gemeinsame Grundlage auf. Sie unterscheiden sich in ihren Befehlen, ihrer Syntax, Sprache und Funktionalität sehr stark. Daher ist es erforderlich für jedes Applikationsprotokoll einen eigenen Proxy zu entwickeln, der in der Lage ist dieses Protokoll zu verstehen, zu filtern und weiterzuleiten. So stellt das HTTP Applikationsprotokoll andere Anforderungen an einen Proxy als das POP3 E-Mail Protokoll.

Kommerzielle Firewalllösungen auf der Basis eines Proxy als auch Open Source Lösungen sind daher nicht in der Lage sämtliche Protokolle nativ zu unterstützen. In solchen Fällen kommen häufig weitere generische Proxies zum Einsatz, die lediglich die Verbindung auf einem Port entgegennehmen und eine neue Verbindung aufbauen. Hierbei ist aber keine Analyse oder Filterung des Datenstroms möglich. Diese Proxies werden auch als Circuit Relay oder Plug Proxy bezeichnet.

Ein Proxy hat durch seine Sicht auf den Datenstrom wesentlich mehr Möglichkeiten als ein einfacher Paketfilter. Dies soll am Beispiel eines HTTP-Proxies für den WWW Zugriff beschrieben werden.

- Der Proxy kann in Abhängigkeit der URL filtern. Ein Paketfilter sieht lediglich die IP Adressen der Kommunikationspartner. Heute werden häufig viele verschiedene Websites auf einem Rechner gehostet. Ein Paketfilter ist nicht in der Lage, zwischen diesen Websites oder zwischen verschiedenen Bereichen eine Site zu unterscheiden.
- Der Proxy kann in Abhängigkeit des Inhaltes der Datei filtern. Ein Proxy erkennt den Beginn und das Ende der Dateien. Dadurch kann er den Dateityp erkennen und überprüfen und den Inhalt auf bestimmte Eigen-

schaften oder Viren testen. Bei einem Bild kann zum Beispiel geprüft werden, ob es sich tatsächlich um ein Bild handelt oder ob es doch eine ausführbare Datei ist.

- Ein Proxy kann den Dateiinhalt verändern. Dies ist zum Beispiel sinnvoll bei aktiven Inhalten von Webseiten. Ein Proxy kann JavaScript Inhalte filtern und so modifizieren, dass sie vom Client nicht ausgeführt werden.
- Ein Proxy kann eine Authentifizierung eines Benutzers vor dem Zugriff auf die Webseite verlangen.

Dies sind Fähigkeiten, die ein normaler Paketfilter nicht zur Verfügung stellen kann. Der Proxy benötigt jedoch auf Grund der fortgeschrittenen Möglichkeiten wesentlich mehr Ressourcen als ein Paketfilter. Speziell ein Virenscan ist sehr zeitaufwendig und ermöglicht teilweise auch Denial-of-Service Angriffe.[3]

Diese Fähigkeiten stehen deshalb auch nicht bei allen Proxies zur Verfügung. Besonders der generische Proxy ist nicht in der Lage derartige Filterfunktionen zur Verfügung zu stellen. In vielen Umgebungen ist die Implementierung fortgeschrittener Filterfunktionen durch einen Proxy nicht möglich, da die Anforderungen an die Bandbreite der Netzwerkverbindung nur von einem Paketfilter erfüllt werden können.

Zusammenfassung

Eine Firewall ist also in der Lage die Kommunikation einzuschränken und nur in einer bestimmten Richtung bestimmte Inhalte zu erlauben. Dennoch kann eine Firewall nur im Rahmen der Richtlinien ihre Filterfunktionen wahrnehmen. Erlaubt eine Firewall den Zugriff auf JavaScript Inhalte einzuschränken, so besteht meist nicht die Möglichkeit zwischen gutartigem und bösartigem JavaScript zu unterscheiden. Ähnliche Einschränkungen gelten für Java und andere aktive Inhalte.

Eine Firewall ist nicht in der Lage die folgenden Punkte zu garantieren:

- **Integrität der übertragenen Daten** Eine Firewall kann nicht erkennen, ob die Daten bei ihrer Übertragung verändert oder ausgetauscht wurden.
- **Vertraulichkeit der übertragenen Daten** Eine Firewall ist nicht in der Lage einen verschlüsselten Kanal aufzubauen, um die Vertraulichkeit der übertragenen Daten sicherzustellen.

Dies sind Funktionen, die von einem VPN bereitgestellt werden.

3. 42.zip führt einen DoS Angriff gegen Mailserver mit Virusscanner durch. Diese etwa 42 kByte große Datei erzeugt beim Auspacken 1.048.576 Dateien mit einer Gesamtgröße von etwa 4 PentaByte (4.503.599.626.321.920 Byte).

1.2.3 Schutz durch ein VPN

Ein VPN bietet Authentifizierung, Vertraulichkeit und Schutz der Integrität der übertragenen Informationen. Diese Punkte sollen im weiteren genauer betrachtet werden.

Authentifizierung

Die Authentifizierung ist ein sehr wichtiger Bestandteil beim Aufbau eines virtuellen privaten Netzwerkes. Eine erfolgreiche Authentifizierung ist die Voraussetzung für den Aufbau einer anschließenden verschlüsselten Verbindung. Wird die Authentifizierung übersprungen so besteht die Gefahr eines sogenannten Man-in-the-Middle-Angriffes (s.u.).

Für eine Authentifizierung können drei unterschiedliche Faktoren einzeln oder in Kombination genutzt werden:

- **Wissen**: zum Beispiel ein Kennwort. Die Authentifizierung kann erfolgen, da der Benutzer etwas weiß.
- **Besitz**: zum Beispiel eine Smartcard. Die Authentifizierung kann erfolgen, da der Benutzer eine Smartcard besitzt. Dieser Besitz zeichnet ihn als korrekten Benutzer aus.
- **Person**: zum Beispiel ein Fingerabdruck. Die Authentifizierung erfolgt biometrisch und testet die Person direkt. Die Identität der Person wird so eindeutig erkannt.

Häufig werden diese Verfahren in Kombination eingesetzt. So sind Smartcards meist zusätzlich mit einem Kennwort geschützt. Die biometrischen Verfahren haben leider noch nicht eine Reife erlangt, die ihren Einsatz im Consumerbereich rechtfertigen würde.[4]

Im Folgenden soll nun die Wichtigkeit der Authentifizierung eines Kommunikationspartners an zwei Beispielen verdeutlicht werden.

Stellen Sie sich vor Sie möchten ein gebrauchtes Auto privat für 5000 Euro erwerben. Dann werden Sie sicherlich nicht mit dem Verkäufer lediglich die Schlüssel gegen den Geldbetrag tauschen. Sie werden zusätzlich eine Authentifizierung verlangen, dass das Fahrzeug auch tatsächlich dem Verkäufer ge-

4. Siehe c't 11/2002, S. 114: Biometrie.

hört. Diese Authentifizierung erfolgt zum Beispiel, indem der Verkäufer Ihnen sowohl den Fahrzeugbrief als auch seinen Personalausweis vorzeigt. Erst dann können Sie sicher sein, dass er das Recht hat das Fahrzeug zu verkaufen, denn er besitzt den Brief. Und Sie wissen, dass er tatsächlich derjenige ist, der er vorgibt zu sein.

Stellen Sie sich nun vor, dass der Verkäufer Ihnen einen italienischen Fahrzeugbrief zeigt und selbst über einen spanischen Personalausweis verfügt. Sie werden sicherlich nicht leicht bereit sein, ihm das Fahrzeug abzukaufen, da Sie zunächst nicht in der Lage sind, die Validität seiner Dokumente zu überprüfen. Im Falle der deutschen Dokumente stellt das jedoch kein Problem dar, da das Layout eines Personalausweises und eines Fahrzeugbriefes grundsätzlich bekannt sind.

Ein ähnliches Problem tritt auf, wenn Sie fünf Tage vor Weihnachten feststellen, dass Ihnen noch ein Geschenk fehlt. Sie haben leider keine Zeit mehr, um lange durch Geschäfte zu streifen und nach einem Geschenk zu suchen. Sie erinnern sich, dass Online Shops wie zum Beispiel Amazon.de eine Versendung bis Weihnachten noch garantieren und suchen dort entsprechende Geschenke aus. Nachdem Sie sämtliche Geschenke in Ihrem virtuellen Einkaufskorb gesammelt haben, gehen Sie zur virtuellen Kasse. Hier stellt Amazon.de fest, dass Sie bisher noch nicht Kunde sind und bittet Sie um die Eingabe Ihrer Konto- oder Kreditkarteninformationen.

Nun stehen Sie vor einem Problem. Zum einen möchten Sie ihre Informationen verschlüsselt übertragen. Hierzu müssen Sie einen verschlüsselten Tunnel aufbauen. Dies ist seit Diffie Hellman den nach ihnen benannten Schlüsselaustausch erfunden haben, (siehe Abschnitt 2.5.6, »Diffie Hellman«) sehr einfach mit der Secure Socket Layer (SSL) des Hypertext Transport Protokolls (HTTP) möglich. Ihnen fehlt jedoch zuvor eine Authentifizierung von Amazon.de. Nur weil die Website überzeugend aussieht, bedeutet das ja noch lange nicht, dass Sie tatsächlich auf der Website von Amazon gelandet sind. Es könnte ja sein, dass ein Angreifer unsere Anfrage an Amazon.de abgefangen hat und auf seinen Rechner umgeleitet hat (siehe Exkurs DNS-Spoofing). Bei dem Autokauf konnten Sie sich den Personalausweis des Verkäufers zeigen lassen. Ganz so einfach ist das in diesem Fall nicht möglich. Ein Man-in-the-Middle Angriff ist möglich (siehe Abbildung 1.3).

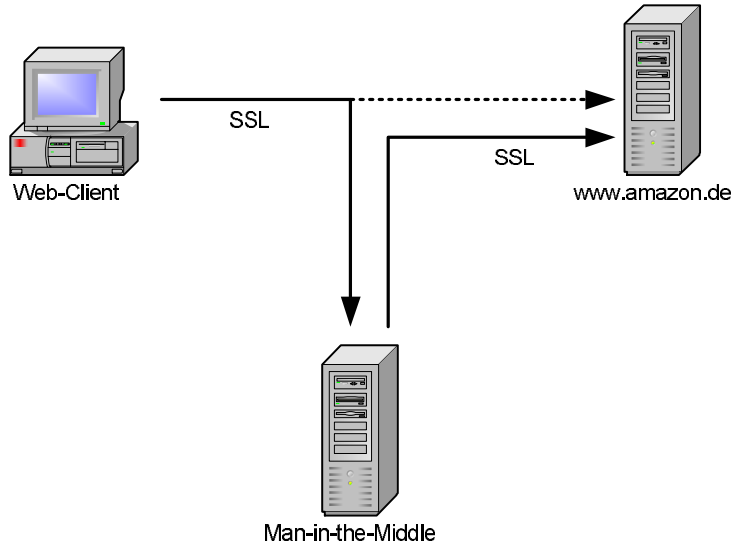


Abbildung 1.3 Ein Man-in-the-Middle Angriff

Exkurs: DNS-Spoofing

Computer im Internet kommunizieren miteinander über ihre IP Adresse. Dies ist eine Nummer, die aus vier Bytes besteht. Zur einfachen Darstellung werden die Bytes üblicherweise durch Punkte voneinander getrennt, zum Beispiel 10.5.171.253. Da es sehr schwer ist sich diese IP Adressen zu merken, erhalten Computer zusätzlich einen Namen. Für die Auflösung des Namens in die entsprechende IP Adresse und umgekehrt haben sich im Laufe der Zeit verschiedene Systeme etabliert, von denen das Domain Name System (DNS) das heute am meisten verwendete System ist. Dieses System ist verantwortlich dafür, dass ein Rechner einen DNS Namen in die entsprechende IP Adresse auflösen und für eine IP Adresse auch den entsprechenden Namen ermitteln kann.

Wenn ein Benutzer in seinem Browser die Adresse *http://www.amazon.de* eingibt, so wird dieser Browser zunächst eine DNS Anfrage stellen um die IP Adresse des entsprechenden Rechners in Erfahrung zu bringen. Erhält er hierbei eine falsche IP Adresse, so spricht man von DNS-Spoofing. So besteht zum Beispiel die Möglichkeit, dass ein Angreifer einen Benutzer auf eine andere Website umlenkt und ihm falsche Informationen unterschiebt.

Es existieren grundsätzlich zwei Methoden, mit denen das DNS-Spoofing erfolgen kann:

1. DNS Server kennen nicht alle DNS Namen des Internets. Daher müssen Sie häufig bei anderen DNS Servern nachfragen um die Auflösung eines DNS Namens in eine IP Adresse zu gewährleisten. Um nicht für denselben DNS Namen nach kurzer Zeit eine neue Anfrage zu starten, cachen die DNS Server diese, von anderen DNS Servern gelieferten, Ergebnisse. Die Dauer der Zwischenspeicherung bestimmt der liefernde DNS Server. Grundsätzlich erlaubt es das DNS Protokoll dem antwortenden DNS Server zusätzliche Informationen, die nicht ursprünglich angefragt wurden, mitzuliefern. Diese werden von dem fragenden DNS Server dann häufig auch gecached. Das wird als DNS Cache Poisoning (siehe Abbildung 1.4) bezeichnet. Moderne DNS Server bieten üblicherweise Funktionen um dies zu unterbinden.

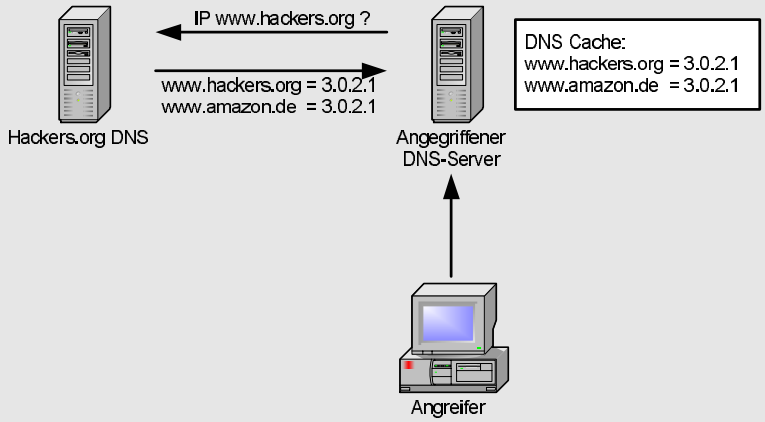


Abbildung 1.4 DNS Cache Poisoning

Solch ein Angriff wurde in dem New Yorker Wahlkampf 1999 auf die Website von Hillary Clinton angewendet, um Zugriffe auf Ihre Website *http://www.hillary2000.org* umzulenken auf die Website *http://www.hillaryno.org*. Genauso kann der Angriff genutzt werden, um Kunden von Amazon.de auf eine andere Website umzuleiten.

2. Bei der zweiten Variante werden direkt die Anfragen des Browsers an den DNS Server oder des DNS Servers an weitere DNS Server aufgefangen und durch ein Programm des Angreifers direkt beantwortet. Da dieses Programm wahrscheinlich wesentlich schneller die Anfrage beantworten kann als ein DNS Server, der zunächst in seiner Datenbank suchen muss, wird diese Antwort als korrekte Antwort akzeptiert. So kann ein Angreifer also warten, bis er eine entsprechende Anfrage im Netz erkennt und dann sein Opfer gezielt auf die falsche IP Adresse lenken.

Um dies im Zusammenhang mit einer SSL verschlüsselten Verbindung ausnutzen zu können wird noch eine Anwendung benötigt, die auf dem Rechner des Angreifers läuft, den verschlüsselten Tunnel aufbaut und dem Opfer den Eindruck vermittelt dies sei der korrekte Rechner. Dug Song hat derartige Werkzeuge bereits vor mehr als zwei Jahren öffentlich vorgestellt. Hierbei handelt es sich um die Werkzeuge `webmitm` und `dnsspoof` seines Programmpaketes `dsniff`.

Public Key Kryptografie bietet hier Hilfe. Eine genauere Betrachtung dieser Methode erfolgt in den späteren Kapiteln. Bei der Public Key Kryptografie, erzeugt ein Benutzer für sich immer zwei Schlüssel. Der eine Schlüssel wird als privater Schlüssel bezeichnet und stellt die Identität des Benutzers dar. Jeder, der über diesen Schlüssel verfügen kann, kann sich als der entsprechende Benutzer ausgeben. Häufig wird dieser Schlüssel zum Schutz noch mit einem Kennwort verschlüsselt und auf einer Smartcard gespeichert. Der zweite Schlüssel wird als öffentlicher Schlüssel bezeichnet. Dieser Schlüssel kann frei abgegeben werden.

Die Besonderheit der Public Key Kryptografie liegt nun in der Beziehung der beiden Schlüssel. Eine Nachricht, die mit dem privaten Schlüssel verschlüsselt wurde, kann *nur* mit dem entsprechenden öffentlichen Schlüssel entschlüsselt werden. Dies gilt dementsprechend auch in die andere Richtung. Eine mit dem öffentlichen Schlüssel verschlüsselte Nachricht kann nur mit dem privaten Schlüssel entschlüsselt werden (siehe Abbildung 1.5).

Dieses Verfahren kann nun zur Authentifizierung von Amazon.de genutzt werden. Für den Webserver von Amazon.de wird ein derartiges Schlüsselpaar erzeugt. Der öffentliche Schlüssel wird zum Kunden übertragen. Anschließend kann der Kunde bevor er sensitive Daten an Amazon überträgt die Authentifizierung von Amazon verlangen. Hierzu kann er eine große zufällige Zahl an Amazon übermitteln und Amazon.de auffordern, diese Zahl mit ihrem privaten Schlüssel zu verschlüsseln. Amazon.de sendet diese verschlüsselte Herausforderung (Challenge) an den Kunden zurück, der sie mit dem öffentlichen Schlüssel entschlüsseln und mit der Original-Zahl vergleichen kann.

Kommen wir zurück zum Problem: Sie wollen wenige Tage vor Weihnachten noch die Geschenke einkaufen. Wie erhalten Sie den öffentlichen Schlüssel von Amazon.de? Ganz einfach. Amazon.de sendet Ihnen diesen Schlüssel über das Internet. Dies erfolgt noch nicht verschlüsselt. Da es sich um den öffentlichen Schlüssel handelt ist das auch nicht erforderlich. Woher wissen Sie nun, dass der Schlüssel tatsächlich von Amazon.de ist und nicht von einem

Man-in-the-Middle gesendet wurde? Dies stellt nun das zentrale Problem dar.

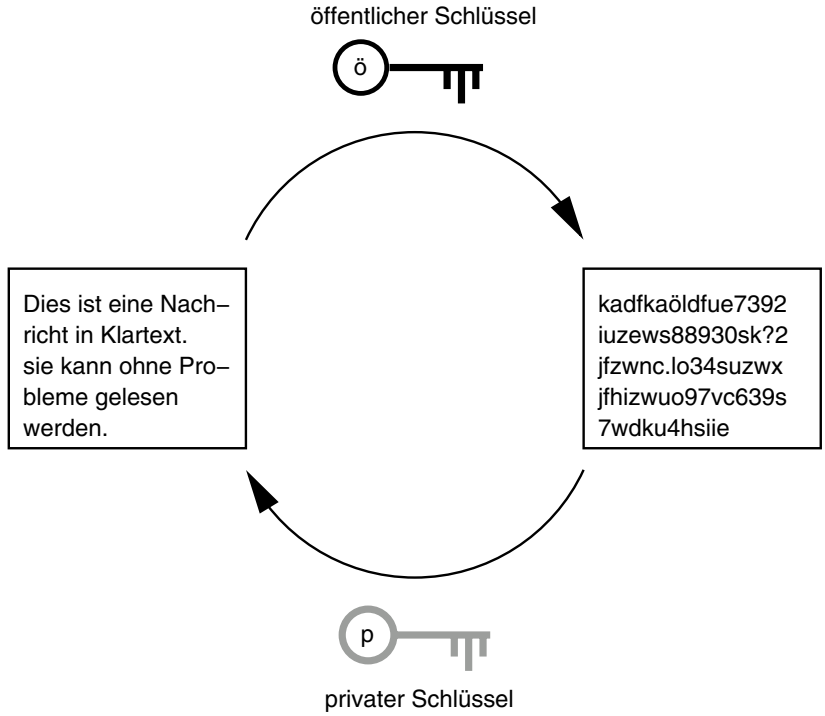


Abbildung 1.5 Ver- und Entschlüsselung mit dem Public Key Verfahren (vereinfacht)

Im Falle des Fahrzeugkaufs war es einfach, wenn der Verkäufer einen deutschen Personalausweis besaß. Dieser Personalausweis zertifizierte ihn als deutschen Staatsbürger und konnte sehr einfach überprüft werden, da das Layout und die Stempel allgemein bekannt sind. Im Grunde vertraut der Käufer der ausstellenden Stelle des Personalausweises, dass sie die Identität der Person geprüft hat. Es wird also hier eine dritte Zertifizierungstelle (Certificate Authority, CA) genutzt.

Das Verfahren wurde auf das Internet übertragen. Hierzu hat Amazon.de den eigenen öffentlichen Schlüssel mit einer »Kopie des eigenen Personalausweises« an eine Zertifzierungsstelle gesendet. Die Zertifizierungsstelle bestätigt die Echtheit des Schlüssels, in dem sie ihn mit ihrem eigenen privaten Schlüssel signiert. Diese Signatur kann nun mit dem öffentlichen Schlüssel der Zertifizierungstelle validiert werden. Ist die Signatur echt, dann ist auch der öffentliche Schlüssel von Amazon.de echt und der Browser des

Amazon.de Kunden kann den Challenge an Amazon.de senden. Wenn Amazon.de den Challenge richtig verschlüsselt, handelt es sich tatsächlich um den Webserver von Amazon.de.

Wie erhält nun der Amazon.de Kunde den öffentlichen Schlüssel der Zertifizierungsstelle um deren Signatur zu prüfen? Hier besteht ja dasselbe Problem wie zuvor mit dem öffentlichen Schlüssel von Amazon.de. Der Trick liegt in der Tatsache, dass die verwendeten Browser bereits sämtliche öffentlichen Schlüssel der anerkannten Zertifizierungsstellen enthalten. So können sie Zertifikate, die von diesen CAs unterzeichnet wurden, validieren.

Dieses Verfahren der Authentifizierung von Kommunikationspartnern mit Zertifikaten wird in späteren Kapiteln noch genauer erläutert. Im Grunde arbeiten die meisten guten Authentifizierungssysteme auf diese oder ähnliche Weise. Jedoch sollte der Stellenwert der Authentifizierung deutlich geworden sein. Ohne eine vorherige Authentifizierung der Kommunikationspartner kann keine Datensicherheit garantiert werden. Die Authentifizierung garantiert den Ursprung der Daten und stellt damit sicher, dass die Daten von dem gewünschten Kommunikationspartner stammen.

Vertraulichkeit

Die Garantie der Vertraulichkeit der übertragenen Daten ist eine weiterer wichtiger Aspekt eines virtuellen privaten Netzwerkes. Diese Vertraulichkeit kann technisch durch einen Provider in Form eines ATM Netzwerkes gewährleistet oder durch eine sichere Verschlüsselung der Daten während des Transports garantiert werden. Die Realisierung durch einen Provider in Form eines ATM Netzwerkes ist nicht Thema dieses Buches und soll daher hier vernachlässigt werden. Wenn heute von einem Software VPN Produkt gesprochen wird, so garantiert dies die Vertraulichkeit durch eine Verschlüsselung (meist mit IPsec) der übertragenen Informationen.

Die heute eingesetzte Verschlüsselungsverfahren werden in symmetrische und asymmetrische Verfahren unterschieden. In beiden Fällen sind die mathematischen Verfahren bekannt und werden dauernd auf Herz und Nieren geprüft.

Bei den symmetrischen Verfahren wird für die Ver- und Entschlüsselung der identische Schlüssel eingesetzt. Bei den asymmetrischen Verfahren handelt es sich um Public Key Algorithmen, die mit zwei Schlüsseln arbeiten. Dabei wird die Nachricht mit einem Schlüssel verschlüsselt und kann nur mit dem entsprechenden Pendant entschlüsselt werden (vereinfacht, siehe Kapitel 2, »Kryptografie«.

Die heute im Einsatz befindlichen symmetrischen Verfahren wie DES, 3DES, AES, Blowfish, Twofish, CAST, RC4, RC5, weisen bei richtiger Anwendung keine wesentlichen Sicherheitslücken auf, die es ermöglichen würden, aus einem verschlüsselten Text auf den Klartext oder den verwendeten Schlüssel zu schließen. Ein Angriff ist lediglich durch einen sogenannten Brute Force Angriff möglich. Hierbei muss der Angreifer sämtliche möglichen Schlüssel ausprobieren. Dies dauert in Abhängigkeit des verwendeten Algorithmus, der verwendeten Schlüssellänge und der zur Verfügung stehenden Hardware unterschiedlich lange. So errechnete das Projekt distributed.net (*http://www.distributed.net*), dass sie bei einer dauerhaften Rechenleistung von 45,998 2GHz AMD Athlon XP Rechnern 790 Tage benötigt hätten um sämtliche möglichen RC5-64 Schlüssel auf einem verschlüsselten Text anzuwenden. Diese für das Knacken aufzuwendene Zeit lässt sich sehr einfach durch einen längeren Schlüssel exponentiell verlängern. So erfordert ein 1 Bit längerer Schlüssel den doppelten und ein 2 Bit längerer Schlüssel bereits den vierfachen Aufwand. Heutzutage übliche Längen eines symmetrischen Schlüssels sind 40, 56, 64, 128, 168 und 256 Bit. Schlüssellängen kleiner als 128 Bit werden jedoch als nicht sicher eingestuft.

Die symmetrischen Verfahren haben jedoch den Nachteil, dass der verwendete Schlüssel beiden Kommunikationspartnern bekannt sein muss. Das bedeutet, dass der symmetrische Schlüssel vor dem Aufbau der Verbindung auf geheimem Weg ausgetauscht werden muss. Erhalten dritte Personen Zugang zu diesem Schlüssel, so sind sie in der Lage die Verbindung mitzulesen.

Diesen Nachteil weisen asymmetrische Public Key Verfahren (RSA, DSA, ElGamal, etc.) nicht auf. Hierbei erzeugt jeder Kommunikationspartner ein Schlüsselpaar aus öffentlichem und privatem Schlüssel. Anschließend werden die öffentlichen Schlüssel ausgetauscht und können zur Verschlüsselung von Nachrichten genutzt werden. Da eine Nachricht, die mit einem öffentlichen Schlüssel verschlüsselt wurde, nur mit dem privaten Schlüssel gelesen werden kann, können die so erzeugten Mitteilungen nur von der gewünschten Person gelesen werden.

Damit die asymmetrischen Verfahren jedoch als sicher gelten können sind wesentlich längere Schlüssel erforderlich. Übliche asymmetrische Schlüssellängen sind 512, 768, 1024, 2048 und 4096 Bit. Schlüssellängen kleiner als 1024 können nicht mehr als sicher eingestuft werden. Diese Schlüssellängen führen jedoch dazu, dass eine asymmetrische Verschlüsselung von rechentechnischer Sicht aufwändiger ist als eine symmetrische Verschlüsselung. Daher werden üblicherweise beide Verfahren gemeinsam in einem sogenannten Hybridverfahren eingesetzt. Dabei wird die Nachricht mit einem zufälligen symmetrischen Schlüssel verschlüsselt und dieser mit einem öffentlichen

Schlüssel verschlüsselt und angehängt. Nur der Besitzer des entsprechenden privaten Schlüssels kann den symmetrischen Schlüssel und damit die ganze Nachricht entschlüsseln.

Integrität

Schließlich ist ein VPN auch in der Lage die Integrität der übertragenen Daten zu sichern. Dies ist erforderlich, damit die übertragenen Daten nicht verfälscht oder zusätzliche Daten injiziert werden können.

Hierfür werden üblicherweise kryptografische Prüfsummen verwendet. Diese haben eine ähnliche Bedeutung wie zum Beispiel die Quersumme. Wenn zwei Personen eine Zahl austauschen und sicherstellen möchten, dass bei der Übertragung kein Fehler passiert, so ermitteln sie eine Prüfsumme zum Beispiel in Form der Quersumme und übertragen diese ebenfalls (siehe Abbildung 1.6).

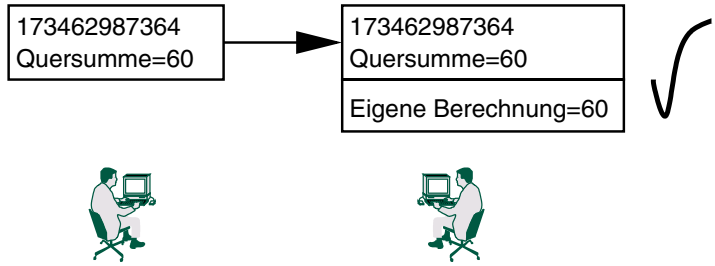


Abbildung 1.6 Schutz vor Übertragungsfehlern mit einer Quersumme

Eine einfache Prüfsumme, wie eine Quersumme, ein Paritätsbit oder die CRC32 Prüfsumme, genügt üblicherweise um zufällige Datenübertragungsfehler zu entdecken. Für diese Anwendung genügt ihre Komplexität. Wenn jedoch bewusste Veränderungen durch einen Angreifer erkannt werden sollen, so reichen diese Prüfsummen nicht mehr. Hier sind kryptografische Prüfsummen, wie MD5, SHA-1 oder RipeMD160 erforderlich. Diese Prüfsummen (Hash) verwenden derartige Algorithmen, dass es in praktikabler Zeit unmöglich ist, einen Text so zu verändern, dass er eine identische Prüfsumme ergibt.

Mit diesen Prüfsummen können nun sogenannte Authentifizierungswerte (Hash Message Authentication Codes, HMAC) erzeugt werden. Dazu erzeugt der Absender aus einem vorher ausgetauschten Geheimnis (Preshared Key, PSK) und der Nachricht eine Prüfsumme und hängt diese an. Der Emp-

fänger liest die Nachricht und erzeugt auf identische Weise die Prüfsumme. Stimmen beide Prüfsummen überein, so wurde die Nachricht nicht verfälscht und stammt aus der erwarteten Quelle. Ein Angreifer kann nicht die Nachricht so verändern, dass der Empfänger es nicht merkt, da ihm das PSK zur Erzeugung des HMAC fehlt.

1.3 Vor- und Nachteile eines VPN

Der Einsatz eines Virtuellen Privaten Netzwerkes weist sowohl Vor- als auch Nachteile auf. Zunächst scheint ein VPN nur Vorteile zu bieten. Seine Funktionen umfassen den Schutz der Vertraulichkeit, der Integrität und garantieren die Authentifizierung der Kommunikationspartner. Damit wird die sichere und vertrauliche Übertragung sämtlicher Daten im VPN gewährleistet. Dies gilt für alle transportierten Informationen. Bei einem VPN ist es nicht erforderlich, jedes Applikationsprotokoll einzeln abzusichern.

In der Vergangenheit wurden häufig einzelne Applikationsprotokolle gegriffen und mit zusätzlichen Methoden (zum Beispiel Secure Socket Layer, SSL) gesichert. Diese zusätzliche Ebene garantierte die Vertraulichkeit, Integrität und Authentifizierung der mit dem Applikationsprotokoll übertragenen Daten. Jedoch traf dies nur für die Daten zu, die mit dem entsprechenden Protokoll übertragen wurden. Mit SSL können HTTP, Telnet, POP, IMAP und die meisten weiteren TCP Applikationsprotokolle gesichert werden. Zusätzlich wurden aber auch komplett neue Anwendungen entwickelt, die die Verschlüsselung bereits enthielten. Die Secure Shell ist ein Beispiel für eine derartige Anwendung. Sie ersetzt die klassischen UNIX r-Dienste durch entsprechende verschlüsselnde s-Dienste.

Dennoch war für jedes Applikationsprotokoll die eigene Entwicklung einer derartigen Verschlüsselung oder eine Anpassung der Secure Socket Layer oder ihrer Weiterentwicklung, der Transport Layer Security (TLS), erforderlich. Ein VPN ist nicht auf ein Applikationsprotokoll beschränkt. Sämtliche übertragenen Daten werden unabhängig von dem verwendeten IP Protokoll verschlüsselt übertragen. Hierbei spielt es keine Rolle, ob es sich um eine TCP Verbindung oder eine UDP Verbindung handelt. Auch ICMP Pakete und selbst das Appletalk DDP Protokoll können über ein VPN übertragen werden (siehe Abbildung 1.7).

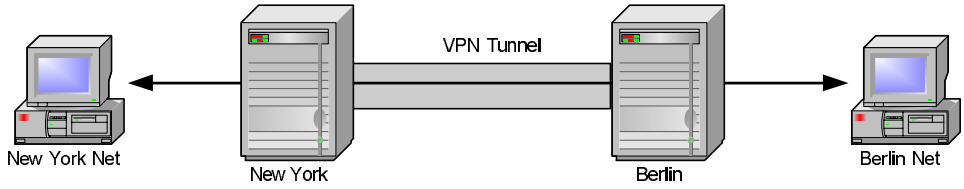


Abbildung 1.7 Ein typischer VPN Tunnel

Der Aufwand für die sichere Übertragung all dieser Protokolle hält sich in Grenzen. Es muss lediglich einmal der verschlüsselte Tunnel aufgebaut werden, anschließend können über diesen Tunnel jegliche Informationen ausgetauscht werden.

Ein derartiges VPN hat jedoch nicht nur Vorteile. Die Verschlüsselung ist nur gewährleistet zwischen den entsprechenden Maschinen, die die Verschlüsselung durchführen. Der Bereich zwischen dem Client A und dem Gateway A beziehungsweise zwischen dem Gateway B und dem Client B in der Abbildung 1.7 ist nicht verschlüsselt. Hier werden die Daten im Klartext übertragen.

Der Endbenutzer ist darüber hinaus nicht in der Lage die korrekte Verschlüsselung seiner Daten zu überprüfen. Beim Einsatz von zum Beispiel HTTPS hat der Benutzer direkt eine positive Rückmeldung der Verschlüsselung durch den Browser. Dieser kennzeichnet den erfolgreichen Aufbau einer verschlüsselten Verbindung üblicherweise mit einem geschlossenen Vorhängeschloß in der unteren Ecke. Hierbei handelt es sich also um eine Ende (Webserver) zu Ende (Webbrowser) Verschlüsselung. Bei einem VPN muss der Endbenutzer vertrauen, dass das VPN (Abbildung 1.7) seine Aufgabe korrekt erfüllt.

Möchte der Endbenutzer gar sicherstellen, dass eine E-Mail von keinem außer dem gewünschten Empfänger gelesen werden kann, so kann ein VPN dies nicht leisten. Eine derartige Verschlüsselung kann nur durch Werkzeuge wie Pretty Good Privacy (PGP) oder GNU Privacy Guard (GnuPG) erreicht werden.

1.3.1 VPNs und Firewalls

Die größten Probleme bei dem Einsatz eines VPN entstehen jedoch, wenn ein VPN gemeinsam mit einer Firewall eingesetzt werden soll. Dabei ist dass zunächst gar nicht zu verstehen. Beide Systeme versuchen die Sicherheit der Daten zu gewährleisten. Sie erhöhen die Sicherheit des Unternehmens. Bei

genauer Betrachtung stellt man jedoch fest, dass eine Firewall und ein VPN vollkommen unterschiedliche Methoden einsetzen um dieses Ziel zu erreichen.

VPN	Firewall
Verschlüsselung erlaubt keinen Einblick	Untersucht den IP Header und den Inhalt und protokolliert dies
Erlaubt üblicherweise über das VPN ungehinderten Zugang	Schränkt den Zugriff stark ein
Erweitert das Netz um weitere Rechner und Netze	Reduziert das zu schützende Netz auf einen Single Point of Defense

Tabelle 1.1 Vergleich VPN – Firewall

Die Tabelle 1.1 versucht bereits die wesentlichen Unterschiede aufzuzeigen.

Die wesentliche Tätigkeit eines VPNs liegt in der Verschlüsselung sämtlicher übertragener Informationen. Eine Firewall kann diese verschlüsselten Daten dann nicht mehr analysieren, unterscheiden oder protokollieren. Die Firewall ist sozusagen blind. Eine Firewall kann lediglich die unverschlüsselten Daten filtern.

Ein weiterer wesentlicher Bestandteil eines VPNs ist häufig der ungehinderte Zugang zum Intranet über das VPN. Der Vorstandsvorsitzende eines Unternehmens möchte von zu Hause über das VPN genauso arbeiten können, als ob er sich an seinem Arbeitsplatz in der Firma befindet. Hierzu benötigt er ungehinderten Zugang zu allen Systemen und Resourcen, die die Firma bietet, einschließlich der Datenbanken, Mailserver oder Dokumentenrepositorien. Die Aufgabe einer Firewall ist es jedoch, derartige Zugänge zu unterbinden oder auf ein Mindestmaß zu reduzieren. Auch hier kommt es zu einem Interessenkonflikt zwischen dem Firewall- und dem VPN-Administrator.

Der letzte Punkt stellt jedoch nach meiner Ansicht das größte Problem dar. Sobald eine VPN Verbindung mit einem anderen Netzwerk oder einem Außendienstmitarbeiter aufgebaut wurde, werden die entsprechenden Rechner Teil des eigenen Netzes. Die eigene Firewall ist plötzlich auch für den Schutz dieser Rechner vor Angriffen von außen verantwortlich. Diese Rechner befinden sich nun logisch hinter der Firewall. Die Sicherheit dieser Rechner definiert plötzlich die Sicherheit des gesamten Rechnernetzes. Wenn die Firewall von Netzwerk New York in Abbildung 1.7 nicht richtig konfiguriert ist und ein Einbruch in Netzwerk New York erfolgte, so kann der Angreifer direkt auf die Rechner in Netzwerk Berlin unter Umgehung der Firewall in Netzwerk Berlin zugreifen.

Dies ist natürlich nur möglich, wenn das VPN eine Umgehung der Firewall erlaubt. Leider wird in vielen Fällen das VPN derartig konfiguriert, dass es einen Zugang parallel zu Firewall erlaubt (Abbildung 1.8).

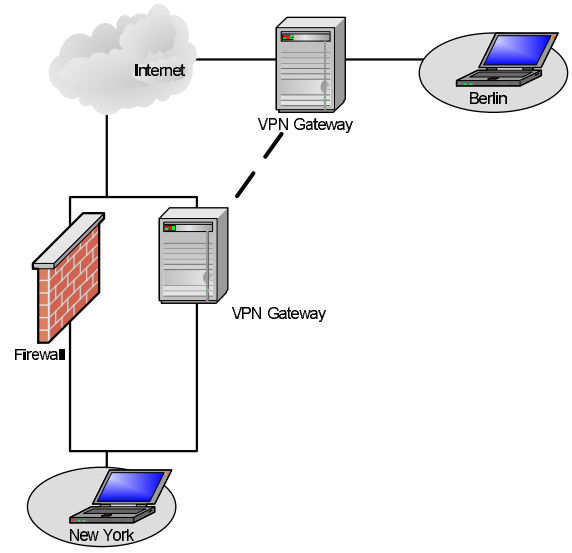


Abbildung 1.8 Firewall und VPN sind parallel zueinander aufgebaut (schlecht)

Daher sollte sich immer zwischen dem VPN Gerät und dem internen Netzwerk noch eine Firewall befinden, die den Zugriff auf das interne Netzwerk über das VPN kontrollieren und beschränken kann. Sinnvollerweise befindet sich auch vor dem VPN Gerät eine Firewall, die das VPN Gerät schützen kann (Abbildung 1.9).

So sind die über das VPN transportierten Daten und das dahinterliegende Netz optimal geschützt. Dieser logische Aufbau einer VPN/Firewall Struktur wird auch von vielen kommerziellen Anbietern geschätzt. Diese bieten häufig ein gebündeltes Produkt an, welches beide Funktionen (VPN und Firewall) bietet. Wird dieses Produkt auf einem physikalischen Gerät installiert, so kann von der logischen Funktion die in Abbildung 1.10 dargestellte Struktur realisiert werden.

Hierbei wird der normale Verkehr durch die Firewall 1 gefiltert. Parallel hierzu existiert ein VPN Gateway, welches durch zwei weitere Firewalls (2 und 3) geschützt wird. Hierbei filtert die Firewall 2 den verschlüsselten Verkehr von und nach außen und schützt die VPN Gateway Software vor Angriffen. Die Firewall 3 filtert den entschlüsselten Verkehr, der über das VPN Gateway in das interne Netz gelangt.

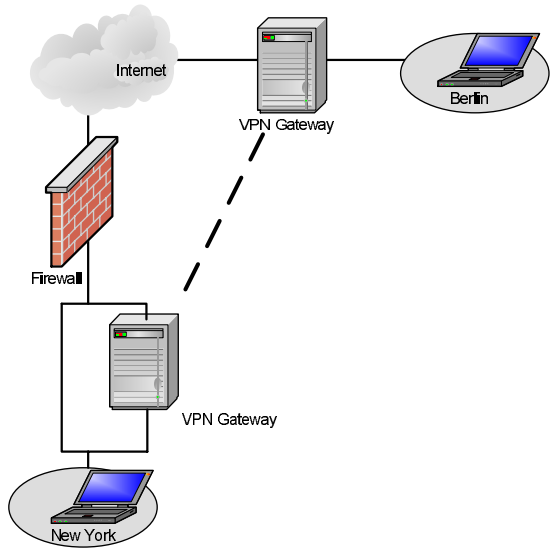


Abbildung 1.9 Firewall und VPN nacheinander geschaltet

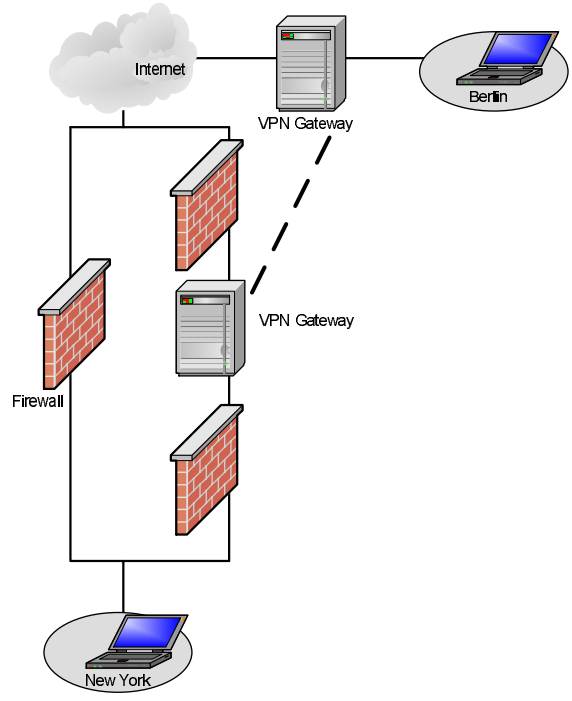


Abbildung 1.10 Ideale VPN/Firewall Struktur

Eine derartige Struktur kann mit Linux ebenfalls aufgebaut werden. Bei der Besprechung der entsprechenden Szenarien und Implementierungen werden Beispiel-Firewallregeln für Linux vorgestellt und erklärt. Dabei kann leider aus Platzgründen nicht sehr in die Tiefe gegangen werden. Wenn Sie weitere Hintergrundinformationen zum Thema Firewall und Linux benötigen, können Sie diese in dem Buch »Linux Firewalls« von Robert Ziegler, auch im Markt+Technik Verlag (ISBN 3-8272-6257-7) erschienen, nachlesen.

1.4 Open-Source und Sicherheit

Beim Einsatz von Sicherheitstechnologien kommt dem Stichwort Open-Source besondere Bedeutung zu – besonders im Zusammenhang mit Verschlüsselungstechnologien. Die Vergangenheit hat mehrfach gezeigt, dass in vielen Fällen Hersteller in Closed Source Produkten mangelhafte Verschlüsselung implementiert haben oder bewusst Hintertüren eingebaut haben, die die Verschüsselung aushebeln konnten. Dabei wählten die Hersteller häufig sichere Algorithmen, jedoch wurden die Parameter falsch gewählt.

Zunächst wird wahrscheinlich jeder normale Mensch aufschrecken, wenn er hört, dass die gesamten Verschlüsselungsalgorithmen und der gesamte Quelltext für jedermann offenliegt. Wie kann eine derartige Software eine sichere Verschlüsselung durchführen?

Wie jedoch im Kapitel zur Kryptografie noch ausgeführt werden wird, liegt die Stärke der modernen kryptografischen Algorithmen genau in diesem Detail. Der Algorithmus verwendet allgemein zugängliche Methoden und Algorithmen um einen Klartext mit einem geheimen Schlüssel in den sogenannten *ciphertext* zu wandeln. Dadurch können sämtliche Kryptoanalytiker weltweit versuchen eine Sicherheitslücke im Algorithmus zu finden. Solange dies nicht der Fall ist und das Geheimnis lediglich im Schlüssel verborgen ist, gilt der entsprechende Algorithmus als sicher (entsprechende Schlüssellängen vorausgesetzt).[5] Implementiert jedoch eine Firma einen neuen Algorithmus und hält ihn geheim, so sollte sich der Verdacht aufdrängen, dass der Erfinder nicht das entsprechende Vertrauen in seinen eigenen Algorithmus besitzt. Dann sollte von dem entsprechenden Produkt Abstand genommen werden.

TIPP

Der bekannte Kryptologe Bruce Schneier vergleicht den Kryptografiealgorithmus mit einem Safe. Selbst bei Kenntnis der Baupläne muss dieser Safe allen Angriffen wiederstehen.

5. Schwache Schlüssel werden in dem Kapitel »Kryptografie« behandelt.

Jedoch ist es nicht nur wichtig, dass der Algorithmus allgemein begutachtet und anerkannt wurde. Dies gilt auch oder umso mehr für die Implementierung des Algorithmus in Software. Hier können Programmierfehler vorliegen, die eine Sicherheitslücke erst ermöglichen. Bei Closed Source Software besteht darüberhinaus die Möglichkeit, dass der Hersteller bewusst eine Hintertür eingebaut hat, um die Verschlüsselung oder die Authentifizierung zu umgehen. Dies kommt leider relativ häufig bei proprietärer Software vor. Der Anwender kann sich aber nicht davor schützen, da es für den Benutzer sehr schwer möglich ist die Vorgänge in der Software nachzuvollziehen.

Im Folgenden sollen nun einige Beispiele gegeben werden:

- **Borland Interbase** Die Datenbank Interbase von Borland wurde nach vielen Jahren als Closed Source Projekt am 25. Juli 2000 als Open Source freigegeben. Dieser Code wurde dann vom Firebird Projekt (*http://firebird.sourceforge.net*) weitergepflegt.

  Etwa ein halbes Jahr später, am 9. Januar 2001, wird in dem Sourcecode ein hart kodiertes Login und Kennwort gefunden (*https://www.kb.cert.org/vuls/id/247371*): `politically correct`. Dies war wahrscheinlich seit 1993 bereits in der Datenbank hartkodiert vorhanden. Mit diesem Login war es möglich mit administrativen Rechten auf die Datenbank zuzugreifen. Wäre die Datenbank nicht ein Open Source Produkt geworden, wäre diese Hintertür wahrscheinlich bis heute nicht bekannt und nicht entfernt worden.

- **Crypto AG** Die Crypto AG (*http://www.crypto.ch*) ist eine Schweizer Firma, die seit über 50 Jahren kryptografische Hard- und Software herstellt und vertreibt.

  Die Crypto AG vertreibt die entsprechenden Geräte weltweit und war als Schweizer Firma nicht an die Exportbeschränkungen der Vereinigten Staaten von Amerika gebunden. Der Export von starker Kryptografie aus den USA fiel dort unter das Kriegswaffenkontrollgesetz und war verboten. So verkaufte die Crypto AG auch Geräte in den Iran. Im März 1992 wurde Hans Bühler, ein Verkaufsrepräsentant der Crypto AG im Iran von iranischen Behörden mit dem Vorwurf der Spionage festgenommen und neun Monate inhaftiert. Die Crypto AG zahlte eine Million Dollar Lösegeld und holte Hans Bühler aus dem iranischen Gefängnis. In der Heimat wurde Hans Bühler entlassen und die Crypto AG forderte das gezahlte Lösegeld von ihm zurück.

  Anschließend wurden Gerüchte laut, dass die iranischen Vorwürfe korrekt seien und die Geräte der Crypto AG tatsächlich vom deutschen Bundesnachrichtendienst und der amerikanischen National Security Agency (NSA) modifiziert wurden, so dass diese den Austausch von Informa-

tionen abhören konnten. Eine offizielle Bestätigung dieser Gerüchte erfolgte jedoch nie.

- **Windows NSA Key** Hierbei handelt es sich wahrscheinlich mehr um heiße Luft als um einen tatsächlichen National Security Agency Key in den verschiedenen Microsoft Windows Betriebssystemen (Windows 9x, NT und 2000). Jedoch existiert hier dennoch ein Problem mit dem Cryptosystem der Betriebssysteme. Das Cryptosystem besitzt *zwei* öffentliche Schlüssel, KEY und _NSAKEY, die von Microsoft verwendet werden um die kryptografischen Anwendungen so zu signieren, dass die Microsoft Betriebsysteme ihnen vertrauen. Der Name des zweiten Schlüssels war Anlass zur Spekulation ob möglicherweise die NSA Zugang zu diesem Schlüssel hätte und ebenfalls derartige Anwendungen zertifizieren könnte. Ob dies der Fall ist mag bezweifelt werden. Jedoch können diese Schlüssel von Microsoft nicht zurückgerufen werden. Beide können von Microsoft eingesetzt werden. Dies ist unüblich. Normalerweise wird für derartige Operationen nur ein Schlüsselpaar eingesetzt. Dies ermöglicht einen theoretischen Angriff auf das Cryptosystem, der vom Anwender nicht erkannt werden kann (*http://www.counterpane.com/crypto-gram-9904.html#certificates*).
- **Clipper Chip** Handelte es sich bei den bisher beschriebenen Verschlüsselungslücken um heimlich implementierte Hintertüren, die öffentlich bekannt wurden, so handelt es sich beim Clipper Chip um ein spezielles Gerät zur Verschlüsselung von privater Kommunikation. Der Clipper Chip wurde am 16. April 1993 vom Weißen Haus angekündigt. Der Clipper Chip stellt sicher, dass staatliche Behörden freien Zugriff auf die verschlüsselten Informationen erhalten und so die Kommunikation überwachen können. Als kryptografischer Algorithmus kam im Clipper Chip der Skipjack Algorithmus zum Einsatz. Für den Zugriff auf die verschlüsselten Informationen erhalten zwei US Bundesbehörden (NIST und Department of Treasury) jeweils einen Schlüssel, die zusammen die Entschlüsselung der Informationen erlauben. Dies bezeichnet man als ein Key Escrow System. So soll eine Verschlüsselung möglich sein, die es dennoch den Strafverfolgungsbehörden erlaubt auf die verschlüsselten Informationen zuzugreifen. Der Nachfolger des Clipper Chip ist der Capstone Chip.
- **PGP und Key Escrow** Die Programmierer der Software PGP haben ebenfalls angefangen seit der Version 5.5 ein Key Escrow System mit einzubinden. Diese Funktionalität steht in der Business Version der Software PGP zur Verfügung. Die Software PGP wird seit 2002 von einer neuen Firma

gepflegt. Die verfügbare PGP Enterprise Version 8.0 enthält ebenfalls derartige Funktionen zur Schlüsselwiederherstellung. Diese Funktion wird von PGP als Key Reconstruction bezeichnet. Es stellt jedoch nichts anderes als ein Key Escrow System dar.

- **Lotus und Key Escrow** Lotus Notes enthielt ebenfalls lange Jahre ein Key Escrow System. Bereits vor der Lockerung der US amerikanischen Exportbeschränkungen für starke Kryptografie wollte IBM 1996 die Lotus Notes Software mit 64 Bit Schlüsseln ausstatten. Die Exportregelungen erlaubten jedoch nur Schlüssellängen von 40 Bit. Um dennoch eine Exportgenehmigung zu erhalten, wurde zusammen mit der NSA das Workgroup Differential Verfahren entwickelt. Hierbei handelt es sich um die sogenannte Differential Workgroup Cryptography. Diese verschlüsselt die Nachricht mit 64 Bit. Anschließend werden 24 Bit des Schlüssels mit einem öffentlichen Schlüssel der NSA verschlüsselt und an die Nachricht angehängt. Die NSA kann so auf 24 Bit des originalen 64 Bit langen Schlüssels in Klartext zugreifen. Die restlichen 40 Bit des Schlüssels können mit modernen Rechnern in Bruchteilen von Sekunden errechnet werden. Dennoch war die Nachricht vor jedem weiteren Angreifer mit 64 Bit geschützt. Nur die NSA konnte 2^{24}=16.777.216 mal einfacher den Schlüssel knacken.

Open-Source Software wird offen entwickelt. Heimlich können hier keine derartigen Hintertüren eingebracht werden, ohne dass diese relativ schnell gefunden werden – vorausgesetzt das Produkt ist so interessant, dass es mehrere Programmierer pflegen!

Es soll allerdings nicht außer Acht gelassen werden, dass in einigen Fällen derartige Hintertüren bewusst gewünscht werden. Hierbei handelt es sich um:

- Gesetze, die den Strafverfolgungsbehörden die Möglichkeit verschaffen wollen auf verschlüsselte Daten zuzugreifen.
- Unternehmen, die weiterhin Zugriff auf verschlüsselte Daten eines ausgeschiedenen Mitarbeiters benötigen. Hierbei genügt es jedoch ein Key Escrow System lediglich für die dauerhaft gespeicherten Daten einzusetzen. Es ist nicht notwendig für verschlüsselte Transaktionen.

In solchen Fällen sind Key Escrow oder Key Recovery Systeme nötig. Jedoch muss in diesen Fällen die Sicherheit des entsprechenden Systems gewährleistet werden. Darüberhinaus betreffen diese Anwendungen meist nur gespeicherte Daten. Ein Key Escrow für Kommunikationen ist meist nicht erforderlich und sollte daher auch nicht eingerichtet werden.

1.5 Kommerzielle Lösungen

Es existieren eine ganze Reihe von kommerziellen Lösungen die den Aufbau eines VPNs ermöglichen. Im Folgenden sollen einige Anbieter vorgestellt werden. Die Liste erhebt keinerlei Anspruch auf Vollständigkeit und ist es sicherlich auch nicht. Die Auswahl erfolgt auf Grund eigener Erfahrungen mit den Systemen. Insbesondere soll bei den einzelnen Systemen die Kompatibilität in einer gemeinsamen heterogenen VPN Lösung mit Linux betrachtet werden.

1.5.1 Cisco

Die Produkte von Cisco umfassen unter anderem Router, Switches, Firewalls und Intrusion Detection Systeme. Hierbei sind die meisten Router und Firewalls in der Lage auch ein VPN mit Hilfe von IPsec aufzubauen.

Die Preise der Cisco Produkte richten sich nach der Größe, dem Ausbau und der verwendeten Software. Besondere Funktionalitäten wie zum Beispiel ein VPN werden bei den Routern als zusätzliche Feature Packs verkauft.

So kostet zum Beispiel ein kleiner aktueller Cisco Router der Reihe 2600 zwischen 2000 und 3000 Euro. Hinzu kommt immer noch das entsprechende IP-Feature Pack zur Verschlüsselung, das mit weiteren etwa 800 Euro zu Buche schlägt. Der Cisco VPN Concentrator 3000 kostet rund 3000 Euro.

Die Interoperabilität der Cisco Produkte mit der IPsec Implementierungen unter Linux ist sehr hoch. Erst im November 2001 wurden diese wieder auf der jährlichen IPsec Konferenz getestet (*http://www.hsc.fr/ressources/ipsec/ipsec2001/*). Die Cisco Produkte sind in der Lage sowohl mit Zertifikaten als auch mit Kennworten (Preshared Keys, PSK) einen verschlüsselten Kanal aufzubauen.

1.5.2 Checkpoint FW-1/VPN-1

Checkpoint stellt mit seiner Software Firewall-1 sicherlich eine der bekanntesten Software Firewalls her. Dieses Produkt benötigt immer zusätzlich einen Rechner mit einem entsprechenden Betriebssystem (Linux/Intel, WinNT/Intel, Solaris/Intel, Solaris/SPARC, HPUX, AIX). Die einzige Ausnahme stellen die Geräte von Nokia dar. Nokia produziert Hardwaregeräte mit einem abgespeckten BSD als Betriebssystem, auf dem dann die Firewall-1 vorinstalliert ist.

Das aktuelle Produkt NG (Next Generation) bietet sowohl Firewall als auch VPN Funktionalitäten und ist sehr modular erweiterbar. Die Lizensierung erfolgt bei Checkpoint in Abhängigkeit der zu schützenden Rechner (siehe Tabelle 1.2). Die jeweiligen VPN-Clients, die benötigt werden um die Verbindung zum Checkpoint VPN Gateway aufzubauen, sind lizenzkostenfrei.

Max. Clients	ca. EUR
25	4500
50	7000
100	11000
250	14000

Tabelle 1.2 Checkpoint VPN-1 NG Lizenzpreise

Die Interoperabilität ist im Falle der Checkpoint IPsec Implementierung nicht in allen Fällen gegeben. Es besteht die Möglichkeit Linux als Client mit Checkpoint kommunizieren zu lassen. Jedoch kann der freie Checkpoint VPN-Client nicht genutzt werden, um eine Verbindung zu einem Linux Gateway aufzubauen. Bei der Authentifizierung unterstützt Checkpoint sowohl x509-Zertifikate als auch Preshared Secret Keys (PSK). Die Verwendung von PSKs setzt sinnvollerweise den sogenannten Aggressive Mode voraus. Dieser wird von FreeS/WAN mit einem Patch unterstützt.

1.5.3 Microsoft Windows 2000 und XP

Microsoft hat mit Windows 2000 und Windows XP begonnen IPsec in seinen Betriebssystemen zu unterstützen. Dies erfolgt im Zuge der Unterstützung für IPv6, der nächsten Generation des Internet Protokolls. Die von Microsoft implementierte VPN Unterstützung setzt jedoch das IPsec Protokoll gemeinsam mit dem L2TP Protokoll ein. Dieses zusätzliche Protokoll erlaubt die Vergabe von dynamischen IP Adressen und die Authentifizierung von Benutzern. Für Linux existieren ebenfalls L2TP Dienste, jedoch ist deren Konfiguration recht aufwendig (siehe Abschnitt 9.4.2, »L2TP«).

Aber sowohl Windows 2000 als auch XP erlauben es, reine IPsec Tunnel aufzubauen, die mit Linux interoperieren können. Dies ist jedoch mit Einschränkungen verbunden (siehe Kapitel 7, »Aufbau heterogener Virtueller Privater Netze«).

1.5.4 Microsoft Windows 98/ME/NT

Die Microsoft Betriebssysteme Windows 98, Windows ME und Windows NT enthalten keine Unterstützung für das IPsec Protokoll. Hier sind normalerweise Werkzeuge von Drittherstellern (siehe SSH Sentinel und Safenet SoftPK) erforderlich. Jedoch hat Microsoft im Juni 2002 einen kostenlosen IPsec/L2TP Client veröffentlicht (*http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/l2tpclient.asp*). Dieser Client unterstützt jedoch leider nicht reine IPsec Verbindungen, sondern nur kombinierte IPsec/L2TP Tunnel (siehe Abschnitt 9.4.2, »L2TP«).

Dieser Client soll in Zukunft das unsicherere Point-To-Point-Tunneling Protokoll (PPTP) ablösen. Dies stellte bislang die einzige von Microsoft unterstützte Möglichkeit zum Aufbau eines VPN unter den genannten Betriebssystemen dar.

1.5.5 SSH Sentinel

Die Firma SSH (*http://www.ssh.com*) ist bekannt geworden durch die Vermarktung der kommerziellen Version der Secure Shell. SSH produziert jedoch inzwischen auch PKI und VPN Produkte. Der SSH Sentinel (ehemals Internet Pilot, *http://www.ssh.com/products/security/sentinel/*) ist ein IPsec VPN Client für Microsoft Windows Betriebssysteme.

Er erweitert diese Betriebsysteme um einen IPsec Stack und erlaubt die Authentifizierung mit x509 Zertifikaten und Preshared Secret Keys. Hierbei kann er in eine vorhanden Public Key Infrastructure (PKI) eingebunden werden. Die privaten Schlüssel können auf Chipkarten gespeichert werden. Zusätzlich verfügt der SSH Sentinel über eine eingebaute Personal Firewall. Der SSH Sentinel ist interoperable mit den entsprechenden Lösungen unter Linux.

Der SSH Sentinel kostet in einer Einzellizenz etwa 160 Euro und kann direkt über den SSH Online Store (*http://www.ssh.com/company/sales/store/*) bezogen werden.

1.5.6 SafeNet SoftRemote

Die SafeNet SoftRemote Software (früher SoftPK) wird von SafeNet, Inc. (ehemals IRE, *http://www.safenet-inc.com*) hergestellt. Dieser Client wird auch von einigen anderen Herstellern eingesetzt und stellt auch die Basis für den

oben erwähnten freien Windows L2TP-Client dar. Auch SoftRemote enthält eine persönliche Firewall. Laut Webpage ist SoftRemote der am häufigsten eingesetzte VPN Client weltweit.

Der Vertrieb der Software erfolgt über verschiedene Partner oder Online über die URL *http://www.safenet.biz/*. Die Safenet SoftRemote Software ist verfügbar für die Microsoft Betriebssysteme Windows 95, 98, 2000, NT 4.0, ME und XP. Zusätzlich existieren SoftRemotePDA Versionen für Pocket PC 2002 und PalmOS ≥ 3.5. Diese Software ist verfügbar für 149 Dollar für die Desktopversionen und 39 Dollar für die PDA Versionen.

SoftRemote weist keine Probleme bei einem kombinierten Einsatz mit den Linuxversionen auf.

1.5.7 OpenBSD, FreeBSD, NetBSD

Bei den Betriebsystemen OpenBSD (*http://www.openbsd.org*), FreeBSD (*http://www.freebsd.org*) und NetBSD (*http://www.netbsd.org*) handelt es sich um freie UNIX Systeme. Alle diese Betriebsysteme sind in der Lage eine IPsec Verbindung aufzubauen und weisen keine Probleme in Kombination mit Linux auf. Die von diesen Systemen verwendeten IKE-Daemonen sind inzwischen auf Linux lauffähig.

1.5.8 Weitere Produkte

Weitere VPN Produkte werden von weiteren zahlreichen Herstellern angeboten. Hier soll kurz auf die folgenden hingewiesen werden:

- **MacOSX** Das Betriebsystem MacOSX enthält einen IPsec Stack. Dieser muss jedoch mit einem Kommandozeilenwerkzeug konfiguriert werden (ähnlich Linux). Ein grafischer Client mit dem Namen *VPN Tracker* ist verfügbar bei Equinux (*http://www.equinux.com/us/products/vpntracker/index.html*).
- **PDAs** Zwei weitere Firmen, die Clients für PDAs herstellen sind Certicom (*http://www.certicom.com/products/movian/movianvpn.html*) und Funk Software (*http://www.funk.com/*). Der Movian VPN Client scheint momentan nicht interoperabel zu sein mit FreeS/WAN. Über den Funk Client existieren keine Informationen.

1.6 Verschiedene VPN Szenarien

Dieses Kapitel soll bereits einige Szenarien für den Einsatz eines Virtuellen Privaten Netzwerks vorstellen. Diese sollen sowohl als Anregung dienen als auch als Einleitung zu den späteren Kapiteln, die dann Lösungen für diese Szenarien bieten.

Die im Folgenden beschriebenen Szenarien sind sicherlich nicht vollständig und beispielhaft für jede mögliche Situation. Sie sollen jedoch die klassischen Fälle für den Einsatz eines VPN beschreiben. Im weiteren Verlauf des Buches werden diese dann wieder aufgegriffen und Lösungen präsentiert.

1.6.1 Kommunikation zwischen zwei Netzwerken

Die Kommunikation zwischen zwei Netzen ist die häufigste Anwendung für ein Virtuelles Privates Netzwerk. Hierbei werden zwei Standorte, die jeweils über eine Internetanbindung verfügen, mit einem VPN vernetzt, so dass sie vertrauliche Informationen austauschen können. Dieses Szenario wird auch als Site-to-Site VPN bezeichnet. Abbildung 1.11 stellt exemplarisch ein derartiges VPN dar. Sämtliche zwischen den beiden Gateways ausgetauschten Informationen werden für den Transport im Internet verschlüsselt und über das Internet transportiert. Die Kommunikation innerhalb der Netze erfolgt im Klartext.

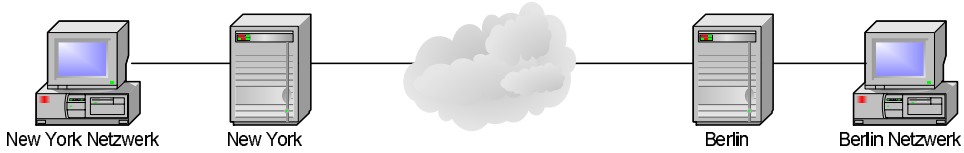


Abbildung 1.11 Site-to-Site VPN

Diese Lösung wird häufig gewählt, um zwei Filialen miteinander zu verbinden.

1.6.2 Kommunikation zwischen zwei Rechnern

Häufig soll nicht die Kommunikation zwischen zwei Netzwerken verschlüsselt und gesichert werden, sondern die Kommunikation zwischen zwei Rechnern. Dies wird auch als ein End-to-End VPN bezeichnet. Abbildung 1.12 skizziert ein derartiges VPN. Hierbei enthalten die kommunizierenden

Rechner auch bereits die VPN Funktionalität. Die Informationen verlassen die Rechner bereits verschlüsselt und müssen nicht mehr durch ein VPN Gateway verschlüsselt werden (Abbildung 1.12).

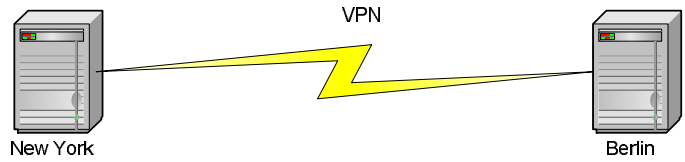


Abbildung 1.12 VPN zwischen zwei Rechnern

1.6.3 Kommunikation zwischen vielen festen Standorten

Dieses Szenario gleicht dem ersten Szenario. Lediglich die Anzahl der zu verbindenden Netzwerke ist größer zwei. Um eine größere Anzahl von Standorten mit einem VPN zu vernetzen existieren grundsätzlich zwei verschiedene mögliche Strukturen für den Aufbau eines VPNs: Stern und Netz.

Bei einer Sternstruktur bauen alle Standorte eine Verbindung zu einem zentralen VPN Gateway auf. Dessen Aufgabe ist es, die über das VPN transportierten Nachrichten, entsprechend zu den Empfängern zu routen (siehe Abbildung 1.13). Diese Struktur erlaubt einen sehr einfachen Aufbau. Es wird nur jeweils ein Tunnel pro Standort benötigt. Jedoch hat diese Struktur auch den Nachteil, dass bei Ausfall des zentralen Gateways die komplette VPN Kommunikation ausfällt. Aus diesem Grunde werden in derartigen Szenarien häufig die zentralen Gateways hochverfügbar ausgelegt.

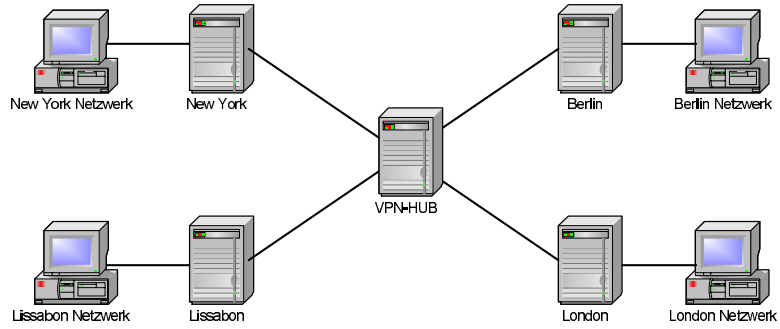


Abbildung 1.13 Sternförmiger Aufbau eines VPNs

Die netzförmige Struktur stellt wesentlich höhere Ansprüche an die Administration und Wartung des VPNs. Hierbei baut jeder Standort einen Tunnel zu jedem weiteren Standort auf. Dies gewährleistet die direkte Kommunikation und umgeht mögliche Verfügbarkeitsprobleme eines zentralen Gateways (siehe Abbildung 1.14).

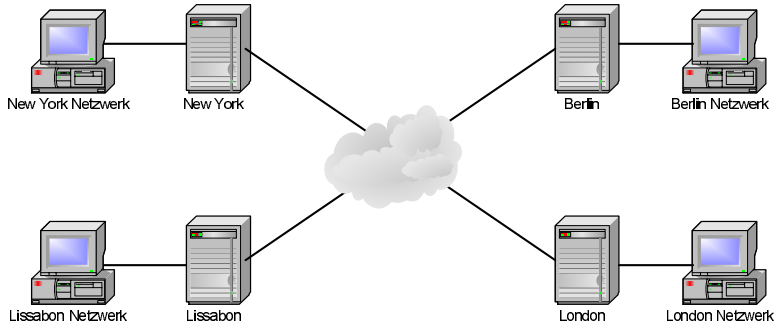


Abbildung 1.14 Netzförmiger Aufbau eines VPNs

1.6.4 Anbindung von Telearbeitsplätzen an einen Standort

Viele Firmen und ihre Mitarbeiter entdecken heute die Möglichkeiten der Telearbeit. Dabei greifen die Mitarbeiter mit ihrem Rechner von zuhause auf das Intranet der Firma zu. Aus Kostengründen werden hierzu immer mehr Internetverbindungen eingesetzt. So können auch DSL Bandbreiten für den Zugang genutzt werden.

Derartige Verbindungen über das Internet müssen jedoch sicher und geschützt aufgebaut werden. Hierfür eigenen sich idealerweise VPN Lösungen auf der Basis von IPsec. Die Einwahl erfolgt über einen lokalen Internetdienste Anbieter (Internet Service Provider, ISP). Dann wird ein IPsec Tunnel zu einem VPN Gateway der Firma aufgebaut und der Zugang hergestellt. Abbildung 1.15 zeigt ein derartiges Szenario.

Derartige Lösungen werfen gegenüber den oben bereits besprochenen Szenarien weitere neue Probleme auf, die hier kurz mit einem Stichwort erwähnt werden sollen:

- Keine statischen IP Adressen. Der Telearbeiter erhält bei seiner Einwahl bei seinem ISP eine beliebige IP Adresse. Die IP Adresse kann daher nicht zur Authentifizierung genutzt werden.

- Benutzerauthentifizierung bei der Anmeldung. Da nun das VPN durch einen Benutzer aufgebaut wird, wird auch häufig eine Authentifizierung gefordert, die einer Anmeldung entspricht.
- Unter Umständen eine Network Address Translation durch den ISP. Die IPsec Protokolle überprüfen die verwendeten IP Adressen der Kommunikation. Änderungen dieser IP Adressen führen häufig zu Problemen.
- Zuweisung einer IP Adresse aus dem internen Netz zur einfachen Kommunikation. Damit sich der Telearbeiter anschließend im Netz normal bewegen kann, soll ihm häufig eine virtuelle IP Adresse aus dem internen Netz zugewiesen werden.

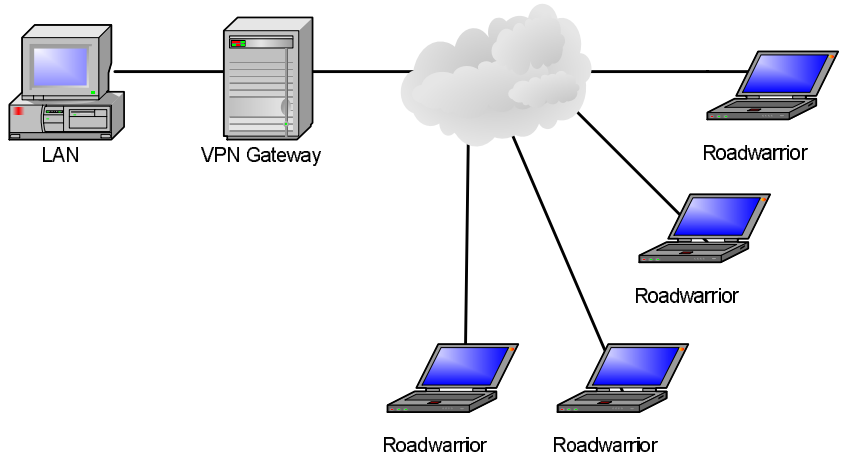


Abbildung 1.15 Zugriff von Telearbeitern auf ein Intranet (Roadwarrior)

Diese Punkte werden im weiteren in diesem Buch näher erläutert und Lösungen werden vorgestellt.

1.6.5 Anbindung von Außendienstmitarbeitern (Roadwarrior) an einen Standort

Dieses Szenario stellt im Grunde eine Kopie des letzten Szenarios dar. Auch hier besteht die Notwendigkeit, dass ein Außendienstmitarbeiter abends aus seinem Hotelzimmer seine Datenbanken mit den zentralen Firmendatenbanken synchronisieren möchte. Auch hier bietet sich ein VPN an. Der Außendienstmitarbeiter kann sich von seinem Telefonanschluss im Hotel bei einem lokalen oder nationalen ISP einwählen und so eine Internetverbindung aufbauen. Sie kann er dann anschließend nutzen um einen Tunnel zum Gate-

way der Firma aufzubauen. Über diese verschlüsselte Verbindung können dann alle Informationen ausgetauscht werden.

Dieses Szenario ist jedoch mit denselben Problemen behaftet, wie das Szenario mit den Telearbeitern.

1.6.6 Absicherung eines Wireless LAN

Wenn heute neue lokale Netzwerke (LAN) implementiert werden, so werden sie immer häufiger als Wireless LAN (WLAN) ausgeführt. WLANs können ohne Umbauarbeiten und dadurch verursachte Ausfallzeiten sehr einfach und schnell aufgesetzt werden. Mietverträge oder Denkmalschutz können die möglichen baulichen Änderungen bei einer Vernetzung stark einschränken. WLANs benötigen keine baulichen Veränderungen und können bereits mit einer Bruttobandbreite von 54 Mbit/s Daten transferieren. Für den Schutz dieser Daten wurde die Wired Equivalent Privacy (WEP) als Standard geschaffen. Sie verschlüsselt die übertragenen Daten mit 40 Bit oder 104 Bit[6] Dieser Verschlüsselungsmechanismus wurde bereits Mitte 2001 von Scott Fluhrer, Itsik Mantin und Adi Shamir geknackt (*http://www.crypto.com/papers/others/rc4_ksaproc.ps*). Die entsprechenden Werkzeuge wurden kurze Zeit später als Open Source Werkzeuge zur Verfügung gestellt (AirSnort: *http://airsnort.shmoo.com/*; WEPCrack: *http://wepcrack.sourceforge.net/*). Die Verschlüsselung kann daher nicht als ausreichend sicher angesehen werden.

Ein Wireless LAN kann jedoch recht gut mit IPsec geschützt werden. Hierzu ist nur wenig mehr Aufwand als beim weiter oben beschriebenen Stern Szenario erforderlich. Ein Beispielprojekt, das dies in universitärem Rahmen durchführt, ist das MOPO Projekt: *http://mopoinfo.wlan.informatik.uni-freiburg.de/*. Hier ist ein etwa 20 Accesspoints umfassendes WLAN aufgebaut worden, welches eine Authentifizierung des Benutzers mit x509 Zertifikaten durchführt und anschließend den Aufbau eines verschlüsselten IPsec Tunnels ermöglicht.

1.6.7 Opportunistische Verschlüsselung

Die opportunistische Verschlüsselung ist eine neue Eigenschaft, die bisher nur unter Linux mit FreeS/WAN zur Verfügung steht (siehe Kapitel 5, »FreeS/WAN«). Hierbei ist Linux in der Lage den VPN Tunnel bei Bedarf und technischer Verfügbarkeit aufzubauen. Dazu ermittelt das VPN Gate-

6. Die Länge des Schlüssels wird teilweise unterschiedlich angegeben. Bei beiden Schlüssellängen wird zusätzlich ein 24 Bit langer Intialisierungsvektor zusätzlich verwendet. Daraus resultieren dann 64 Bit beziehungsweise 128 Bit.

way mit Hilfe des DNS Dienstes ob der entsprechende Kommunikationspartner möglicherweise durch ein VPN Gateway geschützt wird. Ist dies der Fall, so baut das Linux VPN Gateway zu dem entsprechenden zweiten VPN Gateway einen IPsec Tunnel auf und überträgt die Daten verschlüsselt. Steht kein VPN Gateway zur Verfügung, so werden die Daten in Klartext übertragen.

Diese Funktion ermöglicht den Aufbau von VPN Verbindungen mit beliebigen Partnern, die die entsprechenden Informationen in ihren DNS Servern hinterlegen. So kann eine schrittweise Migration erfolgen.

Das ist zum Beispiel sinnvoll, wenn innerhalb eines Netzwerkes die Kommunikation auf IPsec umgestellt werden soll. Es ist meist nicht möglich über Nacht die Konfiguration auf allen Systemen zu modifizieren und anzupassen. Bei opportunistischer Verschlüsselung ermitteln die Systeme selbst, mit welchen Systemen sie eine verschlüsselte Verbindung aufbauen können.

2 Kryptografie

Die Kryptografie ist eine elementare Technologie beim Aufbau von Virtuellen Privaten Netzwerken. Die Kenntnis der verschiedenen Verfahren, ihrer Zusammenhänge und ihrer Vor- und Nachteile ist sinnvoll für das Verständnis. Dieses Kapitel stellt nach einem kurzen Ausflug in die Geschichte die symmetrische und asymmetrische Verschlüsselung, wie sie heute verwendet wird, vor. Anschließend werden der Diffie Hellmann Schlüsselaustausch und die Hash Funktionen besprochen. Ohne diese Methoden sind die IPsec Protokolle nicht denkbar.

2.1 Einleitung

Die Kryptografie ist ein Teilbereich der Kryptologie, die zusätzlich auch die Kryptoanalyse behandelt. Diese beiden Disziplinen der theoretischen Mathematik versuchen entweder die Vertraulichkeit eines Textes durch Verschlüsselung zu garantieren oder die entsprechende Verschlüsselung zu brechen und einen verschlüsselten Text in seinen Klartext zu überführen.

Für das Verständnis der folgenden Seiten ist die Kenntnis einiger Begriffe sinnvoll:

- *Kryptografie* ist die Durchführung und das Studium der Ver- und Entschlüsselung. Hierbei werden Daten so mathematisch kodiert, dass nur bestimmte ausgewählte Personen die Daten dekodieren können. Üblicherweise werden die lesbaren Daten (Klartext, plaintext) entsprechend einem Algorithmus mit einem geheimen Schlüssel kodiert (ciphertext auch Chiffretext). Ziel der Kryptografie ist es möglichst sichere Verschlüsselungssysteme zu entwickeln.
- Die *Kryptoanalyse* beschäftigt sich ebenfalls mit dem Studium der Verschlüsselung und Entschlüsselung. Ihr Ziel ist jedoch die Entdeckung von Lücken in den eingesetzten Algorithmen oder Schlüsseln um so eine Entschlüsselung der kodierten Daten zu ermöglichen.
- Der *Ciphertext-Only Angriff* versucht eine Verschlüsselung nur bei Kenntnis des chiffrierten Textes zu lösen. Dies ist die größte Herausforderung für jeden Kryptoanalytiker.
- Der *Brute Force* Angriff ist ein Angriff, der bei jeder Verschlüsselungsmethode eingesetzt werden kann, deren Algorithmus bekannt ist. Der Brute Force Angriff probiert nacheinander alle verschiedenen möglichen

Schlüssel für die Entschlüsselung durch. Hierfür muss der Angreifer jedoch erkennen können, wann er den richtigen Schlüssel gefunden hat. Daher wird dieser Angriff häufig auch als Klartext Angriff bezeichnet. Häufig genügen jedoch bereits allgemein bekannte Informationen über den Klartext. So beginnt jedes Microsoft Word Dokument unabhängig von seinem Inhalt mit einem einheitlichen Dateiheader, der erkannt werden kann.

- *Alice*, *Bob* und *Eve* oder *Charles* sind die üblicherweise in der kryptografischen Literatur verwendeten Beispielpersonen. Hierbei versuchen Alice und Bob eine gesicherte Kommunikation aufzubauen. Eve (für Eavesdropper, Lauscher) oder Charles (wegen den Buchstaben A, B und C) versuchen die Kommunikation abzuhören. Diese Namen werden daher auch hier verwendet.

2.2 Geschichte

Der Wunsch nach der Vertraulichkeit übertragener Daten ist sehr alt. Es ist bekannt, dass bereits frühe Hochkulturen eine Nachricht in die Kopfhaut eines Sklaven tätowierten. Sobald dessen Haare nachgewachsen waren, wurde der Sklave zum Empfänger geschickt, der nur den Kopf rasieren musste um die Nachricht zu lesen.

Das erste militärisch genutzte System wurde etwa 500 Jahre vor Christus von dem spartanischen General Pasanius verwendet. Bei der Skytale handelte es sich um einen Holzstab mit definiertem Durchmesser. Der Absender wickelte einen Papierstreifen um diesen Holzstab und schrieb seine Nachricht quer zur Wickelrichtung (Abbildung 2.1). Anschließend wurde der Papierstreifen abgezogen und verschickt. Hierbei verschieben sich die Buchstaben in Leserichtung. Daher werden derartige Verfahren als Transpositionsalgorithmus bezeichnet. Der Empfänger benötigte zum erfolgreichen Lesen der Nachricht einen Holzstab identischer Dicke.

Julius Cäsar entwickelte den Cäsar-Code (Caesar's Cipher). Hierbei wird jeder Buchstabe im Alphabet durch einen anderen Buchstaben ausgetauscht. Diese Verfahren werden auch als Substitutionsalgorithmus bezeichnet. Die älteste Form ist wahrscheinlich die Methode Atbash. Hierbei wurde der erste Buchstabe im hebräischen Alphabet durch den letzten, der zweite durch den vorletzten Buchstaben und so weiter ausgetauscht. Dadurch ergibt sich ein Substitutionsalphabet wie in Listing 2.1.

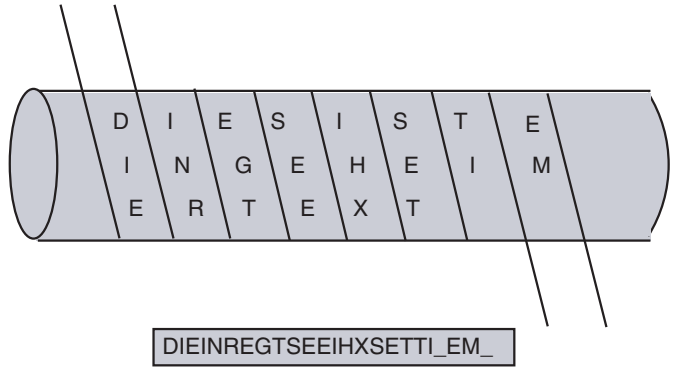


Abbildung 2.1 Der Skytale verwendet einen Transpositions-Algorithmus

```
Klartext: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:   ZYXWVUTSRQPONMLKJIHGFEDCBA
```

Listing 2.1 Bei Atbash wird der erste Buchstabe durch den letzten ausgetauscht

Cäsars Verfahren unterschied sich von Atbash durch eine zusätzliche Variabilität. Hierbei wurden die Alphabete um eine bestimmte Anzahl von Buchstaben gegeneinander rotiert. Für die Ver- und Entschlüsselung der Nachrichten war erstmals nicht nur die Kenntnis des Algorithmus sondern auch eine Art Schlüssel erforderlich. Die Funktionsweise des Verfahrens lässt sich leicht am Beispiel des Namens des Hauptrechners aus dem Spielfilm von Stanley Kubrick »Space Odyssee 2002« nachvollziehen. Dieser Rechner erhielt den Namen HAL. Eine Rotation um eine Stelle im Alphabet ergibt: IBM. Julius Cäsar verwendete jedoch meist eine Rotation um 13 Stellen (Beispiel 2.2).

```
Klartext: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rot-13:   NOPQRSTUVWXYZABCDEFGHIJKLM
```

Listing 2.2 Julius Cäsar verwendete das Verfahren Rot-13

TIPP

Ein schönes Werkzeug zur Demonstration dieser Algorithmen ist CrypTool. Dieses Werkzeug ist über seine Homepage *http://www.cryptool.de/* verfügbar.

Caesar's Cipher weist jedoch einige Schwächen auf. Die wesentliche Schwäche kann durch eine Frequenzanalyse ausgenutzt werden. Hierbei wird die Häufigkeit der einzelnen Buchstaben in der natürlichen Sprache bestimmt. In der deutschen Sprache besitzt der Buchstabe E eine Häufigkeit von etwa

18 Prozent. Die Häufigkeit des Buchstaben X ist jedoch fast Null (siehe Seite 409). Wendet der Kryptoanalytiker dieses Wissen auf den ciphertext an, so kann er sehr schnell erkennen, um wie viele Buchstaben das Alphabet rotiert wurde. Daher sind derartige monoalphabetische Substitutionsalgorithmen, bei denen jeder Buchstabe des Alphabet immer durch denselben Buchstaben ersetzt wird, nicht sehr sicher.

Besser sind hier polyalphabetische Substitutionsalgorithmen. Sie tauschen einen Buchstaben des Alphabets durch eine bestimmte Folge von Buchstaben aus. Die ersten polyalphabetischen Verfahren gehen auf Leon Battista Alberti zurück. Er beschrieb 1466 ein Verfahren bei dem zwei Kupferscheiben, auf denen sich jeweils das Alphabet befand, gegeneinander verdreht wurden. Dabei wurden die Scheiben nach einigen verschlüsselten Worten verdreht und so der Schlüssel ausgetauscht. Dieses Verfahren wurde von dem Abt Johannes Trithemius weiterentwickelt. Hierbei kamen zum ersten Mal Substitutionstabellen zum Einsatz.

Die bekannteste Variante wurde schließlich von Blaise de Vigenere entwickelt. Vigenere entwickelte eine Tabelle (Listing 2.3) und einen Algorithmus, um mit dieser Tabelle einen Text zu verschlüsseln.

```
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
b B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
c C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
d D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
e E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
f F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
g G H I J K L ....
h H I J K L ....
....
```

Listing 2.3 Die Vigenere Tabelle

Um nun einen Text zu verschlüsseln wird zunächst ein Schlüssel gewählt, zum Beispiel: *IPSEC*. Dann wird der Schlüssel durch Wiederholungen auf die identische Länge gebracht. Anschließend kann der Benutzer den Text verschlüsseln, indem er den entsprechenden ciphertext in der Tabelle abliest. Dabei definiert der Schlüssel die Spalten und der Klartext die Zeilen in der Tabelle.

```
Schlüssel : IPSECIPSECIPSECIPSECIPSECIPSECIPSECIPSEC
Klartext  : diesistgeheimertext
ciphertext: LXWWKAIYIJMXEITBTPX
```

Listing 2.4 Verschlüsselung mit Vigenere

Durch den polyalphabetischen Algorithmus werden die Buchstaben nacheinander durch WIMIT ersetzt. Eine Frequenzanalyse ist nun wesentlich aufwändiger.

Der Vigenere Algorithmus wurde bis in die moderne Zeit benutzt.

Seit dem ersten Weltkrieg steigt die Bedeutung der Kryptografie stark und war teilweise kriegsentscheidend. So waren die Allierten im zweiten Weltkrieg in der Lage die deutsche und japanische Verschlüsselung zu brechen. David Kahn veröffentlichte die Geschichte der Kryptoanalyse der Enigma in seinem Buch »The Codebreakers«. Mit dem Erscheinen dieses Buches stieg in den 60er Jahren auch das öffentliche Interesse an der Kryptografie.

Seit dem zweiten Weltkrieg sind Verfahren entwickelt worden, bei denen noch keine Schwächen entdeckt wurden. Diese starken kryptografischen Verfahren bieten bei ausreichender Schlüssellänge praktisch einen totalen Schutz. Ein Knacken der verschlüsselten Texte würde Ressourcen benötigen, die momentan nicht zur Verfügung stehen. Ein Angriff kann daher nicht in annehmbarer Zeit durchgeführt werden.

Aus diesem Grund wird oder wurde der Kryptografie ein ähnlicher Stellenwert zugesprochen wie anderen Kriegsmitteln. Starke Kryptografie wird daher meist von der Kriegswaffenkontrollgesetzgebung reguliert. Die Vereinigten Staaten von Amerika verboten ab 1993 die Ausfuhr starker Kryptografie. Diese Beschränkungen sind inzwischen wieder weitgehend aufgehoben worden, da allgemein verstanden wurde, dass dies nicht durch Gesetze regulierbar ist. Jedoch ist der Export starker Kryptografie aus den USA in als solche bezeichnete »Schurkenstaaten« weiterhin untersagt. Auch in weiteren Ländern wie zum Beispiel Frankreich unterliegt die Ein- oder Ausfuhr starker Kryptografie besonderen Regelungen.

2.3 Symmetrische Verschlüsselung

Alle in der Praxis verwendeten symmetrischen Verschlüsselungsverfahren basieren auf einem allgemein bekannten Algorithmus und einem geheimen Schlüssel. Die Offenlegung des Algorithmus wird als eine Vorraussetzung für die Sicherheit des Verfahrens angesehen. Nur so ist es möglich, dass Kryptoanalytiker in der Lage sind das Verfahren zu studieren und Fehler zu finden. Nur wenn die Verfahren mehrere Jahre derartiger offener Kryptoanalyse durch unabhängige Fachleute überstanden haben, werden sie als momentan sicher anerkannt. Dazu darf keine Schwäche entdeckt werden, die einen Angriff ermöglicht, der ein schnelleres Ergebnis liefert, als ein Brute Force Angriff.

Eine Auswahl von symmetrischen Verfahren, die als sicher angesehen werden sind: DES, 3DES, Blowfish, Twofish, RC5, Serpent und AES (Rjindael).

Wie funktioniert nun die symmetrische Verschlüsselung? Grundvorraussetzung für die erfolgreiche Anwendung ist die Vereinbarung eines zu verwendenden Verschlüsselungsalgorithmus und eines geheimen Schlüssels. Bei der symmetrischen Verschlüsselung müssen beide Kommunikationspartner den identischen Schlüssel besitzen. Ein symmetrisches Verfahren verwendet den identischen Schlüssel für die Kodierung eines Textes und die spätere Entschlüsselung des ciphertextes. Daher der Zusatz »symmetrisch«. Abbildung 2.2 verdeutlicht den Ablauf.

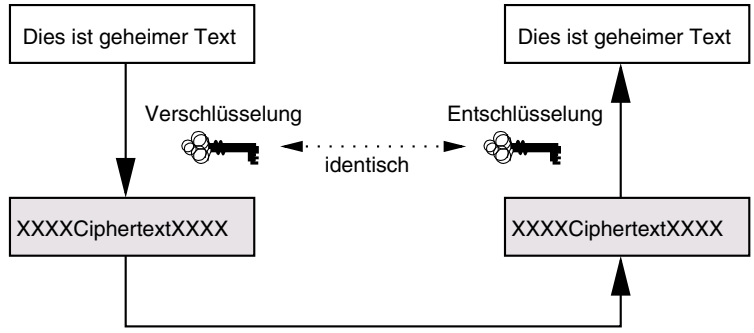


Abbildung 2.2 Bei der symmetrischen Verschlüsselung wird nur ein gemeinsamer Schlüssel verwendet

Werden sichere Verfahren für die Verschlüsselung eingesetzt, so hängt die Sicherheit der Verschlüsselung lediglich von der Schlüssellänge ab. Sie bestimmt, wie viele verschiedene Schlüssel möglich sind. Bei einem erfolgreichen Brute Force Angriff müssen statistisch 50 Prozent aller möglichen Schlüssel getestet werden, im schlechtesten Fall sogar sämtliche Schlüssel.

Um die Schlüssellängen der verschiedenen Verfahren bewerten zu können, soll nun kurz eine einfache Rechnung durchgeführt werden. Bei einem 56 Bit langen Schlüssel existieren 2^{56}=72.057.594.037.927.936 mögliche Schlüssel. Dauert der Test eines Schlüssels eine Mikrosekunde auf einem Rechner, so benötigt der Rechner 2284 Jahre um sämtliche Schlüssel zu testen. Können diese Tests parallel durchgeführt werden, reduziert sich entsprechend die Dauer.

Kürzere beziehungsweise längere Schlüssel verkürzen oder verlängern die benötige Zeit für einen Brute Force Angriff drastisch. Ein 40 Bit langer Schlüssel, wie er typischerweise vor der Lockerung der US amerikanischen Exportbeschränkungen bei exportierter Software eingesetzt wurde, ermög-

licht nur 1.099.511.627.776 verschiedene Schlüssel. Ein Test aller Schlüssel würde in diesem Fall gut zwölf Tage benötigen. Ein paralleler Angriff könnte diese Dauer auf wenige Minuten oder Sekunden reduzieren. Ein entsprechender 128 Bit langer Schlüssel erlaubt 340.282.366.920.938.463.463. 374.607.431.768.211.456 verschiedene Kombinationen. Unter denselben Voraussetzungen würde der Rechner 10.790.283.070.806.014.188.970.529 Jahre für die Berechnung benötigen.

Wenn Sie diese Berechnungen nachvollziehen möchten, so können Sie den universellen Rechner `bc` und das Programm aus Listing 2.5 verwenden.

```
# Dieses Programm berechnet die Dauer für einen Brute Force Angriff

# Wie lang ist der Schlüssel?
print "Wie lang ist der Schlüssel? --> "
keylength = read()

# Wieviel Zeit wird für eine Operation benötigt (in Mikrosekunden)?
dauer=1

schluessel = 2 ^ keylength

schluessel = schluessel * dauer

tag = (((((schluessel / 1000) / 1000) / 60) / 60) / 24)
jahr = ((((((schluessel / 1000) / 1000) / 60) / 60) / 24) / 365)

print "Länge:     ",keylength," Bit\n"
print "Schlüssel: ",schluessel,"\n"
print "Dauer:     ",jahr," Jahre oder ",tag," Tage\n"
quit
```

Listing 2.5 `bc` berechnet den Brute Force Aufwand

Im Folgenden sollen kurz die üblicherweise verwendeten symmetrischen Verschlüsselungsverfahren vorgestellt werden. Dabei wird nicht näher auf ihre mathematischen Hintergründe eingegangen. Es existieren hierfür wesentlich bessere Bücher, die in der Bibliografie (siehe »Bibliografie«, S. 417) erwähnt werden. Für den Einsatz in einem VPN ist außer der Sicherheit des Algorithmus lediglich die Schlüssellänge und die Geschwindigkeit entscheident.

2.3.1 Data Encryption Standard (DES)

Die Entwicklung von DES dauerte mehrere Jahre unter der Federführung von IBM. 1977 wurde DES in den USA zum nationalen Standard erklärt. Lange Jahre wurde der Algorithmus mit Skepsis betrachtet, da auch die NSA

(National Security Agency) an der Entwicklung beteiligt war. Inzwischen gilt es jedoch als sicher, dass die NSA keine Hintertür in dem Algorithmus eingebaut hat. DES ist ein Verschlüsselungsalgorithmus, der immer ganze Datenblöcke von 64 Bit verschlüsselt. Er verwendet einen Schlüssel mit 56 Bit Länge. Dieser Schlüssel enthält zusätzlich 8 Bit Paritätsdaten, so dass der gesamte Schlüssel 64 Bit lang ist. DES stellte für lange Jahre den Standard in der Kodierung dar und lässt sich sehr gut in Software und noch besser in Hardware implementieren. Es existieren Hardwarechips der unterschiedlichsten Hersteller, die mehrere 100 MByte/s verschlüsseln können. Dies führt inzwischen auch zur Unsicherheit von DES. Es sind bis heute keine wirksamen Angriffe gegen DES bekannt, jedoch entwickelte bereits 1993 Michael Wiener einen Rechner für eine Million Dollar, der in der Lage ist einen Brute Force Angriff in 3,5 Stunden im Schnitt erfolgreich durchzuführen. 1999 konnte die Electronic Frontier Foundation (EFF) einen 56 Bit Schlüssel in 22 Stunden knacken. Das Problem liegt nicht im Algorithmus, sondern im kurzen Schlüssel.

Die IPsec Standards verlangen die Implementierung von DES als Verschlüsselungsalgorithmus. DES soll als kleinster gemeinsamer Nenner bei allen IPsec Implementierungen zur Verfügung stehen. Einige kleine Router sind nur in der Lage DES zu verwenden. Viele kommerzielle Anbieter knüpfen den Einsatz stärkerer Verschlüsselung an besondere Lizenzen und verlangen hierfür gesondert Gebühren. Grundsätzlich sollte von einem Einsatz von DES in einem VPN jedoch abgesehen werden.

2.3.2 3DES

3DES oder auch Triple-DES stellt die Antwort auf die Verwundbarkeit von DES dar. Wie oben dargestellt wurde, ist DES auf Grund des kurzen Schlüssels von 56 Bit recht schnell zu knacken. 3DES umgeht dieses Problem, in dem es zwei beziehungsweise drei 56 Bit Schlüssel verwendet. Diese Schlüssel werden genutzt um dreimal eine Verschlüsselung der Daten durchzuführen.

Hierzu werden die Daten zunächst mit dem ersten Schlüssel kodiert, anschließend mit dem zweiten Schlüssel entschlüsselt und schließlich mit dem ersten (bei zwei Schlüsseln) oder dem dritten Schlüssel (bei drei Schlüsseln) erneut verschlüsselt. So verwendet 3DES einen 112 Bit oder 168 Bit langen Schlüssel. Leider ist dieses Verfahren sehr zeitaufwändig. Dieser Nachteil wird jedoch vielfach dadurch aufgewogen, dass es für DES optimierte Chipsätze gibt, die relativ einfach auf 3DES erweitert werden können. So kann durch den Einsatz von Hardware-Kryptoprozessoren die Verschlüsselung beschleunigt werden.

3DES ist heute sicherlich der am häufigsten eingesetzte Verschlüsselungsalgorithmus in IPsec basierten VPN Lösungen. Es existieren langjährige Erfahrungen mit DES und ein großes Vertrauen in die Sicherheit des Algorithmus, die einfach auf 3DES übertragen werden können. 3DES bietet die Stabilität von DES mit der Sicherheit eines 168 Bit langen Schlüssels.

2.3.3 International Data Encryption Algorithm (IDEA)

IDEA ist ein patentierter Verschlüsselungsalgorithmus, der wie DES 64 Bit Datenblöcke bearbeitet. Er verwendet einen Schlüssel mit 128 Bit Länge. Bruce Schneier bewertet ihn in seinem Buch »Angewandte Kryptografie« als einen der besten und sichersten verfügbaren Algorithmen. Softwareimplementierungen des IDEA Algorithmus sind etwa doppelt so schnell wie DES. Patentinhaber ist die Firma Ascom Systec AG in der Schweiz.

Auf Grund des IDEA Patents wird dieser Algorithmus nur sehr selten für die Verschlüsselung in IPsec basierten VPN Lösungen genutzt. Insbesondere Open Source Lösungen deaktivieren häufig diesen Algorithmus oder implementieren ihn gar nicht, um mögliche Patentstreitigkeiten zu vermeiden.

2.3.4 RC4/RC5/RC6

Die RC-Algorithmen wurden von Ron Rivest entwickelt. Der RC5-Algorithmus ist in der Lage mit variabler Blockgröße, Schlüssellänge und Schleifendurchläufen zu arbeiten. Kryptoanalysen zeigen, dass der Algorithmus wahrscheinlich ab einer Schleifenanzahl von sechs sicher ist. Ron Rivest empfiehlt eine Schleifenanzahl von wenigstens zwölf.

Der Name RC5 ist als Warenzeichen und Patent angemeldet.

Der Nachfolger RC6 wurde von Ron Rivest gemeinsam mit Matt Robshaw, Ray Sidney und Yiqun Lisa Yin bei der Firma RSA Laboratories für den AES Wettbewerb entwickelt. Es handelt sich um einen sehr modernen und schnellen Algorithmus der auf den Erfahrungen von RC5 basiert. Die Schlüssellänge ist variabel zwischen 0 und 255 Bytes (entsprechend 2048 Bit).

2.3.5 Blowfish

Blowfish wurde von Bruce Schneier entwickelt. Die Entwicklung zielte auf den Einsatz auf großen Mikroprozessoren. Ein 32 Bit Mikroprozessor kann ein Byte Daten üblicherweise in 26 Takten verschlüsseln. Der Algorithmus benötigt nur fünf KByte Speicher und verwendet lediglich einfache 32 Bit

Operationen. Die Schlüssellänge von Blowfish ist variabel und kann bis zu 448 Bits lang sein. Blowfish ist ideal für Anwendungen, bei denen der Schlüssel nur selten getauscht wird und große Datenmengen mit demselben Schlüssel verarbeitet werden. Der Algorithmus ist auf einem Pentium Prozessor wesentlich schneller als DES. Blowfish weist einige Schwächen auf, wenn er nicht komplett implementiert wird. Wie die meisten anderen Algorithmen arbeitet diese Methode mit Schleifen. Wird die Schleifenanzahl gekürzt, so sinkt möglicherweise die Sicherheit drastisch (Seite 409).

Blowfish ist frei von Patenten und in der Public-Domain. Der Algorithmus wird unter anderen von der OpenSSH und von OpenBSD eingesetzt. Insbesondere in Open Source VPN Lösungen wird auch Blowfish unterstützt. Die Unterstützung von Blowfish bei IPsec basierten VPN Lösungen nimmt zu.

2.3.6 Twofish

Twofish wurde ebenfalls wie Blowfish von Bruce Schneier entwickelt. Es verarbeitet Daten in 128 Bit Blöcken und kann einen bis zu 256 Bit langen Schlüssel verwenden. Es eignet sich für die Implementierung in Hardware, auf Smartcards und in Software. Bisher konnte kein Angriff gegen Twofish entwickelt werden. Twofish gelangte mit einigen anderen Kandidaten (Serpent, RC6, Rijndael) in die Endausscheidung zum Advanced Encryption Standard.

Twofish ist wie Blowfish nicht patentiert, frei von Copyright und in der Public Domain.

2.3.7 AES

Im Jahr 1997 hat das National Institute of Standards and Technology (NIST) einen Wettbewerb für einen neuen Verschlüsselungsstandard ausgeschrieben. Dieser Advanced Encryption Standard (AES) sollte die längst überfällige Ablösung des DES darstellen. Insgesamt 15 Algorithmen nahmen als Kandidaten am Wettbewerb teil (*http://csrc.nist.gov/encryption/aes/index2.html*). Zu den Finalisten gehörten MARS, RC6, Rijndael, Serpent und Twofish. Schließlich wurde der Algorithmus Rijndael von den Belgiern Joan **Dae**men und Vincent **Rij**men zum AES gekürt. Hierbei handelt es sich um einen Algorithmus mit variabler Block- und Schlüssellänge. Definiert wurde bisher das Verhalten für Blöcke und Schlüssel von 128, 192 und 256 Bit Länge.

Rijndael kann sehr effizient sowohl in Hard- als auch in Software implementiert werden. Er stellt einen der schnellsten momentan verfügbaren Algorithmen dar.

Die Spezifikation zur Wahl des AES erfordert unter anderem die lizenzfreie weltweite Verfügbarkeit des Algorithmus. Er unterliegt also keiner Einschränkung.

Moderne IPsec Implementierungen verwenden diesen Algorithmus besonders gerne, da er frei von Einschränkungen einen der schnellsten verfügbaren Verschlüsselungsalgorithmen mit ausreichender Schlüssellänge darstellt.

ACHTUNG

Es existieren inzwischen Zweifel an der Qualität des AES Verschlüsselungsalgorithmus (*http://www.counterpane.com/crypto-gram-0209.html#1*). Courtois und Pieprzyk haben einen Angriff gegen Serpent und AES beschrieben. Die Wirksamkeit dieses Angriffes ist umstritten. Eric Filiol hat zwei Artikel veröffentlicht, in denen er Angriffe gegen den Algorithmus beschreibt. Diese Angriffe konnten jedoch inzwischen widerlegt werden. Dennoch sollte der Einsatz von AES überdacht werden.

2.3.8 Weitere Verfahren

Es existieren eine ganze Reihe weiterer Verfahren, die aber meist keine besondere Bedeutung für den Einsatz in einer IPsec VPN Lösung haben. Hierzu gehören CAST, CAST-128, CAST-256, MARS und Serpent. Das bedeutet nicht, dass sie nicht geeignet sind, häufig existieren nur gleich gute oder bessere Verfahren. Weitere Informationen können in der einschlägigen Literatur nachgelesen werden.

2.4 Cipher Block Chaining (CBC)

Im Zusammenhang mit der geschichtlichen Entwicklung der Kryptografie wurde die Frequenzanalyse erwähnt. Sie kann auch auf die klassischen symmetrischen Verfahren angewendet werden, die im letzten Abschnitt besprochen wurden. Immerhin verschlüsselt das DES Verfahren identische Infor-

mationen mit dem gleichen Ergebnis bei gleichen Schlüsseln. Stehen genug Daten zur Verfügung, kann damit ein statistischer Angriff durchgeführt werden.

Um dies zu vermeiden wird eine Verkettung der Datenblöcke durchgeführt. Dabei wird hauptsächlich der zu verschlüsselnde Datenblock vor seiner Verschlüsselung exklusiv- oder mit dem letzten verschlüsselten Datenblock verknüpft. Erst nach dieser XOR-Verknüpfung wird die Verschlüsselung durchgeführt. Gleiche Klartextblöcke werden so unterschiedlich modifiziert und unterschiedlich verschlüsselt. Ein besonderes Problem stellt aber der erste zu verschlüsselnde Datenblock dar. Für ihn wird ein Initialisierungsvektor (IV) erzeugt. Dieser IV wird meist aus Zufallszahlen erzeugt. Da dieser IV auch für die Entschlüsselung benötigt wird, muss er dem Empfänger übermittelt werden. Seine Vertraulichkeit ist jedoch nicht erforderlich, daher wird er meist im Klartext übermittelt. Die IPsec Protokolle übertragen den IV Vektor in jedem Datenpaket (siehe ESP 3.2.2, »Verschlüsselung«).

2.5 Asymmetrische Verschlüsselung

Bei der asymmetrischen Verschlüsselung (auch Public Key Kryptografie) werden im Gegensatz zur symmetrischen Verschlüsselung zwei verschiedene Schlüssel benötigt. Sie werden als öffentlicher Schlüssel oder Public Key und als privater Schlüssel oder Private Key bezeichnet.

Grundlage der Public-Key-Kryptografie sind ungelöste oder sehr schwierige mathematische Probleme. Dabei kommen üblicherweise Funktionen zum Einsatz, deren Berechnung in einer Richtung sehr schnell und einfach durchzuführen ist, deren Umkehrung aber unmöglich ist. Als unmöglich definiert man in diesem Zusammenhang einen Rechnenaufwand, der einige tausend bis Millionen Jahre beträgt. Das Diffie Hellmann-Verfahren (siehe Abschnitt 2.5.6) beruht auf einer derartigen Funktion. Wenn die Verfahren tatsächlich zur Verschlüsselung eingesetzt werden sollen, so ist es erforderlich eine Umkehrung zu ermöglichen. Hier werden Falltürfunktionen eingesetzt, die bei Kenntnis eines privaten Schlüssels eine Umkehrung ermöglichen.

Üblicherweise wird der Public Key zur Verschlüsselung und der Private Key zur Entschlüsselung verwendet. Ein Dokument, das mit dem Public Key verschlüsselt wurde, kann nicht mit dem Public Key entschlüsselt werden. Hierzu ist der Private Key erforderlich (siehe Abbildung 2.3).

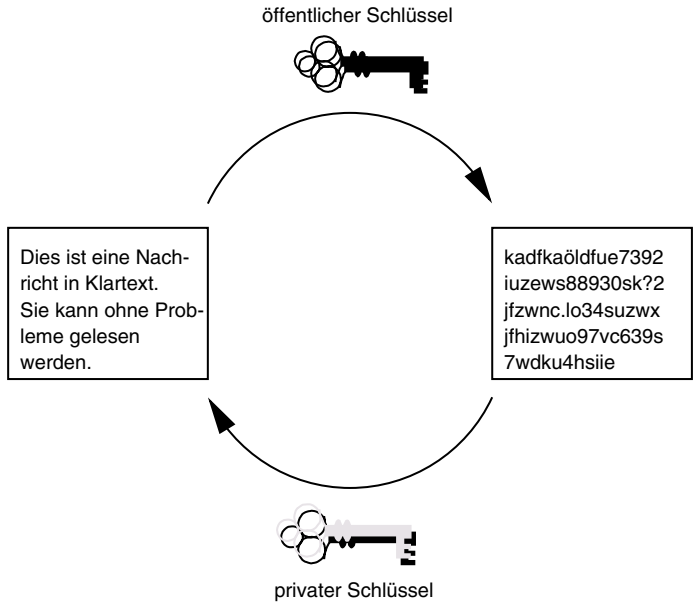


Abbildung 2.3 Der Public Key verschlüsselt

Möchte Alice ein Dokument verschlüsselt an Bob senden, erzeugt Bob zunächst ein derartiges Schlüsselpaar. Den privaten Schlüssel verwahrt Bob möglichst sicher und erlaubt keinem Dritten den Zugriff. Den öffentlichen Schlüssel kann er über ein beliebiges Medium verteilen. So erhält auch Alice den öffentlichen Schlüssel von Bob. Alice kann nun mit diesem Schlüssel das Dokument kodieren. Da dieser Schlüssel aber lediglich zur Verschlüsselung eingesetzt werden kann, ist sie nicht in der Lage, die verschlüsselte Nachricht zu dechiffrieren. Dies ist nur mit dem entsprechenden privaten Schlüssel möglich. Lediglich Bob besitzt den Private Key und nur Bob kann daher den ciphertext wieder in den Klartext überführen (Abbildung 2.4).

Bei diesem Verfahren ist es nicht nötig, vorher einen gemeinsamen geheimen Schlüssel zu vereinbaren, dessen sichere Übertragung gewährleistet werden muss. Dies ist der entscheidene Vorteil des Verfahrens. Bei einer symmetrischen Verschlüsselung ist es erforderlich, bevor das Dokument verschlüsselt werden kann, eine Vereinbarung über den symmetrischen Schlüssel zu treffen. Da zu diesem Zeitpunkt noch keine Verschlüsselung möglich ist, ist es sehr schwer den Code vertraulich auszuhandeln. Es handelt sich um ein klassisches Henne-Ei Problem.

Einige asymmetrische Verfahren (zum Beispiel DSA) verwenden jedoch auch die Schlüssel umgekehrt. Der Private Key bietet die Verschlüsselung und der Public Key kann genutzt werden um ein so verschlüsseltes Dokument zu

entschlüsseln. Einige Verfahren unterstützen beide Verwendungen (zum Beispiel RSA). Diese Verwendung wird nicht zur eigentlichen Verschlüsselung eingesetzt, sondern dient zur digitalen Signatur. Hiermit ist es auch möglich eine Authentifizierung durchzuführen.

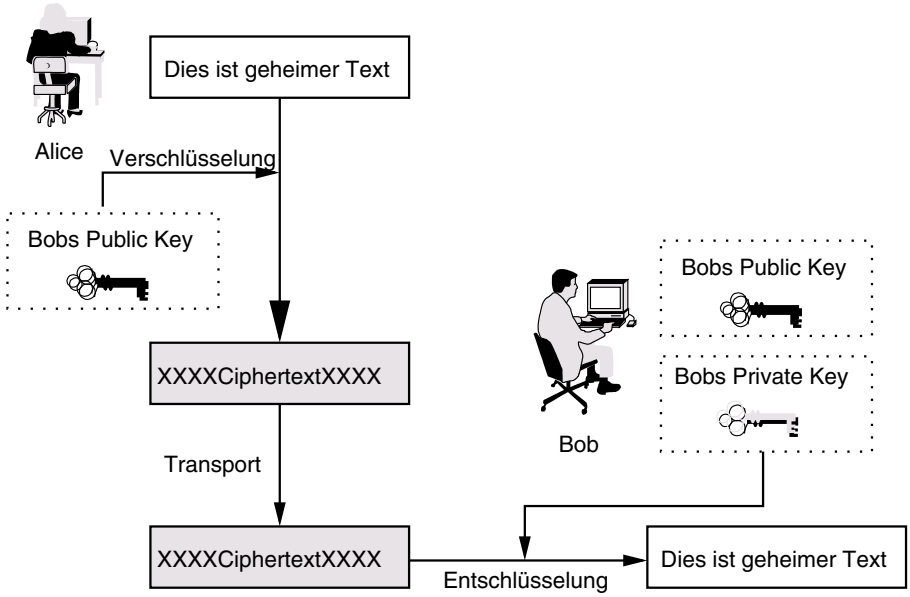


Abbildung 2.4 Asymmetrische Verschlüsselung

Die Abbildung 2.5 zeigt den schematischen Ablauf. Hierbei möchte Bob eine Nachricht an Alice senden. Alice soll in der Lage sein, Bob eindeutig als Absender zu identifizieren und die Integrität der Nachricht nachzuvollziehen. Hierzu kann Bob die Nachricht mit seinem privaten Schlüssel verschlüsseln. Besitzt Alice den öffentlichen Schlüssel von Bob, kann sie die Nachricht dekodieren. Ist sie erfolgreich, so wurde die Nachricht von Bob geschrieben, denn lediglich Bob besitzt den für die Verschlüsselung erforderlichen privaten Schlüssel. Außerdem kann die Nachricht nicht verändert worden sein, denn dann hätte die Entschlüsselung nicht funktionieren können. So kann zweifelsfrei die Echtheit der Nachricht überprüft werden.

Um dieses Verfahren nun zur Authentifizierung einzusetzen, sendet Alice eine sehr große Zufallszahl in Klartext an Bob. Dies bezeichnet man als Herausforderung oder Challenge. Bob nimmt diese Zahl, verschlüsselt sie mit seinem Private Key und sendet sie an Alice zurück. Alice kann diese Information mit dem Public Key von Bob entschlüsseln. Erhält sie die identische Zahl, so muss es sich um Bob handeln, da nur Bob den entsprechenden privaten Schlüssel besitzt.

In Verbindung mit VPN Lösungen werden asymmetrische Verfahren in erster Linie zur Authentifizierung und zum Schlüsselaustausch mit dem Diffie Hellmann Verfahren eingesetzt. Das Diffie Hellmann Verfahren wird in einem eigenen Abschnitt (siehe 2.5.6) besprochen.

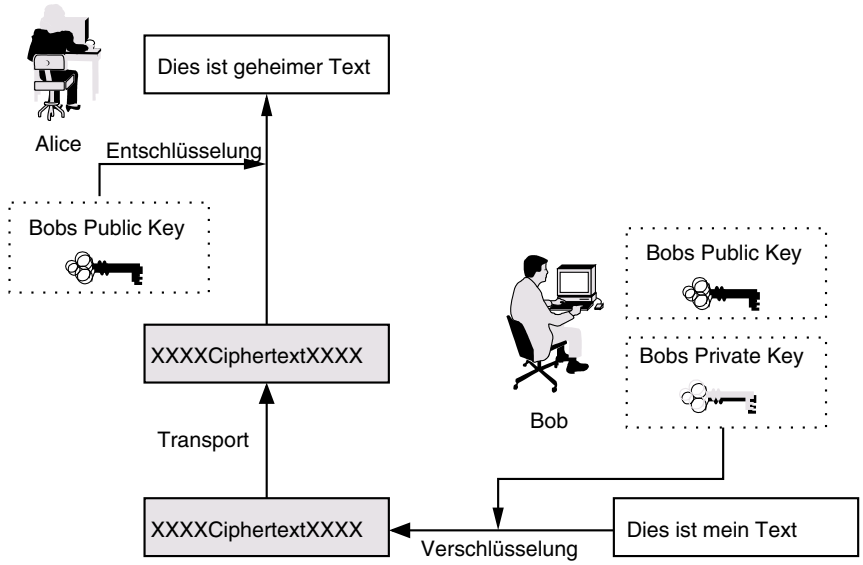


Abbildung 2.5 Digitale Signatur

Die Verwaltung der Schlüssel und ihre Einbettung in einer Schlüsselinfrastruktur (PKI) werden im Kapitel »Keymanagement« (Kap. 6) untersucht.

2.5.1 Das beste aus beiden Welten

Asymmetrische Verfahren sind wesentlich aufwändiger in der Implementierung und ihrer tatsächlichen Durchführung. Dies hängt auch mit den wesentlich längeren Schlüsseln zusammen, die für einen sicheren Einsatz erforderlich sind. Typischerweise werden 1024 oder 2048 Bit lange Schlüssel eingesetzt. Die Verschlüsselung oder die Signatur von Nutzdaten wird durch diese großen Schlüssel jedoch vollkommen unpraktikabel. Daher werden in der Praxis häufig Abkürzungen gewählt, die keine Beeinträchtigung der Sicherheit bedeuten.

Im Vergleich zur Abbildung 2.4 verschlüsselt Alice in Abbildung 2.6 nicht direkt den Text mit dem asymmetrischen Verfahren. Sie ermittelt zunächst zufällig einen beliebigen symmetrischen Schlüssel. Dieser Schlüssel wird verwendet um den Text zu verschlüsseln. Anschließend kodiert Alice diesen

Schlüssel mit dem Public Key von Bob und hängt ihn an die verschlüsselte E-Mail an. Bob kann zunächst den symmetrischen Schlüssel mit seinem Private Key dekodieren und anschließend hiermit die Nachricht entschlüsseln. Dieses Hybridverfahren verbindet die Vorteile beider Verfahren (symmetrisch und asymmetrisch) in einem. Der Schlüssel wird sicher mit einem Public Key Verfahren übertragen und die Verschlüsselung schnell und sicher mit einem symmetrischen Verfahren durchgeführt.

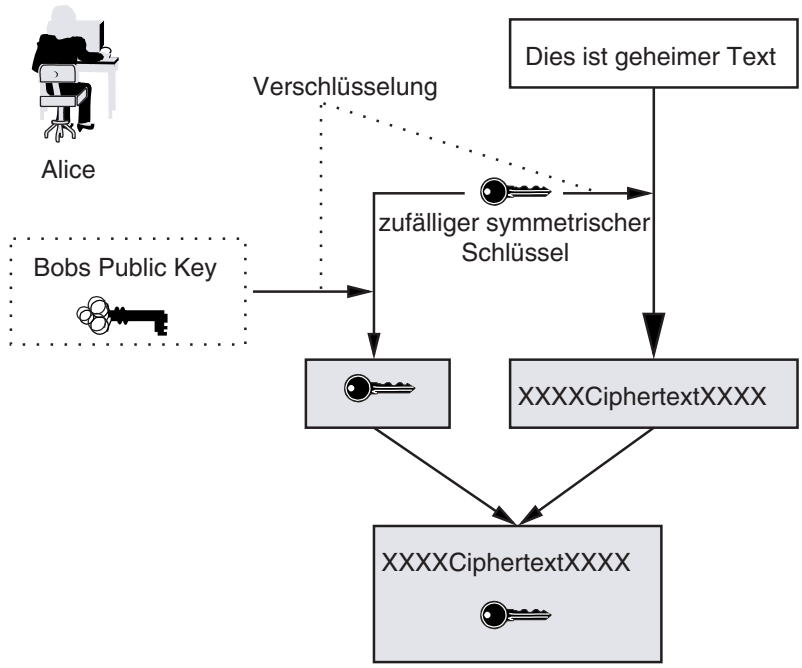


Abbildung 2.6 Das Hybridverfahren kombiniert symmetrische und asymmetrische Verschlüsselung

Ähnlich kann die Digitale Signatur durch den Einsatz eines kombinierten Verfahrens beschleunigt werden. Hierzu wird nicht das ganze Dokument von Bob mit dem privaten Schlüssel verschlüsselt wie in Abbildung 2.5 dargestellt, sondern zuvor mit einer Hash-Funktion eine kryptografische Prüfsumme ermittelt. Sie wird mit dem privaten Schlüssel verschlüsselt und zum Beispiel an die E-Mail angehängt. Der Empfänger kann den Inhalt ohne weitere Aktion lesen. Möchte er die Herkunft und die Integrität prüfen, so ermittelt er mit der bekannten Hash-Funktion ebenfalls die kryptografische Prüfsumme und entschlüsselt mit dem Public Key des Absenders die angehängte Prüfsumme. Stimmen die beiden Zahlen überein, so ist die Echtheit bestätigt. Stimmen die beiden Zahlen nicht überein, existieren folgende Möglichkeiten:

1. Das Dokument wurde während der Übertragung verändert. Daher ergibt die Hash-Funktion einen anderen Wert.
2. Das Dokument wurde nicht vom Absender Bob erzeugt. Der Erzeuger hatte keinen Zugriff auf den privaten Schlüssel von Bob und konnte daher nicht die Prüfsumme so verschlüsseln, dass sie mit dem Public Key von Bob dechiffriert werden konnte.
3. Bob hat seine Schlüssel gewechselt und verwendet einen anderen Private Key.

2.5.2 Public Key Schlüssellängen

Asymmetrische Verfahren sollten Schüssel mit ausreichenden Längen einsetzen. Was ist nun ausreichend? Dies hängt im Grunde von den zu schützenden Informationen ab. Alle eingesetzten asymmetrische Verfahren weisen Lücken in ihrem Algorithmus auf, die eine Abkürzung des Brute Force Angriffes erlauben. Daher müssen die Schlüssel etwa Faktor zehn größer sein als bei den symmetrischen Verfahren. Bei den asymmetrischen Verfahren sind die Schlüssellängen meist variabel und nicht fest von den Verfahren vorgeschrieben. Bereits im Februar 2000 gelang es eine Faktorisierung eines 512 Bit langen RSA Modulus durchzuführen (siehe Seite 409). Die Forscher vermuten, dass zehn Jahre später, also 2010, die Faktorisierung von 768 Bit langen Schlüsseln erfolgreich durchgeführt werden kann. Namhafte Kryptografen empfehlen jedoch schon eine geraume Zeit längere Schlüssellängen (siehe Seite 409). Ein Schüssel mit 768 Bit Länge (zum Beispiel Standard SSH Serverkey) sollte nur noch eingesetzt werden, um Informationen, deren Vertraulichkeit nur für wenige Minuten oder Stunden gesichert werden muss, zu verschlüsseln. Im Falle von der Secure Shell wird der Serverkey alle 60 Minuten neu generiert. Ansonsten sollten Schlüssellängen von 1024 Bit für einige Monate und 2048 Bit Schlüssel für einige Jahrzehnte ausreichen. Wenn die Vertraulichkeit von Informationen darüberhinaus sichergestellt werden muss, sollten noch längere Schlüssel verwendet werden.

2.5.3 RSA

RSA wurde von Ron **R**ivest, Adi **S**hamir und Leonard **A**dlmeman entwickelt. Dies war der erste komplette Public Key Algorithmus und bis heute auch der populärste. Über die Jahre konnte durch Kryptoanalyse weder seine Sicherheit nachgewiesen werden, noch eine Sicherheitslücke entdeckt werden. Das Diffie Hellmann Protokoll ist zwar älter und wurde bereits 1976

entwickelt, jedoch handelt es sich lediglich um ein Protokoll zum Schlüsselaustausch und nicht zur Verschlüsselung und Signatur. RSA unterstützt sowohl den Einsatz als Verschlüsselungsverfahren als auch zur Signatur.

Der Algorithmus beruht auf der Faktorisierung sehr großer Zahlen in ihre Primfaktoren. Die Berechung des Produkts zweier Primzahlen ist sehr einfach. Bei gegebenen Produkt die beiden Primzahlen zu berechnen, ist bei entsprechender Zahlengröße unmöglich. Der öffentliche und der private Schlüssel sind Funktionen eines Primzahlenpaares mit 100 bis 500 Stellen.

Die Implementierung von RSA in Hardware ist etwa um Faktor 1000 langsamer als DES. RSA war lange Zeit durch die Firma RSA patentiert. Jedoch wurde der Alorithmus im Jahre 2000 kurz vor Ablauf des Patentschutzes öffentlich zur Verfügung gestellt. RSA ist heute der Standard für öffentliche Kryptografie.

2.5.4 ElGamal

ElGamal kann wie RSA zur digitalen Signatur als auch zur Verschlüsselung eingesetzt werden. Seine Sicherheit beruht auf der Komplexität der Berechnung diskreter Logarithmen im finiten Feld.

Die Berechnung von ElGamal ähnelt sehr stark dem Diffie Hellman Schlüsselaustausch. In der Gleichung $y = g^x \bmod p$ sind y, g und p der öffentliche Schlüssel. x ist der private Schlüssel.

Auf Grund seiner Ähnlichkeit war, obwohl ElGamal selbst nicht patentiert war, dennoch der Algorithmus bis 1997 durch das Diffie Hellmann Patent geschützt.

2.5.5 DSA

Der Digital Signature Algorithm (DSA) wurde 1991 von dem National Institute of Standards and Technology (NIST) vorgeschlagen. Der Standard wurde als Digital Signature Standard (DSS) bezeichnet.

DSA wurde ursprünglich nur für die digitale Signatur entwickelt. Dieser Algorithmus sollte nicht in der Lage sein, eine Verschlüsselung durchzuführen. Jedoch besteht die Möglichkeit mit diesem Algorithmus eine ElGamal Verschlüsselung durchzuführen.

Die Sicherheit von DSA ist stark von der Schlüssellänge abhängig. Der Algorithmus ist erst ab einer Schlüssellänge von 1024 Bits als sicher einzustufen.

Die weltweite Verwendung des Algorithmus ist momentan problematisch. Es existiert ein weltweites Patent des Herrn Schnorr für den gleichnamigen Algorithmus. Dieses überschneidet sich mit dem DSA Algorithmus. Jedoch herrscht inzwischen die einhellige Meinung vor, dass das Schnorr Patent nur Anwendung bei bestimmten Smartcards findet und nicht auf die Implementierung in Software zu übertragen ist.

2.5.6 Diffie Hellman

Das Diffie Hellmann-Verfahren ist das erste öffentlich bekannte Public Key Verfahren. Es bietet nicht die Möglichkeit der Verschlüsselung oder Signatur, aber es löst das klassische Henne-Ei Problem bei der symmetrischen Verschlüsselung: Wie wird der Schlüssel übertragen? Daher wird das Verfahren häufig auch als Diffie Hellmann-Schlüsselaustausch bezeichnet. In Wirklichkeit wird jedoch kein Schlüssel ausgetauscht, sondern ein Schlüssel erzeugt.

Aus diesem Grund ist das Diffie Hellmann-Verfahren anfällig für Man-in-the-Middle-Angriffe. Bevor das Diffie Hellmann-Verfahren angewendet wird, ist für die Sicherheit zwingend eine Authentifizierung erforderlich. Hierfür können sehr gut die beschriebenen Public-Key-Verfahren eingesetzt werden.

Die Besonderheit des Verfahrens liegt in seinem möglichen Einsatz über unsichere Kanäle. Salopp ausgedrückt, erlaubt es das Diffie Hellmann-Verfahren, dass sich zwei Personen Alice und Bob öffentlich einige Zahlen zurufen, die von sämtlichen umstehenden Mathematikern und Kryptoanalytikern mitgehört werden dürfen. Dennoch sind Alice und Bob in der Lage sich auf eine geheime Zahl zu einigen, die allen weiteren Personen unbekannt ist und auch nicht errechnet werden kann.

Mathematisch formuliert einigen sich Alice und Bob zu Beginn auf eine große Primzahl p und eine weitere zufällige Zahl z. Diese Informationen sind öffentlich. Nun wählen Alice und Bob jeder persönlich eine weitere zufällige geheime Zahl a und b . Alice wie Bob berechnen nun die Potenz z^a bzw. z^b. Anschließend berechnen Alice und Bob den Rest einer Division (Modulo) durch p. Alice hat nun berechnet $A=z^a \bmod p$. Bob hat berechnet $B=z^b \bmod p$. Nun tauschen Alice und Bob A und B aus. Dieser Austausch erfolgt nun wieder öffentlich. Bisher sind öffentlich bekannt A, B, p und z. Ein direkter Rückschluss auf die Zahlen a und b ist nicht möglich. Durch die Anwendung der Modulo-Operation können dies unendlich viele Zahlen sein, die einzeln ausprobiert werden müssen. Alice und Bob führen nun dieselben Operationen ein weiteres Mal durch. Alice berechnet also $B^a \bmod p$ und Bob berechnet $A^b \bmod p$. Diese beiden Zahlen sind identisch, da $B^a \bmod p = (z^b)^a \bmod p = z^{(ab)}$

mod p = $(z^a)^b$ mod p = A^b mod p ist. Da die weiteren Anwesenden im Raum nicht die Zahlen b und a kennen, können sie diese Operation nicht durchführen.

Damit dies nicht rein trockene und schwer nachvollziehbare Theorie bleibt, soll dies kurz mit echten (kleinen) Zahlen nachvollzogen werden.

Stellen wir uns vor, Alice und Bob wählen als Primzahl 479 und als Zufallszahl 5. Diese beiden Zahlen sind öffentlich bekannt. Anschließend wählt Alice als geheime Zahl 8 und Bob 13. Nun wird gerechnet:

```
Alice: 5 ^  8 % 479 = 390625     % 479 = 240
Bob  : 5 ^ 13 % 479 = 1220703125 % 479 = 365
```

Diese Zahlen werden nun ausgetauscht und Alice und Bob führen die Operationen ein weiteres Mal durch.

```
Alice: 365 ^  8 % 479 = 315023473396125390625 % 479 = 88
Bob  : 240 ^ 13 % 479 = 876488338465357824000000000000 % 479 = 88
```

Soll dies nachvollzogen werden, so kann dies mit dem Befehl bc unter Linux erfolgen. bc ist ein Rechenprogramm mit beliebiger Genauigkeit. Ein Taschenrechner würde bei den obigen Zahlen bereits Rundungsfehler einführen. Das folgende Programm erlaubt die Berechnung des Diffie Hellmann Schlüsselaustausches.

```
primzahl = 479
zufall   = 5
alice    = 8
bob      = 13

alicepot = zufall ^ alice
alicemod = alicepot % primzahl
print "\nAlice: ",zufall," ^ ",alice," % ",primzahl," = ",alicemod,"\n";

bobpot   = zufall ^ bob
bobmod   = bobpot % primzahl
print "\nBob: ",zufall," ^ ",bob," % ",primzahl," = ",bobmod,"\n";

print "Austausch!\n"
alicepot = bobmod ^ alice
print bobmod," ^ ",alice," = ";alicepot
aliceres = alicepot % primzahl
print "Alice erhält als Ergebnis  = ",aliceres,"\n"

bobpot   = alicemod ^ bob
print alicemod," ^ ",bob," = ";bobpot
bobres   = bobpot % primzahl
```

```
print "Bob erhält als Ergebnis  = ",bobres,"\n"

quit
```

Warum ist dieses Verfahren nun sicher? Um als Außenstehender ebenfalls das Ergebnis berechnen zu können ist es erforderlich die geheime Zahl von Alice oder Bob zu ermitteln.

Vernachlässigen wir zunächst die Bildung des Modulo. Dann reduziert sich dass Problem auf $A = z^a$. Sind die Zahlen A und a bekannt, so ist dieses Problem bereits relativ aufwändig in der Lösung. Es muss der Logarithmus von A zur Basis Z berechnet werden.

Zusätzlich erschwert der Modulo die Berechnung. Stellen Sie sich ein kleines Zahlenspiel vor. Sie sollen eine Zahl erraten. Sie wissen, dass diese Zahl den Rest 15 bei einer Division durch 30 hat. Die möglichen Lösungen sind: 15, 45, 75, 105, 135, 165, 195 und so weiter Um die richtige Zahl zu erraten, müssen Sie sämtliche Zahlen durchprobieren. In der Kombination mit dem Logarithmus Problem und den langen Schlüssellängen ist dieses Problem unlösbar.

ACHTUNG

Bei der Verwendung des Diffie Hellmann-Verfahrens bei dem IKE Protokoll sind die beiden öffentlichen Zahlen p (Primzahl) und z (Generator) allgemein festgelegt. Dadurch ist es nicht erforderlich, sie zu übertragen. Die beiden Kommunikationspartner bestimmen lediglich eine sogenannte MODP Gruppe mit einer bestimmten Länge (768, 1024, 1536, 2048, 3072, 4096, 6144 oder 8192 Bits) die dann die entsprechenden Zahlen vorschreibt (*http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ike-modp-groups-05.txt*).

2.6 Hash-Funktion

Hash-Algorithmen sind unter vielen Namen bekannt. Sie werden auch als Kompressionsfunktion, Message Digest, Fingerabdruck und kryptografische Prüfsumme bezeichnet. Die letzte Bezeichnung gibt ihren Sinn am besten wieder. Es handelt sich bei Ihnen um mathematische Funktionen, die aus einer Eingabe meist variabler Länge eine Ausgabe bestimmter Länge erzeugen. Dieselbe Eingabe erzeugt immer eine identische Ausgabe. Hierbei versuchen die meisten Hash-Algorithmen eine Gleichverteilung der Ausgaben zu erzeugen. Dass bedeutet, dass bei vier Eingaben AAAA, AAAB, AAAC und ZZZZ, die Ausgaben über den gesamten Bereich der möglichen Hash-Ausgaben gleichverteilt sind. Dies ist vergleichbar mit Fingerabdrücken. Werden die Fingerabdrücke von drei Europäern und einem Nordamerikaner aufgenom-

men, so zeigen sie nicht die nähere Verwandschaft der drei Europäer. Ein Rückschluss vom Fingerabdruck auf die Person ist ohne Vergleich nicht möglich. Die »Erzeugung« eines zweiten Fingers mit demselben Abdruck e ist ebenfalls sehr unwahrscheinlich. Genauso arbeiten die Hash-Funktionen in der Kryptografie. Ist nur das Ergebnis der Hash-Funktion bekannt, so ist ein Rückschluss auf die Eingabe unmöglich. Die Erzeugung einer zweiten Eingabe mit demselben Hash-Wert ist sehr schwierig.

Allgemein werden an Hash-Funktionen die folgenden Forderungen gestellt:

1. Die Hash-Funktion soll schnell aus einer beliebigen Eingabe einen Hash-Wert definierter Größe erzeugen.
2. Eine Umkehrung der Funktion, das Berechnen der Eingabe aus dem Hash-Wert, soll unmöglich sein.
3. Es soll möglichst schwierig sein, eine zweite verschiedene Eingabe zu erzeugen, die denselben Hash-Wert erzeugt.

Weist der Hash-Algorithmus im dritten Punkt Schwächen auf, so bezeichnet man diese als Kollision und den Angriff als Birthday-Attack (eine mathematische Darstellung des Birthday-Problems findet sich auf *http://mathworld.wolfram.com/BirthdayProblem.html*).

Hashes werden für Integritätsprüfungen und Authentifizierungen eingesetzt. Die Integritätsprüfung ermittelt die kryptografische Prüfsumme eines Datums und kann diese mit einer zu einem früheren Zeitpunkt ermittelten Prüfsumme vergleichen. Stimmen sie überein, so wurde das Datum nicht modifiziert. Für die Authentifizierung wird das Klartext Kennwort einer Person in seinen Hash umgewandelt und abgespeichert. Anschließend kann bei jeder Anmeldung der Person aus dem eingegebenen Klartextkennwort sofort der Hash ermittelt werden und mit dem abgespeicherten Hash verglichen werden. Stimmen sie überein, so wurde das richtige Kennwort eingegeben.

Eine besondere Form ist der Hash Message Authentication Code (HMAC). Dieser HMAC wird verwendet um die Integrität einer Nachricht zu garantieren. Dazu wird die Hash-Funktion auf die Nachricht und einen geheimen Schlüssel angewendet. Nur die Person, die den geheimen Schlüssel kennt, kann die Hash-Funktion ausführen und die Integrität überprüfen. Für diesen Zweck werden die Hash-Funktionen bei den IPsec Protokollen eingesetzt.

Erfreulicherweise gelten diese Verfahren als grundsätzlich kollisionsfrei, selbst wenn beim eingesetzten Algorithmus Kollisionen nachgewiesen wurden.

Im Zusammenhang mit IPsec kommen üblicherweise nur zwei Hash-Funktionen zum Einsatz: MD5 und SHA.

2.6.1 MD5

MD5 ist eine verbesserte Variante des MD4 Hash-Algorithmus. Der MD4 Algorithmus wird zum Beispiel vom Microsoft Windows NT Betriebsystem zur Speicherung der Kennworte eingesetzt. Es erzeugt einen 128 Bit langen Hash. Dies erfolgt in mehreren Durchläufen. Bei dem MD5 Verfahren wurden bereits Kollisionen nachgewiesen. Diese haben keine direkte praktische Auswirkung auf die Sicherheit von MD5 als HMAC. Jedoch schreibt Bruce Schneier in seinem Buch »Angewandte Kryptografie«, dass er der Verwendung mit Vorsicht gegenübersteht.

2.6.2 SHA

Der Secure Hash Algorithm ist für die Verwendung beim DSA Algorithmus entwickelt worden. Er erzeugt klassischerweise einen 160 Bit Hash. Dies ist wesentlich mehr als bei MD5. Es existieren jedoch inzwischen weitere SHA Implementierungen die auch eine Verschlüsselung mit 256, 384 und 512 Bit ermöglichen.

Es gibt keine bekannten kryptoanalytischen Angriffe auf SHA. Aufgrund des längeren Hashes wird er Brute Force Angriffen länger widerstehen könnnen.

3 VPN Protokolle

Es existieren mehrere verschiedene Protokolle, die für den Aufbau eines VPNs genutzt werden können. Dieses Buch nutzt in erster Linie die IPsec Protokolle. Daher beschäftigt sich dieses Kapitel auch vorrangig mit diesen Protokollen. Da jedoch auch das L2TP Protokoll immer stärker in Kombination mit IPsec genutzt wird, wird auch dieses Protokoll kurz vorgestellt. Das PPTP Protokoll wird heute nur noch selten eingesetzt. Eine Unterstützung unter Linux ist möglich, aber nicht Thema dieses Buches.

3.1 Einleitung

Der Bedarf, Daten verschlüsselt über Computernetze zu übertragen, ist so alt wie die Computernetze selbst. Die eigentlichen Netzwerkprotokolle, wie zum Beispiel das Internet Protokoll IP, bieten diese Funktion jedoch nicht. Daher gibt es eine Vielzahl von verschiedenen Protokollen, die dies basierend auf den Netzwerkprotokollen ermöglichen. Einige dieser Protokolle haben einen höheren Bekanntheitsgrad erreicht (zum Beispiel das Secure Shell Protokoll und die Secure Socket Layer (SSL)) und wurden standardisiert (zum Beispiel Transaction Layer Security (TLS)). Meist sind die Protokolle für eine ganz bestimmte Anwendung entwickelt worden. So wurde das SSL Protokoll ursprünglich für den Schutz der HTTP-Kommunikation zwischen einem Browser und einem Webserver entworfen.

Die Protokolle, die in einem VPN eingesetzt werden, unterscheiden sich in ihrer Universalität meist von den anwendungsorientierten Protokollen. Sie ermöglichen es sämtliche Daten und alle Kommunikationsströme unabhängig vom verwendeten Protokoll gemeinsam zu verschlüsseln und ihre Vertraulichkeit und Integrität zu garantieren.

Üblicherweise werden dazu die zu übertragenden Nutzpakete komplett im VPN Protokoll abgekapselt. Um dies zu unterstützen muss das VPN-Protokoll in der Lage sein zum Beispiel IP-Pakete zu übertragen. Das SSL-Protokoll kann lediglich die Pakete einer beliebigen TCP Verbindung übertragen. SSH ist nur in der Lage ASCII Verbindungen zu ermöglichen und einzelne TCP Verbindungen zu tunneln.

Erste Versuche des Aufbaus eines VPNs unter UNIX oder Linux basierten häufig auf der Secure Shell. Hierzu wird eine Secure-Shell-Verbindung zwischen zwei UNIX Systemen aufgebaut. Über diese Verbindung können nun

ASCII Daten ausgetauscht werden. Anschließend wird auf beiden Seiten der Point-to-Point-Protokoll-Daemon (PPP-Daemon) gestartet. Der PPP-Daemon kommuniziert nun über die Secure-Shell-Verbindung und überträgt Netzwerkpakete. Beim Start des PPP-Daemon wurde automatisch auf beiden Systemen eine zusätzliche Netzwerkkarte `ppp0` initialisiert. Alle Pakete, die an diese Karte geschickt werden, werden nun vom PPP-Daemon über die Secure-Shell-Verbindung an den zweiten Rechner geschickt, dort entgegen genommen und über die `ppp0` Karte wieder zur Verfügung gestellt. Existieren nun entsprechende Routingeinträge auf den Rechnern, so kann diese Verbindung als einfaches VPN genutzt werden. Das Linux VPN HowTo (*http://www.tldp.org/HOWTO/VPN-HOWTO/*) beschreibt einen derartigen Aufbau.

Das erste in großem Stil eingesetzte VPN Protokoll war das Point-to-Point-Tunneling-Protokoll (PPTP). Dieses Protokoll wurde unter anderem von Microsoft entwickelt und steht in vielen Microsoft Windows Betriebsystemen (Windows 9x, ME, NT, 2000) zur Verfügung.

Bei genauer Betrachtung ähnelt das PPTP Protokoll der Variante mit SSH/PPP. Das PPTP Protokoll ermöglicht den Aufbau eines GRE (Generic Routing Encapsulation) Tunnels zwischen zwei Rechnern. Auf der Basis dieses GRE Tunnels wird auf beiden Seiten der PPP Daemon gestartet. Der PPP Daemon übernimmt die IP Pakete und verpackt sie in PPP Pakete, die über den GRE Tunnel transportiert werden. Das PPTP Protokoll ist aber im Gegensatz zum SSH Protokoll beim oben beschriebenen Tunnel in keiner Weise für die Verschlüsselung oder Authentifizierung zuständig. Es überträgt lediglich die für die PPP Sitzung zu verwendenden IP Adressen. Die Authentifzierung wird vom PPP Daemon durchgeführt. Hierbei erfolgt die Anmeldung mit einem Benutzernamen und einem Kennwort.

Um dieses Protokoll dennoch als Basis für ein VPN einsetzen zu können wurde von Microsoft das PPP-Protokoll erweitert. Für eine sichere Authentifzierung wurde von Microsoft ein Challenge/Response Authentication Protokoll (CHAP) in Form von MS-CHAP implementiert. Das wurde, nachdem einige Fehler bekannt wurden, als MS-CHAPv2 überarbeitet vorgestellt. Zusätzlich soll ein VPN die Vertraulichkeit der übertragenen Pakete sicherstellen. Hierzu wurde von Microsoft der PPP-Daemon um das Microsoft-Point-to-Point-Encryption-Protokoll (MPPE) erweitert. Dieses Protokoll erlaubt die Verschlüsselung der Pakete mit 40 oder 128 Bit. Der hierbei verwendete Schlüssel wird vom eingesetzten Kennwort abgeleitet und typischerweise alle 256 Pakete neu ausgetauscht.

Die wesentlichen Schwäche des PPTP-Protokolls liegt in der Authentifizierung, aber es wurden auch weitere verschiedene Schwächen im MS-CHAP

Protokoll nachgewiesen. Bruce Schneier hat diese in mehreren Dokumenten aufgeführt (*http://www.counterpane.com/pptp.html*). Das größte Problem stellt die Authentifizierung mit einem Kennwort dar. Kennworte werden üblicherweise nicht zufällig erzeugt und sind daher von geringer Entropie. Es ist wesentlich einfacher ein Kennwort zu raten, als einen Brute Force-Angriff auf einen 128 Bit langen Schlüssel durchzuführen.

Heute wird das PPTP Protokoll nur noch selten für die Implementierung einer neuen VPN Lösung eingesetzt. Selbst Microsoft hat erkannt, dass das PPTP Protokoll kein Vertrauen mehr genießt und einen freien IPsec/L2TP Client für die Betriebssysteme veröffentlicht, die bisher keine IPsec Unterstützung bieten (*http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/l2tpclient.asp*).

3.2 IPsec

Bei der Entwicklung des Protokolls IP Version 4 wurden Sicherheitsaspekte vernachlässigt. Die Betonung wurde auf Geschwindigkeit und Robustheit der Anwendung gelegt. Bei der Entwicklung des IP-Protokolls Version 6 sollten derartige Fehler von Beginn an vermieden werden. Dies führte zur Entwicklung der IPsec Protokolle, die im Anschluss auf das IP Protokoll Version 4 portiert wurden.

Die IPsec-Protokolle können die Vertraulichkeit, Authentizität und Integrität der übertragenen Daten garantieren. Hierfür stehen das Authentication Header-Protokoll (AH) und das Encapsulated-Security-Payload-Protokoll (ESP) zur Verfügung. Diese Protokolle bieten zusätzlich einen Schutz vor Replay Angriffen. Dabei werden mit einem Schiebefenster automatisch wiederholt gesendete Pakete erkannt und verworfen.

Für die Verschlüsselung, Authentifizierung und die Integritätsüberprüfung werden symmetrische Schlüssel benötigt. Sie müssen vorher ausgehandelt werden. Um dies automatisch zu ermöglichen wurde das Internet Key Exchange-Protokoll (IKE) entwickelt. IKE authentifiziert die Kommunikationspartner mit geeigneten Mitteln und verhandelt die zu verwendenen Algorithmen und Verwaltungsinformationen. Hierzu erzeugt das IKE Protokoll auch mit dem Diffie Hellmann-Verfahren symmetrische Schlüssel in der benötigten Anzahl.

Bei der Entwicklung des IPsec Protokolls wurden viele Aspekte des IPv4 Protokolls vernachlässigt. Dies ist verständlich, da die IPsec Protokolle zunächst für IPv6 entwickelt wurden. IPv6 wird wahrscheinlich keine Network Address Translation (NAT) benötigen, da der Adressraum mit 128 Bit langen

Netzwerkadressen wesentlich größer ist als der IPv4 Adressraum mit 32 Bit langen Netzwerkadressen. Die IPsec Protokolle unterstützen daher kein NAT. Neue Erweiterungen der IPsec Protokolle erlauben es diese erneut in UDP Paketen zu kapseln. Da das UDP Protokoll ein NAT unterstützt, erben die IPsec Protokolle diese Funktion.

Eine weitere Erweiterung der IPsec Protokolle stellt die DHCP-over-IPsec Funktion dar. Hierbei besteht die Möglichkeit über den IPsec Tunnel IP Adressen auszutauschen, die anschließend für die Verbindung genutzt werden.

Beide Protokollerweiterungen werden am Ende dieses Kapitels vorgestellt.

3.2.1 Integrität und Authentifizierung

Die IPsec Protokolle stellen die Integrität der übertragenen Pakete sicher und authentifizieren die Quelle der Pakete. Hierzu werden Hash-Algorithmen (siehe Abschnitt 2.6, »Hash-Funktion«) im Hash-Message-Authentication-Code (RFC 2104) Verfahren eingesetzt. Dabei wird der Hash-Wert aus einem geheimen, nur den Kommunikationspartnern bekannten, Schlüssel und den zu schützenden Daten ermittelt. Das Ergebnis, der HMAC, stellt gewissermaßen eine Signatur dar und erlaubt es, die Herkunft und die Integrität des Paketes zu testen.

Die IPsec Protokolle verwenden einen HMAC mit einer Länge von 96 Bits. Dieser HMAC kann mit den verschiedensten Protokollen ermittelt werden. Die IPsec Standards verlangen die Implementierung von HMAC-MD5-96 (RFC 2401) und HMAC-SHA-1-96 (RFC 2404). Hierbei wird bei MD5 ein geheimer Schlüssel von 128 Bit und bei SHA-1 ein geheimer Schlüssel von 160 Bit eingesetzt. Aus diesem Schlüssel und den zu schützenden Daten berechnet der Algorithmus einen HMAC. Dieser HMAC ist bei MD5 128 Bit lang und bei SHA-1 160 Bit lang. Die höchstwertigen 96 Bit dieses Schlüssels werden tatsächlich als HMAC (auch ICV Integrity Check Value) von den IPsec Protokollen verwendet (siehe Abbildung 3.1).

Der geheime Schlüssel und der eingesetzte HMAC Algorithmus werden entweder vom Administrator von Hand festgelegt (manuell verschlüsselte Verbindungen) oder das IKE Protokoll ermittelt diese Parameter und erzeugt die entsprechenden Security Associations.

Seit einigen Jahren werden zusätzliche HMAC Varianten entwickelt. So existieren Drafts, die die Verwendung von SHA-2 mit 256, 384 und 512 Bit Länge beschreiben. Weitere teilweise für IPsec weniger gebräuchliche HMAC Verfahren sind HMAC-RIPEMD-160, HMAC-PANAMA und HMAC-TIGER.

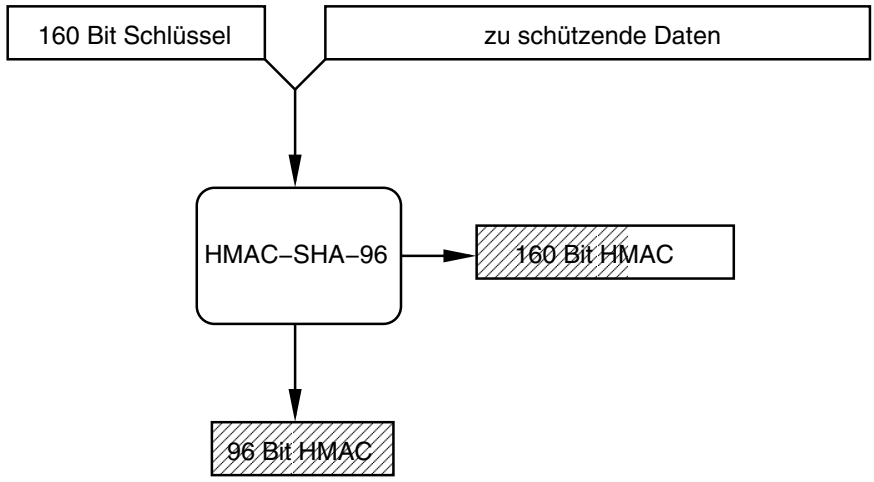


Abbildung 3.1 Der HMAC wird aus einem geheimen Schlüssel und den zu sichernden Daten ermittelt. Verwendet werden die 96 höchstwertigen Bits.

3.2.2 Verschlüsselung

Die IPsec Protokolle stellen die Vertraulichkeit der übertragenen Daten durch eine symmetrische Verschlüsselung der Informationen sicher. Hierbei werden die Daten mit einem geheimen Schlüssel im Cipher-Block-Chaining Modus mit symmetrischen Verfahren verschlüsselt. Lediglich die Kommunikationspartner verfügen über den geheimen Schlüssel und können so die Vertraulichkeit der Daten garantieren.

Die IPsec Protokolle können unterschiedliche Verschlüsselungsverfahren mit unterschiedlichen Schlüssellängen einsetzen. Die IPsec Standards verlangen die Unterstützung von NULL (RFC 2410), CBC-DES (RFC 1829) und CBC-DES mit expliziten IV (RFC 2405).

Die DES Verschlüsselung verwendet jedoch nur einen 64 Bit langen Schlüssel. 8 Bits dieses Schlüssels dienen lediglich als Paritätsinformation. Daher beträgt die wirksame Länge des Schlüssels nur 56 Bit. Aus diesem Grund werden heute weitere CBC-Verschlüsselungsverfahren von den meisten IPsec Implementierungen unterstützt (RFC 2451): CAST-128, RC5, IDEA, Blowfish und 3DES. Zusätzliche Drafts beschreiben die Verwendung von AES oder RC6.

Sämtliche eingesetzten Verschlüsselungsverfahren sind Block-Ciphern. Das bedeutet, dass diese die zu schützenden Daten in ganzen Blöcken verarbeiten. Die Blocklänge variiert und hängt vom Verfahren ab. Da in den seltens-

ten Fällen die zu verschlüsselnden Daten exakt ein Vielfaches der Blocklänge sind, ist ein Auffüllen der Daten bis zur nächsten Blockgrenze erforderlich. Dies wird als Padding bezeichnet.

Da es sich in allen Fällen um monoalphabetische Verschlüsselungsverfahren (siehe Abschnitt 2.4, »Cipher Block Chaining (CBC)«) handelt, ist es erforderlich, das Cipher Block Chaining-Verfahren einzusetzen, um einen Angriff durch eine Frequenzanalyse zu verhindern. Dabei wird ein Initialisierungsvektor verwendet, der vor jeder Verschlüsselung mit dem Klartextblock exklusiv verknüpft wird. Der erste Initialisierungsvektor wird zunächst zufällig ermittelt. Anschließend wird ein verschlüsselter Block als Vektor des nächsten Blockes genutzt. Jedes verschlüsselte IPsec Paket enthält einen Initialisierungsvektor. Um den Zufallszahlengenerator nicht übermäßig zu belasten, wird meist auch bei allen folgenden Paketen als Initialisierungsvektor der letzte verschlüsselte Block des vorhergehenden Paketes verwendet. Getreu dem Motto »Ein Bild sagt mehr als tausend Worte.« gibt die Abbildung 3.2 dies graphisch wieder.

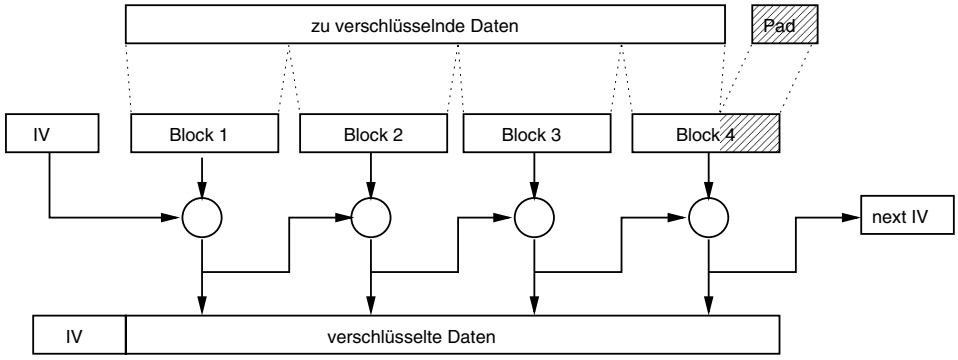


Abbildung 3.2 Im CBC Modus werden die zu verschlüsselnden Blöcke miteinander verknüpft. Der erste Block verwendet hierzu einen expliziten Initialisierungsvektor. Vor der Verschlüsselung ist unter Umständen ein Padding erforderlich.

3.2.3 Anti Replay Schutz

Die IPsec Protokolle implementieren einen optionalen Anti Replay Service. Hierzu muss der Absender jedes Paket mit einer (monoton) steigenden Sequenznummer versehen. Der Empfänger kann diese Nummer zum Schutz vor Replay Angriffen nutzen. Hierzu verwendet der Empfänger ein Schiebefenster bestimmter Größe (typisch sind Größen von 32 oder 64). Erhält der Empfänger nun ein Paket, dessen Sequenznummer links außerhalb des Fensters liegt, so wird dieses Paket sofort verworfen. Befindet sich die Sequenz-

nummer im Fenster und wurde das Paket bereits erhalten, so wird es ebenfalls verworfen. Lediglich neue Pakete im oder rechts vom Fenster werden nach erfolgreicher Authentifizierung akzeptiert und entschlüsselt. Befindet sich die Sequenznummer des akzeptierten Paketes rechts außerhalb des Fensters, so wird nach der erfolgreichen Authentifizierung das Fenster soweit nach rechts verschoben, dass sich dieses Paket nun auch innerhalb des Fensters befindet (siehe Abbildung 3.3).

Ein Angreifer, der die korrekten Pakete einer VPN Verbindung aufzeichnet und sie zu einem späteren Zeitpunkt wieder abspielt um einen Denial-of-Service zu erreichen, kann so erfolgreich daran gehindert werden. Würden nicht diese Sequenznummern eingesetzt werden, so müsste der Empfänger alle Pakete entschlüsseln, da er sie erfolgreich authentifizieren würde. Sie stammen ja alle vom korrekten Absender und wurden vom Angreifer nicht verändert. Nun werden veraltete und bereits erhaltene Pakete vor dem Test der Authentifzierung bereits verworfen. Die aufwändige Prüfung der Integrität und die Entschlüsselung des Paketes braucht nicht durchgeführt zu werden.

Das Fenster sollte nicht zu klein gewählt werden. Die Pakete können auf Grund der Struktur des Internets in unterschiedlicher Reihenfolge beim Empfänger eintreffen. Wird das Fenster zu klein gewählt, so sind häufige Neuübertragungen mit neuen Sequenznummern erforderlich. Die Mindestgröße beträgt laut Standard 32 Bit.

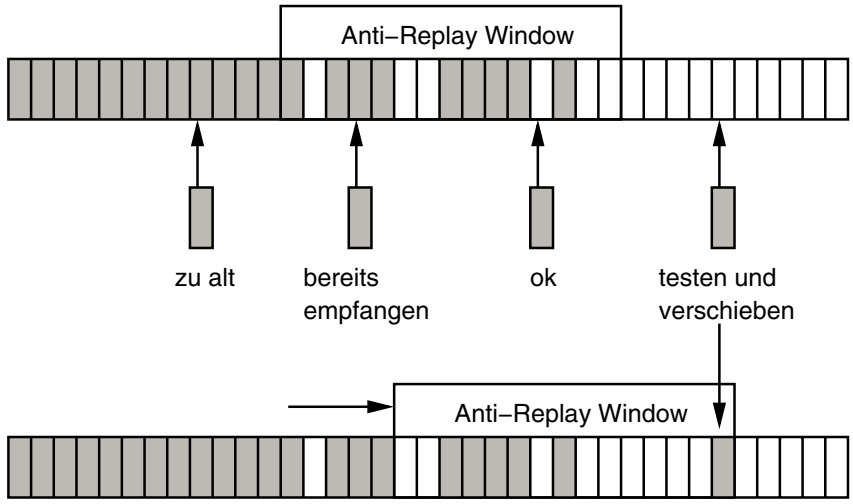


Abbildung 3.3 Das Anti-Replay Fenster schützt vor Replayangriffen mit alten aufgezeichneten Paketen

3.2.4 Tunnel- und Transport Modus

Die IPsec Protokolle unterstützen zwei verschiedene Modi zur Übertragung der Informationen: Tunnel Modus und Transport Modus. Im Transport Modus wird lediglich das Upper-Layer Protokoll (zum Beispiel TCP oder UDP) durch das IPsec Protokoll geschützt. Dabei wird der IPsec-Header zwischen dem IP-Header und dem Header des höheren Protokolls eingeschoben. Das *Next-Header* Feld im IPsec-Header enthält dann die Nummer des höheren Protokolls. Dieser Modus kann jedoch nur zum Einsatz kommen, wenn die beiden kommunizierenden Rechner direkt die Pakete verschlüsseln.

Häufiger wird daher der Tunnel Modus eingesetzt, bei dem die Rechner als VPN-Gateway fungieren und komplette IP-Pakete einschließlich ihrer Header schützen. Dabei wird das gesamte IP-Paket in einem IPsec-Header eingefasst, mit einem neuen IP-Header versehen und zum gegenüberliegenden Endpunkt des IPsec-Tunnels transportiert, wo es ausgepackt und mit seinem originalen IP-Header weitergeschickt wird. Das *Next-Header* Feld des IPsec-Headers enthält hier die Zahl 4 für das Protokoll IPv4 (siehe `/etc/protocols`).

Die Konstruktion der Header ist auch in der Abbildung 3.4 schematisch nochmals dargestellt.

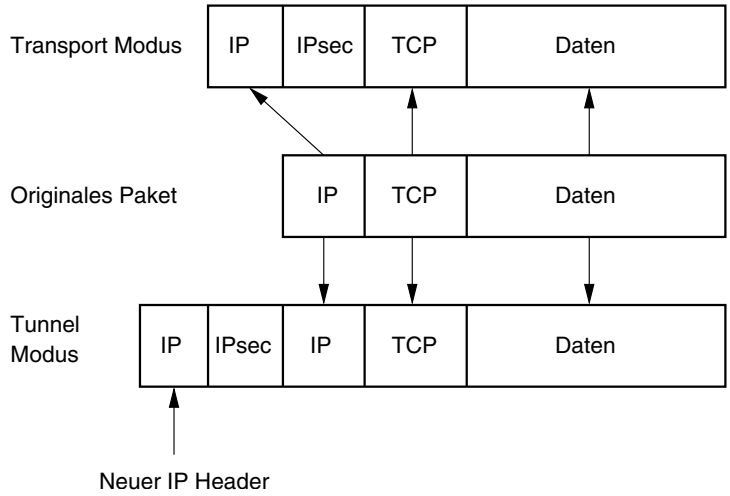


Abbildung 3.4 Im Tunnel Modus wird das gesamte IP-Paket durch einen IPsec-Header und einen neuen IP-Header gekapselt

3.2.5 Authentication Header – AH

Das Authentication Header Protokoll (IP Protokoll 51) stellt die Authentifizierung und die Integrität der IP Pakete sicher. Das Authentication Header Protokoll wird in dem RFC 2402 beschrieben. Dieser löst den älteren RFC 1826 ab.

Für seine Funktion setzt das AH Protokoll einen Integrity-Check-Value ein. Hierbei handelt es sich um die 96 höchstwertigen Bits eines Hash-Message Authentication Codes. Üblicherweise werden als HMAC sowohl HMAC-MD5-96 und HMAC-SHA-96 unterstützt. Neuere IPsec Varianten setzen jedoch auch HMAC-SHA-256-96, HMAC-SHA-384-96, HMAC-SHA-512-96 und HMAC-RIPEMD-160-96 ein. Als zusätzlichen Schutz versieht das AH Protokoll jedes Paket mit einer Sequenznummer, die vom Empfänger zum Schutz vor Replay-Angriffen genutzt werden kann.

Der Header des AH Protokolls ist in Abbildung 3.5 dargestellt.

Der Header enthält folgende Informationen:

- **Next Header** Dieses Feld (8 Bit) gibt das Protokoll der im Paket übertragenen Informationen an. Handelt es sich um ein Paket im Transport Modus, so befindet sich hier die Protokollnummer des entsprechenden höheren Protokolls, zum Beispiel 6 (TCP) und 17 (UDP). Wird gleichzeitig auch das ESP Protokoll eingesetzt, so befindet sich hier die Nummer 50 (ESP). Im Tunnel Modus ist hier bei Verwendung von IPv4 die Nummer 4 zu finden.
- **Payload Length** Hier wird die Länge des Headers in 8 Bit angegeben. Der Name Payload Length ist daher irreführend. Außerdem handelt es sich um einen IPv6 Header. Daher wird die Headerlänge in Vielfachen von 32 Bit angegeben, nachdem zuvor 64 Bit abgezogen wurden. Bei einem Header aus 3*32 Bit plus 96 Bit ICV ergibt sich 192 Bit Gesamtlänge. Abzüglich 64 Bit bleiben 128 Bit geteilt durch 32 Bit ergibt einen Wert von 4 für die Payload Length (siehe RFC 2402).
- **Reserved** Dieses 16 Bit Feld ist für zukünftige Zwecke reserviert und muss mit Nullen aufgefüllt werden.
- **Security Parameter Index (SPI)** Diese 32 Bit Zahl identifiziert in Kombination mit der Ziel-IP Adresse und dem IPsec Protokoll (AH oder ESP) eindeutig die Security Association. Die Zahl 0 ist für interne Verwendung reserviert. Die Zahlen 1 bis 255 (0x1-0xff) sind für die Verwendung durch die Internet Assigned Numbers Authority (IANA) reserviert.

- **Sequence Number** Die 32 Bit lange Sequenznummer ist eine monoton steigende Nummer, die vom Absender jedem Paket zugewiesen wird. Der Empfänger kann diese Nummer nutzen, um sich vor Replay-Angriffen zu schützen.
- **Integrity Check Value (ICV)** In diesem 96 Bit langen Feld werden die Authentifizierungsdaten gespeichert. Diese Daten können mit den verschiedenen HMAC Verfahren erzeugt werden. In allen Fällen werden nur die 96 höchstwertigen Bits des ermittelten Hashes hier abgespeichert.

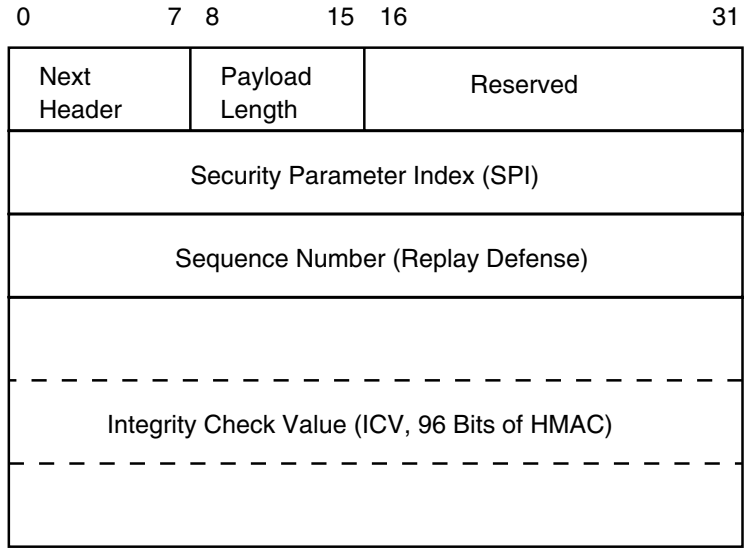


Abbildung 3.5 Das Authentication Header Protokoll hat einen 24 Byte langen Header

Das Authentication Header Protokoll sichert das gesamte IP-Paket. Das bedeutet, dass bei der Berechnung des ICVs nicht nur der AH Header und die gekapselten Informationen herangezogen werden, sondern auch der äußere IP-Header berücksichtigt wird. Dieser äußere IP-Header fließt in die Berechnung des ICV mit ein. Nicht berücksichtigt werden lediglich die folgenden Felder des IPv4-Headers: Type of Service (TOS), Flags, Fragment Offset, Time to Live (TTL) und Header Checksum. Diese Felder werden für die Berechnung des ICV auf Null gesetzt. Im Falle von IPv6 werden die folgenden Felder nicht berücksichtigt: Class, Flow Label, Hop Limit.

ACHTUNG

Die Einbeziehung des äußeren IP-Headers und damit auch der IP Adressen in die Berechnung des ICV schließt die gleichzeitige Verwendung von Network Address Translation (NAT) aus. Eine Modifikation der IP Adressen nach Berechnung des ICV würde zu dessen Ungültigkeit führen.

3.2.6 Encapsulated Security Payload – ESP

Das Encapsulated Security Payload Protokoll (IP Protokoll 50) stellt die Authentifizierung, die Integrität und die Vertraulichkeit der IP Pakete sicher. Das ESP Protokoll wird in dem RFC 2406 beschrieben. Dieser löst den älteren RFC 1827 ab.

Für seine Funktion setzt das ESP Protokoll einen Integrity Check Value ein. Hierbei handelt es sich um 96 höchstwertige Bits eines Hash-Message Authentication Codes. Üblicherweise werden als HMAC sowohl HMAC-MD5-96 und HMAC-SHA-96 unterstützt. Neuere IPsec Varianten setzen jedoch auch HMAC-SHA-256-96, HMAC-SHA-384-96, HMAC-SHA-512-96 und HMAC-RIPEMD-160-96 ein. Als zusätzlichen Schutz versieht das AH Protokoll jedes Paket mit einer Sequenznummer, die vom Empfänger zum Schutz vor Replay-Angriffen genutzt werden kann. Um die Vertraulichkeit zu gewährleisten werden die zu schützenden Informationen zusätzlich verschlüsselt.

Der Header des ESP Protokolls ist in Abbildung 3.6 dargestellt.

Der Header enthält folgende Informationen:

- **Security Parameter Index (SPI)** Diese 32 Bit Zahl identifiziert in Kombination mit der Ziel-IP Adresse und dem IPsec Protokoll (AH oder ESP) eindeutig die Security Association. Die Zahl 0 ist für die interne Verwendung reserviert. Die Zahlen 1 bis 255 (0x1-0xff) sind für die Verwendung durch die Internet Assigned Numbers Authority (IANA) reserviert.
- **Sequence Number** Die 32 Bit lange Sequenznummer ist eine monoton steigende Nummer, die vom Absender jedem Paket zugewiesen wird. Der Empfänger kann diese Nummer nutzen, um sich vor Replay-Angriffen zu schützen.

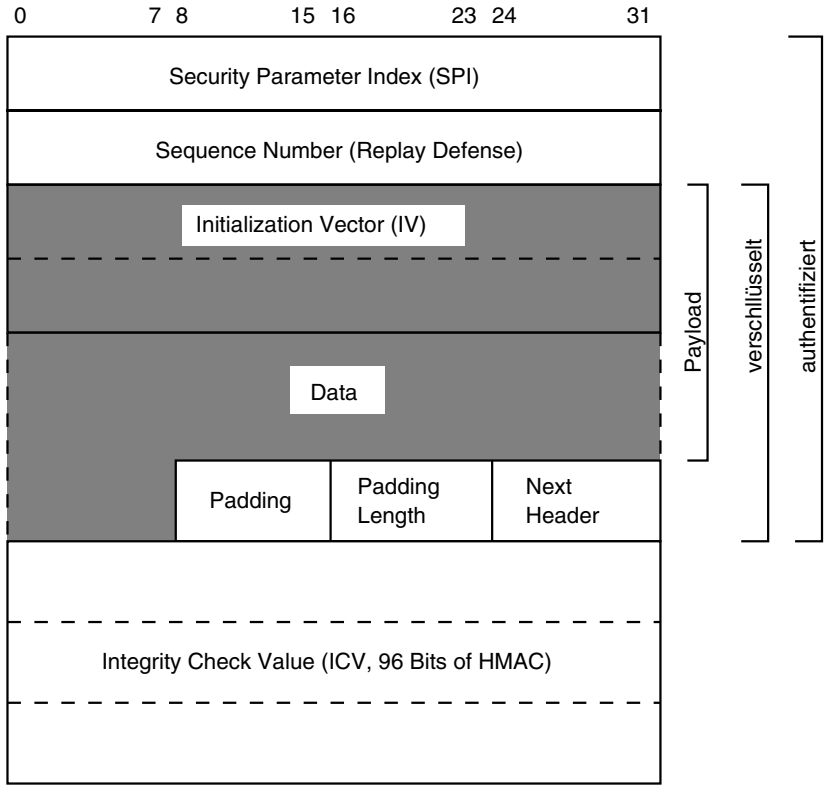


Abbildung 3.6 Das Encapsulated Security Payload Protokoll hat einen geteilten Header, der die zu schützenden Informationen umschließt

- **Payload** Hier werden die eigentlichen meist verschlüsselten Nutzinformationen abgespeichert. Wenn der Verschlüsselungsalgorithmus einen Initialization Vector benötigt, wird dieser zu Beginn abgespeichert.
  - **Initialization Vector (IV)** Sämtliche eingesetzten symmetrischen Verschlüsselungsverfahren sind monoalphabetisch und werden daher im Cipher-Block-Chaining Modus eingesetzt. Hierfür ist ein Initialisierungsvektor notwendig. Dieser wird zu Beginn abgespeichert. Die Länge richtet sich nach dem eingesetzten Verschlüsselungsverfahren. 3DES benötigt zum Beispiel einen 64 Bit langen IV.
  - **Data** Direkt im Anschluss an den IV werden die verschlüsselten Daten abgespeichert.
- **Padding** Da grundsätzlich bei IPsec Block-Ciphern eingesetzt werden, ist häufig ein Padding erforderlich. Hiermit werden die Daten bis zur nächsten Blockgrenze aufgefüllt. Dass Padding kann 0 bis 255 Bytes lang sein.

- **Padding Length** Um nach der Entschlüsselung das Pad entfernen zu können wird hier dessen Länge abgespeichert.
- **Next Header** Dieses Feld (8 Bit) gibt das Protokoll der im Paket übertragenen Informationen an. Handelt es sich um ein Paket im Transport Modus, so befindet sich hier die Protokollnummer des entsprechenden höheren Protokolls, zum Beispiel 6 (TCP) und 17 (UDP). Im Tunnel Modus ist hier bei Verwendung von IPv4 die Nummer 4 zu finden.
- **Integrity Check Value (ICV)** In diesem 96 Bit langen Feld werden die Authentifizierungsdaten gespeichert. Diese Daten können mit den verschiedenen HMAC Verfahren erzeugt werden. In allen Fällen werden nur die 96 höchstwertigen Bits des ermittelten Hashes hier abgespeichert.

Bei der Verarbeitung der Daten durch das ESP Protokoll, werden zunächst die Informationen verschlüsselt und anschließend erst die Authentifizierungsinformationen (ICV) berechnet. Dadurch ist sichergestellt, dass der Empfänger die Daten nicht aufwändig entschlüsseln muss, wenn die Überprüfung der Authentifzierung fehlschlägt.

Bei der Berechnung des ICV wird im Gegensatz zum AH-Protokoll nicht der äußere IP-Header mit einbezogen. Der ICV bezieht sich lediglich auf den ESP-Header und die zu sichernden Daten. Daher ist es grundsätzlich möglich die IP Adressen im äußeren IP-Header auszutauschen (Network Address Translation, NAT) ohne dass der ICV ungültig wird. In vielen Fällen ist ein NAT dennoch nur eingeschränkt möglich, da die meisten NAT-Geräte eine interne Zuordnung der »genatteten« Verbindungen über die in der Verbindung verwendeten Ports durchführen. Sowohl das AH als auch das ESP Protokoll benutzen jedoch keine Ports.

3.2.7 Security Association

Die Security Association (SA, dt. Sicherheitsassoziation, RFC 2401) definiert die von den IPsec Protokollen zu verwendenden Parameter: Ziel-IP Adresse, IPsec-Protokoll, Verschlüsselungsalgorithmus, Authentifizierungsalgorithmus und Schlüssel. Zusätzlich wird eine Lebensdauer, der Modus (Transport Modus/Tunnel Modus) und weitere Eigenschaften (zum Beispiel Anti Replay Service (ARS)) abgespeichert.

Die Security Associations sind immer unidirektional. Das bedeutet, dass in der Praxis immer für eine bidirektionale Kommunikation zwei unidirektionale IPsec SA existieren müssen. Eine bestimmt den ausgehenden Verkehr und eine weitere den ankommenden Verkehr.

Ein Zugriff auf die Security Associations ist eindeutig möglich mit dem Triplett: SPI, Ziel-IP Adresse und IPsec-Protokoll. Daher können alle Security Associations in einer Security Association Datenbank (SAD) gespeichert werden.

Eine Security Association enthält mindestens die folgenden Parameter:

- Ziel-IP Adresse
- IPsec Protokoll
- Security Parameter Index (SPI)
- aktuelle Sequenznummer
- Definition für den Fall eines Sequenznummerüberlaufs.
- Anti-Replay-Fenster
- AH/ESP Algorithmus mit Schlüsseln
- Lebensdauer der SA
- IPsec Modus (Tunnel Modus/Transport Modus/Beliebig[1])
- Path MTU

3.2.8 Security Policy

Die Security Association alleine führt noch nicht zu einer Verschlüsselung und/oder Authentitifizierung des Netzwerkverkehrs. Sie spezifiziert lediglich, wie der Verkehr geschützt werden soll, aber nicht was und wann. Dies ist die Aufgabe der Security Policy (SP, RFC 2401).

Die Security Policies werden in der Security Policy Datenbank (SPD) gespeichert. Für jedes ein- wie ausgehende Paket wird die SPD konsultiert, ob dieses Paket in irgendeiner Form modifiziert werden muss. Die SPD liefert für jedes Paket eine von drei möglichen Antworten: DISCARD (verwerfen), PASS (unverändert durchlassen) und APPLY (IPsec SAs anwenden).

TIPP

Die DISCARD Funktion führt dazu, dass unter Windows 2000 mit Hilfe der Security Policies auch Firewallfunktionen realisiert werden. Viele Microsoft Windows Administratoren bringen daher IPsec nur mit Firewallfunktionen in Verbindung.

1. Achtung: Der Linux Kernel 2.6 unterstützt bisher keine SAs, die sowohl im Tunnel- als auch im Transport Modus verwendet werden können.

Verlangt die Security Policy die Anwendung einer IPsec SA, so muss sie zusätzlich diese IPsec SA spezifizieren.

Um die Gestaltung der Security Policies möglichst flexibel zu ermöglichen existieren die IPsec Selektoren. Sie erlauben es in der IPsec Security Policy spezifisch den Dienst zu definieren, der mit IPsec geschützt werden soll. Hierzu können die IP Adressen und das höhere Transportprotokoll (zum Beispiel TCP oder UDP) angegeben werden. Wenn das Transportprotokoll Ports unterstützt, können sie zusätzlich angegeben werden.

Hiermit ist es möglich nur den Verkehr eines NTP-Zeitservers zu schützen. Das Network Time Protokoll verwendet den UDP Port 123.

ACHTUNG

Achtung! FreeS/WAN unterstützt nur mit den aktuellen Versionen des X.509 Patches Port- und Protokollselektoren. `racoon` und `isakmpd` unterstützen mit dem Linux Kernel 2.6 immer diese Selektoren.

3.2.9 Internet Key Exchange – IKE

Das Internet Key Exchange (IKE) Protokoll (RFC 2409) basiert auf dem Internet Security Association Key Management Protokoll (ISAKMP, RFC 2408), dem Oakley Key Determination Protokoll (RFC 2412), der IPsec Domain of Interpretation (DOI, RFC 2407) und dem SKEME Protokoll.

Hier soll aus Platzgründen lediglich das IKE Protokoll betrachtet werden. Weitere Ausführungen sind in den RFCs und in den Büchern [Lipp2001] und [Doroswamy1999] zu finden.

Das IKE Protokoll ermöglicht einen sicheren authentifizierten Schlüsselaustausch und die Aushandlung von IPsec Security Associations. Hiermit ermöglicht es eine automatische Erzeugung der IPsec SAs und einen automatischen Aufbau der VPN Verbindungen. Zusätzlich kann es die Sicherheit der VPN Verbindung dauerhaft gewährleisten, da es in der Lage ist bei Ablauf einer SA diese mit neuen Schlüsseln neu zu erzeugen.

Im Gegensatz zum AH und dem ESP Protokoll handelt es sich bei IKE nicht um ein eigenständiges IP Protokoll. Es setzt vielmehr auf dem UDP Protokoll auf und verwendet üblicherweise den Port 500.

Das IKE Protokoll arbeitet hierzu in zwei Phasen. In Phase 1 handeln die beiden Kommunikationspartner eine ISAKMP Security Association (SA) aus. Sie stellt einen sicheren authentifzierten Kanal dar, über den alle weiteren

Verhandlungen erfolgen. Hierzu definiert das IKE Protokoll zwei verschiedene Modi: Main Modus und Aggressive Modus. Diese Modi unterscheiden sich in ihrem Aufwand und in ihrer Sicherheit (s.u.).

In Phase 2 wird vom IKE Protokoll der Quick Modus verwendet. Dieser sehr schnelle Modus greift auf die ISAKMP SA der Phase 1 zurück, nutzt deren sicheren Kanal und braucht daher nicht erneut die Authentifizierung durchzuführen. Mit dem Quick Modus erzeugt das IKE Protokoll die IPsec SAs.

Der New Group Modus stellt eine Modus dar, der zwischen Phase 1 und Phase 2 verwendet werden kann, um eine neue Diffie Hellmann (Oakley) Gruppe zu vereinbaren.

Main Modus

In Phase 1 des IKE Protokolls wird eine ISAKMP SA ausgehandelt. Hierzu stehen zwei verschiedene Modi zur Verfügung: Main Modus und Aggressive Modus. Hier soll zunächst der Main Modus mit seinen Austauschvorgängen beschrieben werden. Sie unterscheiden sich jedoch stark je nach eingesetztem Authentifizierungsverfahren. Daher werden die Authentifizierungsverfahren einzeln betrachtet.

Authentifizierung mit RSA-Signaturen
Die Authentifizierung mit einer digitalen Signatur erfordert sechs Nachrichten zwischen dem Initiator und dem Empfänger (Responder). Die Abfolge der Nachrichten ist in Abbildung 3.7 dargestellt.

Der Initiator sendet zunächst in seinem ersten ISAKMP Paket ein oder mehrere Vorschläge (Proposal) für die ISAKMP SA. Diese Vorschläge enthalten die angebotenen und unterstützten Authentifizierungsalgorithmen, Verschlüsselungsalgorithmen und Oakley Gruppen. Der Responder wählt einen dieser Vorschläge aus und bestätigt diesen in der zweiten Nachricht. Lehnt der Responder sämtliche Vorschläge ab, so endet die Kommunikation bereits hier und die Verbindung kommt nicht zustande.

Anschließend sendet der Initiator in der dritten Nachricht das öffentliche Ergebnis seiner Diffie Hellmann Berechnung als `KEi` und einen zufälligen Nonce `Ni`, der vom Responder für die Berechnung der Signatur verwendet wird. Wenn der Initiator nicht den öffentlichen Schlüssel des Responders in Form eines Zertifikates besitzt, kann er es in dieser Nachricht anfordern.

Der Responder antwortet in der vierten Nachricht mit den analogen Informationen.

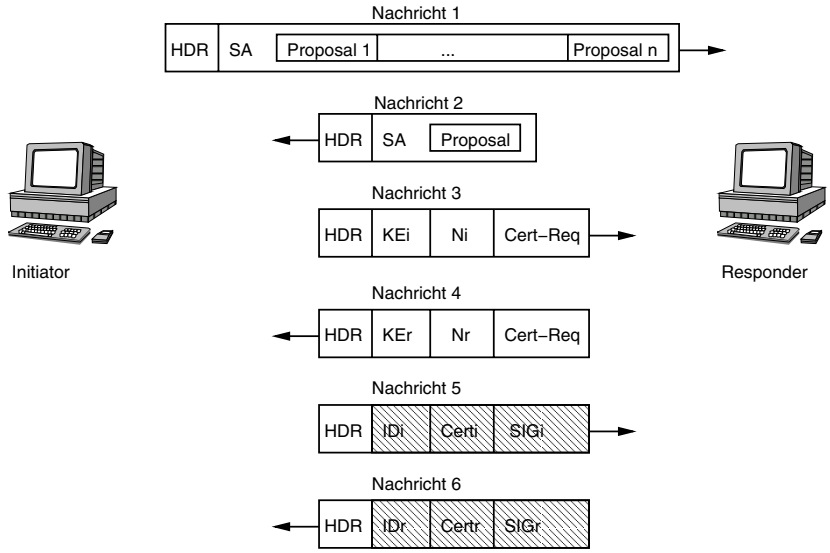


Abbildung 3.7 Die Authentifizierung mit digitalen Signaturen erfordert sechs Nachrichten im IKE Main Modus

Der Initiator kann nun mit dem Wert `KEr` die Diffie Hellmann Berechnung zu Ende führen und erhält einen symmetrischen Schlüssel. Er berechnet nun die digitale Signatur und sendet sie zusammen mit seiner Identität und bei Bedarf seinem Zertifikat an den Responder. Dabei werden diese Daten bereits mit dem symmetrischen Schlüssel verschlüsselt (schraffiert). Ein Dritter ist nicht in der Lage die Identität oder das Zertifikat unverschlüsselt zu lesen. Der Main Mode bietet hier einen Identitätsschutz.

Der Responder antwortet mit den analogen Informationen in der sechsten Nachricht.

Sowohl Initiator als auch Responder können die Inhalte der Nachrichten fünf und sechs entschlüsseln und die Signaturen mit den öffentlichen Schlüsseln überprüfen. Wurden diese in Form von Zertifikaten übermittelt, so müssen zuvor die Zertifikate auf ihre Korrektheit überprüft werden.

Dieses Verfahren wird meist eingesetzt, wenn X.509 Zertifikate genutzt werden.

Authentifizierung mit Public Key Verschlüsselung

Bei der Authentifizierung mit Public Key Verschlüsselung werden RSA Schlüssel eingesetzt. Hierzu ist es erforderlich, dass die öffentlichen RSA Schlüssel des jeweiligen Partners im Vorfeld ausgetauscht wurden. Ein Austausch während Phase 1 ist nicht vorgesehen.

Nachricht 1 und 2 unterscheiden sich nicht von der Authentifizierung mit digitalen Signaturen.

In der Nachricht drei überträgt der Initiator wie gehabt auch das öffentliche Ergebnis seiner Diffie Hellmann Berechnung. Zusätzlich überträgt er seinen Nonce und seine Identität (Abbildung 3.8). Diese verschlüsselt er einzeln mit dem öffentlichen Schlüssel des Empfängers (schraffiert). Verfügt er über mehrere öffentliche Schlüssel des Empfängers, so wählt er einen aus und überträgt zusätzlich den Hash-Wert dieses Schlüssels (`H(Cert)`).

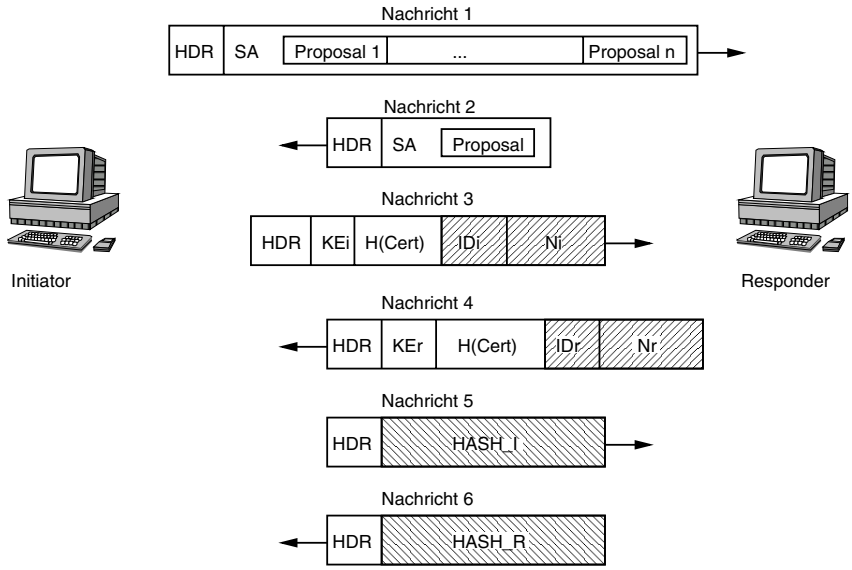


Abbildung 3.8 Die Authentifizierung mit Public Key Verschlüsselung erfordert vier Public Key Operationen

Der Responder antwortet analog in Nachricht vier. Insgesamt werden also vier Informationen `IDi`, `Ni`, `IDr` und `Nr` mit RSA verschlüsselt.

Nun können Initiator und Responder mit ihren privaten RSA Schlüsseln die Identität des Partners und dessen Nonce entschlüsseln. Zusätzlich können die Diffie Hellmann Berechnungen mit den Werten `KE` zu Ende geführt und der symmetrische Schlüssel bestimmt werden.

Initiator und Responder berechnen nun aus dem Nonce und weiteren Informationen einen Hash, der anschließend mit dem symmetrischen Schlüssel übertragen wird. Kann die jeweilige Gegenstelle den Hash authentifizieren, so besaß der Erzeuger des Hash den Nonce. Um den Nonce zu besitzen musste dieser jedoch zuvor mit dem privaten Schlüssel entschlüsselt werden.

Somit ist die Gegenseite authentifiziert. Dieses Verfahren wird nur selten eingesetzt.

Authentifizierung mit revidierter Public Key Verschlüsselung

Die Authentifizierung mit Public Key Verschlüsselung bietet Vorteile, da ein Angreifer sowohl die symmetrische als auch die asymmetrische RSA Verschlüsselung knacken muss. Sie besitzt aber auch den Nachteil, dass vier Public Key Operationen pro System durchzuführen sind: zwei Verschlüsselungen und zwei Entschlüsselungen. Das revidierte Verfahren reduziert dies auf zwei Public Key Operationen.

Das revidierte Verfahren (Abbildung 3.8) ähnelt dem Public Key Verschlüsselungsverfahren. Die erste und zweite Nachricht sind wieder identisch. In Nachricht drei überträgt der Initiator den Nonce verschlüsselt mit dem Public Key des Responders. Erneut muss er den Hash des verwendeten Public Keys übertragen, wenn mehrere Public Keys zur Verfügung stehen.

Das Ergebnis der Diffie Hellmann Berechnung, seine eigene Identität und ein möglicher eigener Public Key werden jedoch nicht mit dem Public Key des Empfängers verschlüsselt, sondern mit einem symmetrischen Schlüssel. Da das Diffie Hellmann Verfahren aber noch nicht abgeschlossen wurde, ermittelt der Initiator diesen Schlüssel aus seinem eigenen Nonce. Der Responder kann den Nonce mit seinem privaten Schlüssel dekodieren und den symmetrischen Schlüssel nach dem identischen Verfahren ableiten. Mit diesem kann er dann die Daten `IDi` und `KEi` entschlüsseln.

Der Responder antwortet wieder mit analogen Daten.

Die beiden Nachrichten fünf und sechs enthalten wieder Hashes, die neben anderen Informationen auch auf den ausgetauschten Nonces beruhen. Sie dienen der Authentifizierung.

Authentifizierung mit Preshared Keys (PSK)

Die Authentifizierung mit einem Preshared Key ist ebenso in der Phase 1 des IKE Main Modus möglich. Hierzu werden die beiden ersten Nachrichten wie bei den anderen Methoden ausgetauscht (Abbildung 3.9).

Anschließend tauschen Initiator und Responder den Nonce und das öffentliche Ergebnis der Diffie Hellmann Berechnung aus (Nachrichten drei und vier). Dabei wird der PSK bereits für die Erzeugung der Ausgangswerte für die Schlüsselberechnung verwendet.

Nach dem Austausch der Werte `KE` können Initiator und Responder die Nachrichten fünf und sechs übermitteln. Hier werden unter anderem die Identitäten verschlüsselt ausgetauscht.

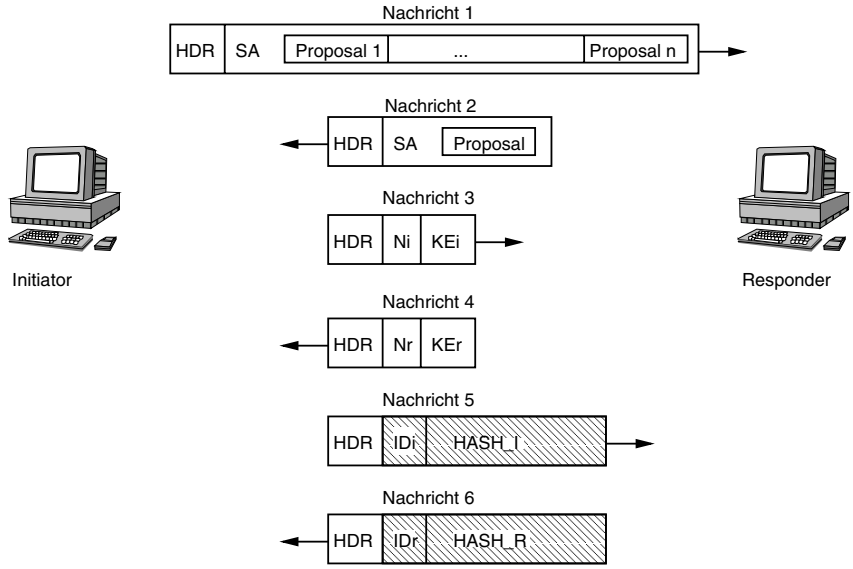


Abbildung 3.9 Bei der Authentifizierung mit Preshared Keys wird die Identität erst in den letzten Nachrichten verschlüsselt übertragen

Dies ist der wesentliche Grund, warum bei der Verwendung von PSKs im Main Mode bei unbekannten IP Adressen alle Kommunikationspartner denselben PSK verwenden müssen. Werden unterschiedliche PSKs benutzt, so muss der Kommunikationspartner bereits vorm Senden der Nachrichten drei und vier den richtigen PSK wählen. Da die Identität noch nicht übertragen wurde (Identitätsschutz des Main Modus) kann die Auswahl der PSKs nur auf der IP Adresse beruhen. Ist auch die unbekannt, kann nur ein allgemeiner PSK verwendet werden.

Aggressive Modus

Der Aggressive Modus stellt eine schnellere Variante zur Verhandlung der Phase 1 dar. Hierbei werden sämtliche Informationen in nur drei Nachrichten ausgetauscht. Dadurch sind im Gegensatz zum Main Modus Denial-of-Service Angriffe möglich, da bereits das erste Paket eine Verschlüsselung verlangt. Ein Angreifer kann so eine Vielzahl von Paketen erzeugen und durch unzählige sinnlose Verschlüsselungsvorgänge einen DoS erzeugen.

Der zweite Unterschied beim Aggressive Modus betrifft die Authentifizierung mit Preshared Keys und digitalen Signaturen. Werden PSKs oder digitale Signaturen im Aggressive Modus eingesetzt, so wird die Identität im Klartext übertragen. Der Aggressive Modus verlangt keinen Identitätsschutz wie der Main Modus. Daher ist es möglich im Aggressive Modus unter-

schiedliche PSKs auch bei dynamischen unbekannten IP Adressen einzusetzen. Bei der Authentifizierung mit Public Key Verschlüsselung werden weiterhin die Identitäten geschützt.

Der fehlende Identitätsschutz des Aggressive Modus stellt eine Sicherheitslücke dar. Es existieren einige Werkzeuge (IKEcrack, *http://sourceforge.net/projects/ikecrack*) und Artikel (*www.ernw.de/download/pskattack.pdf*), die diese Sicherheitslücke beschreiben und ausnutzen können.

Eine Verwendung des Aggressive Modus sollte aus diesen Gründen nur in Betracht gezogen werden, wenn der Main Modus von einem Kommunikationspartner nicht unterstützt wird.

Quick Modus

Der Quick Modus wird in der Phase 2 vom IKE Protokoll verwendet. Er setzt immer eine erfolgreich durchlaufene Phase 1 vorraus. Hiermit können die IPsec Security Associations auf der Basis der ISAKMP SA der Phase 1 erzeugt werden. Dabei können unter dem Schutz einer ISAKMP SA mehrere IPsec SAs sogar innerhalb eines Quick Mode erzeugt werden. Hier wird nun der Vorteil der Aufteilung des IKE Protokolls in Phase 1 und Phase 2 deutlich. Werden mehrere IPsec SAs benötigt, so müssen die aufwändigen Authentifizierungsnachrichten nur einmalig beim Aufbau der ISAKMP SA ausgetauscht werden. Die ISAKMP SA stellt damit auch schon Dienste zur Überprüfung der Integrität, Authentifizierung und zur Sicherstellung der Vertraulichkeit zur Verfügung.

Sämtliche Pakete der Phase 2 werden verschlüsselt ausgetauscht. Für den Aufbau einer IPsec SA werden drei Nachrichten benötigt (Abbildung 3.10).

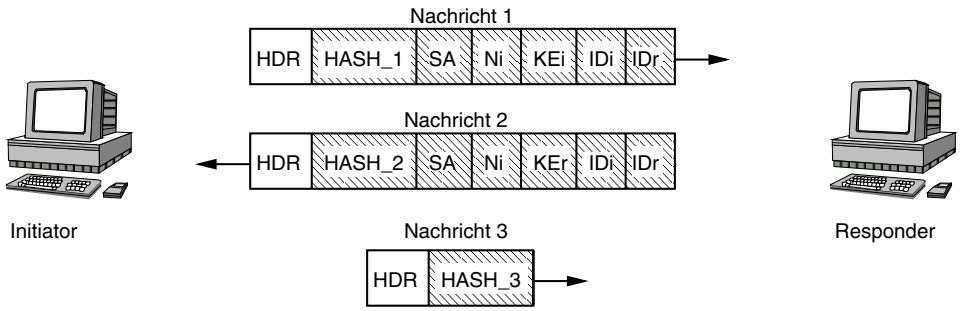


Abbildung 3.10 Der Quick Mode überträgt alle Informationen verschlüsselt

Hierbei müssen der SA Vorschlag, der Hash-Wert und der Nonce-Wert übertragen werden. Optional sind die Identitäten und der Diffie Hellman Wert. Der Diffie Hellman Wert wird benötigt, wenn Perfect Forward Secrecy gewünscht wird. Dann werden sämtliche benötigten Schlüssel neu berechnet und nicht von den Masterschlüsseln der ISAKMP SA abgeleitet. Damit wird sichergestellt, dass ein Angreifer durch die Berechnung eines Schlüssels nicht weitere Schlüssel erschließen kann.

3.2.10 UDP Encapsulation

Einer der wichtigsten Beweggründe für die Entwicklung des IPv6 Protokolls war die prognostizierte Knappheit der IPv4 Adressen zu Beginn der 90er Jahre. Das IPv6 Protokoll bietet mit einem 128 Bit großen Adressraum ausreichend IP Adressen »für jeden Toaster und jede Ampel auf der Erde«.

Dies führte jedoch auch dazu, dass bei der Entwicklung der IPsec Protokolle nicht berücksichtigt wurde, dass Network Address Translation (NAT) erforderlich sein könnte. Network Address Translation wurde entwickelt, um der Knappheit der IPv4 Adressen entgegenzutreten. So benötigt ein großes Unternehmen mit mehreren Tausenden Rechnern nur wenige offizielle IPv4 Adressen um jedem Rechner einen Zugriff aufs Internet zu ermöglichen und erreichbar zu sein. Intern werden private IPv4 Adressen verwendet, die beim Zugriff auf das Internet meist von einer Firewall in offizielle IPv4 Adressen umgesetzt werden.

Das Authentication Header (AH) Protokoll verhindert diese Umsetzung wirkungsvoll, da es auch die IP Adressen des äußeren IP-Headers authentifiziert. Werden sie verändert, so wird das Paket ungültig. Das Encapsulated Security Payload weist diese Einschränkung nicht auf. Dennoch führt sein Einsatz in NAT Umgebungen häufig zu Problemen.

Um dies zu verstehen soll kurz vorgestellt werden, wie in den meisten Fällen das NAT durchgeführt wird. Hierbei wird nur auf das Source NAT, das Austauschen der Absender IP Adresse eingegangen. Jedoch arbeitet das Destination NAT, das die Ziel IP Adresse modifiziert, analog.

Die Abbildung 3.11 zeigt die Funktion des NAT Gateways.

Ein NAT Gerät erzeugt eine Tabelle, in der sämtliche Verbindungen gepflegt werden. Um einen eindeutigen Index auf diese Tabelle zu erhalten, werden die Verbindungen nach der Client IP Adresse und dem Client Port (bei TCP und UDP) sortiert. Um ankommende Antwort Pakete eines Servers eindeutig zuordnen zu können, wird jedem Paar aus Client IP Adresse und Port ein Paar aus IP Adresse des NAT Gateways und Port zugewiesen. Dies ist jedoch

nur möglich wenn das Protokoll Ports unterstützt. Bei einigen Protokollen (zum Beispiel ICMP) können auch andere Parameter des Protokolls genutzt werden.

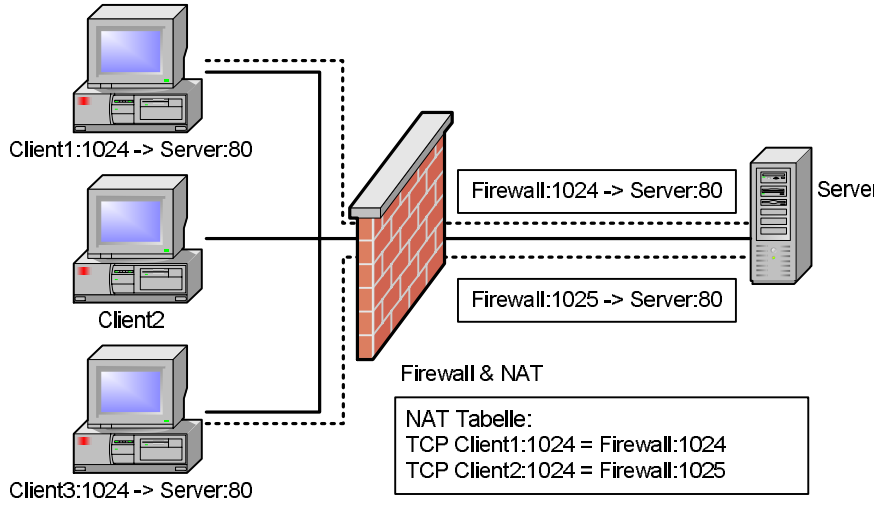


Abbildung 3.11 Ein NAT Gateway ordnet die Pakete den genatteten Verbindungen über die Client IP Adresse und den Client-Port zu

Das Protokoll ESP unterstützt nun jedoch keine Ports. Die einzige Information, die dem Port nahe kommt und in Klartext lesbar ist, ist der SPI. Jedoch unterstützen die wenigsten NAT Gateways den SPI in ihren NAT Tabellen.

Aus diesem Grund gibt es Probleme, sobald mehr als ein Rechner über ein NAT Gateway eine IPsec Verbindung aufbauen möchte. Das NAT Gateway hat Schwierigkeiten, die Antwortpakete den Clients wieder zuordnen zu können, da keine Unterscheidung der Verbindungen auf der Basis des Ports vorgenommen werden kann.

Hier hilft die erneute Kapselung aller IPsec Pakete in UDP Paketen. UDP Pakete weisen einen Port auf, und können daher durch eine Standard NAT Tabelle unterschieden werden.

Zwei Internet Engineering Task Force (IETF) Drafts beschreiben das sogenannte NAT Traversal: »Negotiation of NAT Traversal in the IKE« und »UDP Encapsulation of IPsec Packets«.

Das erste Draft beschreibt eine Erweiterung des IKE Protokolls. Damit ist es möglich, automatisch die Unterstützung von NAT Traversal durch den Kommunikationspartner und das Vorhandensein eines NAT Gerätes zwischen den Kommunikationspartnern zu erkennen.

Diese Erkennung erfolgt in Phase 1 des IKE Protokolles. Die NAT Traversal Unterstützung wird durch spezielle Vendor ID Nachrichten erkannt.

Die Lokalisierung eines möglichen NAT Gerätes erfolgt durch spezielle NAT Discovery (NAT-D) Pakete in Phase 1 (Abbildung 3.12).

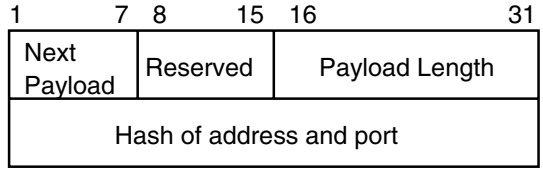


Abbildung 3.12 Das NAT Discovery Paket

Diese NAT D Pakete enthalten jeweils den Hash der Quell- und der Ziel IP Adresse und des Ports. Berechnet der Empfänger einen anderen Hash aufgrund der IP Adresse und des Ports im IP Header, so befindet sich ein NAT Gerät zwischen Absender und Empfänger. Diese Pakete sind beim Main Mode in den Nachrichten drei und vier und im Aggressive Mode in den Nachrichten zwei und drei enthalten. Verfügt ein Absender über mehrere IP Adressen über die das Paket den Rechner verlassen darf, so kann er mehrere dieser Pakete in seinen Nachrichten übertragen.

Da es einige NAT Geräte gibt, die versuchen mit IPsec intelligent umzugehen und dies beim NAT Traversal störend sein kann, sollte möglichst bald der IKE-Port 500/udp gewechselt werden. Diese intelligenten NAT Geräte erkennen den IPsec Verkehr meist nur am IKE-Port 500/udp. IKE Daemonen, die NAT Traversal unterstützen, verwenden daher, sobald sie NAT erkannt haben, den Quell- und den Ziel-Port 4500/udp.

Alle weiteren Rekey-Vorgänge der ISAKMP SA und der IPsec SAs werden mit diesem Port durchgeführt.

Hiermit ist Phase 1 abgeschlossen. Nun können in Phase 2 mit dem Quick Modus die IPsec SAs verhandelt werden. Hierbei stehen zwei neue Kapselmodi zur Verfügung:

- UDP-Encapsulated-Tunnel
- UDP-Encapsulated-Transport

Diese beiden Modi werden im Draft »UDP Encapsulation of IPsec Packets« beschrieben. Die wesentlichen Grundzüge werden im Folgenden dargestellt.

Die IPsec ESP Pakete werden in UDP Pakete eingepackt. Hierbei wird derselbe Port verwendet, wie er auch bereits für die IKE Verhandlungen verwendet wurde: 4500/udp. Der UDP ESP Header ist in Abbildung 3.13 dargestellt.

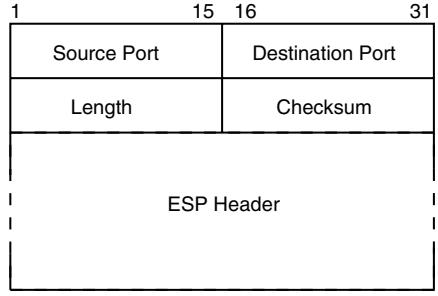


Abbildung 3.13 Das ESP Paket wird in einem UDP Header eingepackt

Damit die NAT Verbindung über NAT Geräte am Leben gehalten werden kann, auch wenn kein Verkehr zu transportieren ist, werden NAT Keepalive Pakete gesendet. Ansonsten verwerfen einige NAT Geräte bereits nach kurzer Zeit, nämlich 180 Sekunden, die Verbindung aus der NAT Tabelle. Der NAT Keepalive Header ist in Abbildung 3.14 zu sehen.

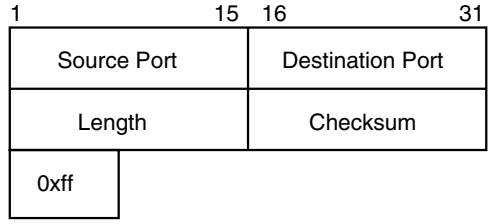


Abbildung 3.14 Der UDP NAT Keepalive Header hält die Verbindung aufrecht, wenn keine Daten transportiert werden müssen

Diese Keepalive Pakete werden normalerweise alle 20 Sekunden versandt, wenn kein anderes Paket über den Tunnel transportiert wurde.

3.2.11 DHCP-over-IPsec

In vielen Fällen, bei denen ein entfernter Rechner über eine VPN Verbindung mit einem LAN in Kontakt tritt, ist es von Vorteil, wenn anschließend der entfernte Rechner eine IP Adresse aus dem internen Netzwerk erhält und

sich scheinbar im LAN befindet. Dies kann erreicht werden, indem der Client eine virtuelle IP Adresse per DHCP erhält.

Es existieren eine Vielzahl von Drafts und ein RFC (RFC 3456), die den Einsatz von DHCP über IPsec beschreiben. Hier soll kurz die Technologie beschrieben werden, wie sie vom RFC 3456 als Standard vorgeschlagen wird. Dies weicht in einzelnen Punkten von den frühen Drafts und Implementierungen ab.

Die Verwendung von DHCP ist nur für IPv4 Adressen im IPsec Tunnel Modus beschrieben und sinnvoll. Hierbei kann mit DHCP die IP Adresse an den Client zugewiesen werden. Die Verwendung von DHCP bietet folgende Vorteile:

- Die Verwendung von DHCP erleichtert den Einsatz und die Integration in großen Netze, da diese meist sowieso bereits mit Hilfe des DHCP Protokolls verwaltet werden.
- Zusätzlich bietet das DHCP Protokoll die Möglichkeit den Adressen 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.
- Das DHCP Protokoll verfügt bereits über Verfahren zur Neukonfiguration der IP Adressen. Diese Funktion muss daher nicht im 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.

Die Abbildung 3.15 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 vom 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.

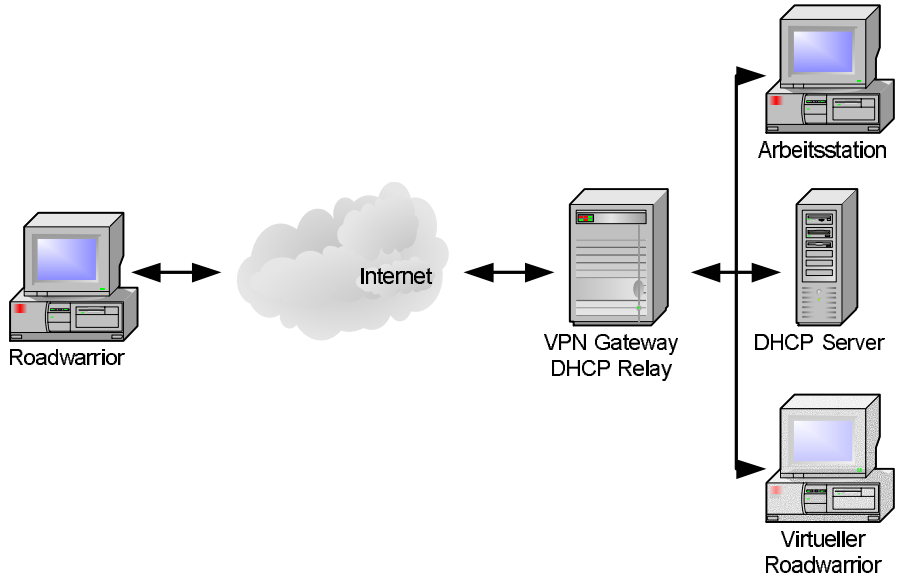


Abbildung 3.15 Der Client fordert zunächst per DHCP eine IP Adresse an, um später mit dieser Adresse über den Tunnel zu kommunizieren

Die DHCP-over-IPsec Kommunikation funktioniert folgendermaßen:

1. Der Client baut eine ISAKMP SA in der Phase 1 des IKE Protokolls auf.
2. Der Client baut eine DHCP SA mit dem IPsec Tunnel Mode VPN Gateway auf. Hier werden Protokoll- und Portselektoren verwendet, um sicherzustellen, dass lediglich DHCP Verkehr über diese SA ausgetauscht werden darf.
3. Die DHCP Nachrichten werden zwischen dem Client und dem DHCP Relay auf dem VPN Gateway ausgetauscht. Das DHCP Relay fordert die virtuelle IP Adresse für den Client vom DHCP Server an und gibt sie an den Client weiter.
4. Nun kann die Original DHCP SA gelöscht und eine neue SA mit der neuen IP Adresse aufgebaut werden (üblich) oder die vorhandene durch Erweiterung der Protokoll- und Portselektoren für den allgemeinen Verkehr geöffnet werden.

Obwohl sicher andere Varianten existieren, wird normalerweise das VPN Gateway nicht auch der DHCP Server sein. Daher ist auf dem VPN Gateway ein DHCP Relay erforderlich.

TIPP

FreeS/WAN mit dem X.509 Patch kann als VPN Gateway diese Funktionen wahrnehmen. Der Linux Kernel 2.6 unterstützt mit `isakmpd` den IKE Config Modus.

3.3 L2TP

Das Layer-Two-Tunneling-Protokoll (L2TP) ist kein VPN Protokoll per se. Es bietet lediglich Tunnelfunktionen. Hierzu tunnelt es beliebige Pakete in einem UDP Tunnel. Es verwendet den Port 1701. Das L2TP ist nicht in der Lage die Integrität, Authentizität oder Vertraulichkeit der übertragenen Daten zu garantieren. Dennoch soll das Protokoll hier betrachtet werden, da verschiedene Hersteller, zum Beispiel Microsoft, es in Kombination mit dem IPsec Protokoll einsetzen um VPN-Funktionen anzubieten.

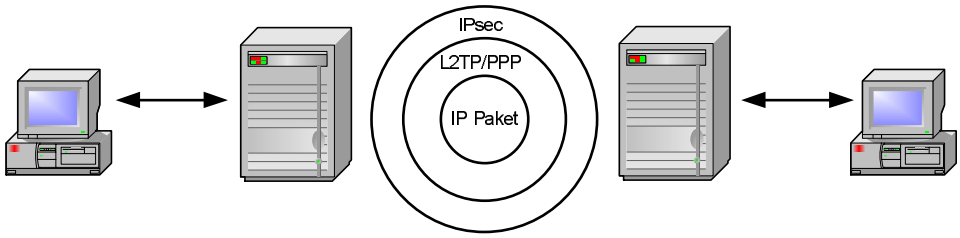


Abbildung 3.16 IPsec schützt den L2TP Tunnel

Hierbei wird zunächst mit dem IPsec Protokoll eine verschlüsselte Verbindung zwischen den Kommunikationspartnern aufgebaut. Diese Verbindung bietet nun garantiert die Authentizität, Integrität und Vertraulichkeit der übertragenen Informationen. Auf der Basis dieser Verbindung wird anschließend mit dem L2TP Protokoll ein weiterer L2TP Tunnel aufgebaut (Abbildung 3.16). Dieser Tunnel bietet alle Vorteile des L2TP Protokolls.

Im einzelnen sind das die folgenden Punkte:

- Der Aufbau des L2TP Tunnels erfordert eine erneute Benutzerauthentifizierung. Sie kann von der IPsec Authentifizierung verschieden sein.
- Der L2TP Tunnel kann einen anderen Endpunkt haben, als die IPsec Verbindung.
- Der L2TP Tunnel kann auch nicht IP-Pakete transportieren, zum Beispiel IPX. Es ist ein Tunnel auf Layer 2.

- Der L2TP Tunnel kann andere IP Adressen verwenden, als die IPsec Verbindung. Das L2TP Protokoll bietet sogar Funktionen, diese IP Adressen automatisch einem Client zuzuweisen (ähnlich DHCP over IPsec).

Damit ist das L2TP Protokoll in Kombination mit dem IPsec Protokoll ideal dazu geeignet, Rechnern mit dynamischer IP Adresse (Roadwarrior) von außen einen gesicherten Zugriff auf ein Unternehmensnetzwerk zu ermöglichen.

Die IPsec Protokolle wurden bereits ausführlich betrachtet. Für die Sicherung des L2TP Protokolls muss verpflichtend das ESP Transport Protokoll implementiert werden. Die Implementierung des ESP Tunnel Protokolls ist optional. Eine automatische Schlüsselverwaltung ist erforderlich. Das IKE Protokoll wird für diesen Zweck empfohlen. Diese Anforderungen werden von allen dem Autor bekannten Implementierungen eingehalten.

Jetzt soll das L2TP Protokoll kurz erläutert werden. Ausführlichere Informationen finden sich in den RFCs RFC 2661 (Layer Two Tunneling Protocol) und RFC 3193 (Securing L2TP using IPsec).

3.3.1 Einführung

Das L2TP Protokoll ist eine Erweiterung des Point-to-Point Protokolls (PPP). Das PPP Protokoll definiert die Schachtelung beliebiger Pakete über eine Layer 2 (L2) Point-to-Point Verbindung. Hierzu fordert der Benutzer zunächst eine L2 Verbindung an und benutzt anschließend PPP über diese Verbindung.

L2TP erlaubt dabei im Gegensatz zu PPP, dass die L2 Verbindung und die PPP Verbindung auf unterschiedlichen physikalischen Geräten enden.

Für die Beschreibung des L2TP Protokolls ist die Bedeutung einiger Begriffe erforderlich, von denen die wichtigsten im Folgenden beschrieben werden sollen:

- **CHAP** Challenge Handshake Authentication Protokoll (RFC 1994). Ein Authentifizierungsverfahren, bei dem das Kennwort nicht im Klartext übertragen wird.
- **Attribute Value Pair (AVP)** Ein Paar aus einem Attribut und einem Wert. Steuerungsmitteilungen bestehen aus mehreren AVPs.
- **L2TP Access Concentrator (LAC)** Ein Endpunkt eines L2TP Tunnels.
- **L2TP Network Server (LNS)** Ein Endpunkt eines L2TP Tunnels und der Kommunikationspartner des LAC.

- **Incoming Call** Ein Anruf, der von einem LAC zur Weitergabe an ein LNS empfangen wurde.
- **Outgoing Call** Ein Anruf, der von einem LAC für ein LNS getätigt wird.

3.3.2 Protokoll

Das L2TP Protokoll verwendet zwei unterschiedliche Nachrichtenformen: Steuerungsnachrichten und Datennachrichten. Steuerungsnachrichten werden verwendet, um Tunnel und Anrufe aufzubauen, zu verwalten und zu löschen. Datennachrichten werden verwendet um die Daten PPP gekapselt über den Tunnel zu transportieren. Im Gegensatz zum Steuerungskanal garantiert der Datenkanal nicht die erfolgreiche Übertragung der Daten.

Der L2TP-Header ist für die Steuerungs- und Datennachrichten identisch und in Abbildung 3.17 dargestellt.

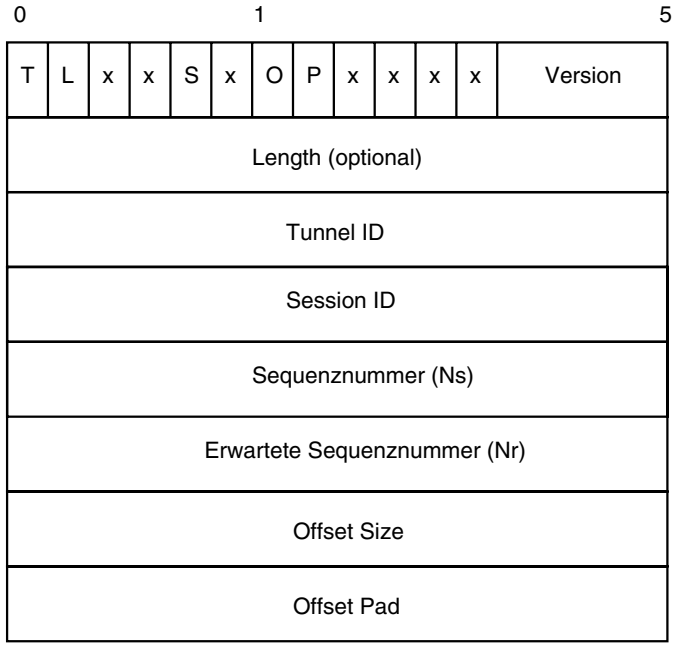


Abbildung 3.17 Der L2TP Header ist identisch für Steuerungs- und Datennachrichten

Die Felder im Header haben die folgenden Aufgaben:

- **T** Typ. 0 Datennachricht, 1 Steuerungsnachricht
- **L** Wenn dieses Feld auf 1 gesetzt ist, ist das Feld *Length* gesetzt.

- **x** Reserviert für zukünftige Verwendung
- **S** Wenn dieses Feld gesetzt ist, enthalten die Pakete eine Sequenznummer
- **O** Wenn dieses Feld gesetzt ist, enthalten die Pakete ein Offsetfeld
- **P** Wenn dieses Feld gesetzt ist, sind die Pakete mit erhöhter Priorität zu behandeln.
- **Ver** L2TP Version. Dieses Feld muss den Wert 2 tragen für L2TP.
- **Length** Dieses optionale Feld enthält die Gesamtlänge in Bytes.
- **Tunnel ID** Dies kennzeichnet den einzelnen Tunnel. Diese Angabe ist nur lokal eindeutig. Die Endpunkte können unterschiedliche Tunnel IDs verwenden. Die Tunnel ID in einer Nachricht, ist immer die Tunnel ID des Empfängers.
- **Session ID** Dies kennzeichnet die einzelne Sitzung in einem Tunnel. Die in einer Nachricht gesetzte Session ID ist immer die ID des Empfängers.
- **Ns, Nr** Sequenznummern für gesendete Pakete (s) und empfangene Pakete (received, r).
- **Offset** Dieses Feld definiert, wo im L2TP Paket die eigentlichen Daten beginnen.

Diese Ausführungen sollen hier genügen, um das L2TP Protokoll einsetzen und analysieren zu können.

4 Keymanagement

Keymanagement stellt die sichere Erzeugung, Verteilung, Speicherung und Zerstörung der Schlüssel sicher. Diese Schlüsselverwaltung ist von höchster Wichtigkeit für die dauerhafte Sicherheit kryptografischer Anwendungen. Sobald ein Schlüssel erzeugt wurde, muss verhindert werden, dass dieser Schlüssel in die Hände von Dritten gerät.

Bei den modernen Public-Key-Verfahren sind Angriffe auf das Keymanagement wahrscheinlich erfolgreicher als Angriffe gegen den kryptografischen Algorithmus!

Ziel des Keymanagement muss es sein, einem Benutzer zu erlauben, sein Schlüsselpaar zu erzeugen, die öffentlichen Schlüssel anderer Benutzer zu suchen und seinen eigenen öffentlichen Schlüssel zu veröffentlichen. Dabei ist es erforderlich, dass dieser Austausch der öffentlichen Schlüssel authentifiziert erfolgt, so dass ein Dritter nicht als Man-in-the-Middle die Identität eines Benutzers annehmen kann.

Die privaten Schlüssel müssen in geeigneter Weise geschützt gespeichert werden können, um eine Kompromittierung zu verhindern. Wenn dennoch ein privater Schlüssel bekannt wird, muss das Keymanagement die Möglichkeit bieten, diesen Schlüssel für ungültig zu erklären. Diese Information muss allen Beteiligten zur Verfügung stehen.

4.1 Einleitung

Das Keymanagement ist eine zentrale Aufgabe beim Aufbau einer VPN-Lösung, die leider viel zu häufig vernachlässigt wird. Die Schlüssel werden auf beliebigen unsicheren Medien wie Disketten oder Festplatten ungesichert gespeichert. Teilweise werden für die Aufgaben der Authentifizierung einfache Kennworte ungenügender Länge und Komplexität eingesetzt. Ein Angriff, der bei sinnvoller (zufälliger) Wahl des Kennwortes, so lange dauern würde, dass er nicht durchführbar wäre, kann in wenigen Sekunden durch den Einsatz eines Wörterbuches zum Erfolg führen.

4.1.1 Zufallszahlen

Bei der Erzeugung der Schlüssel ist es besonders wichtig, dass diese Schlüssel zufällig erzeugt werden. Dabei ist es unerheblich, ob der Schlüssel später

als symmetrischer oder asymmetrischer Schlüssel eingesetzt werden soll. Wenn der Schlüssel nicht zufällig gewählt wird, sondern nur aus lesbaren Zeichen besteht, kann der Angreifer diese Information für einen Angriff auf den Schlüssel ausnutzen.

Bei einem zufälligen Schlüssel kann ein Angreifer keine Rückschlüsse auf den Schlüssel ziehen und muss in einem Brute Force Angriff jeden möglichen Schlüssel ausprobieren. Kann er bestimmte Eigenschaften des Schlüssels vorhersagen, zum Beispiel, dass er nur aus lesbaren Buchstaben besteht, so verringert sich die Menge der Möglichkeiten für den Schlüssel rapide. Der Angreifer kann dies ausnutzen und seinen Angriff dementsprechend anpassen.

Was ist nun eine gute Quelle für Zufallszahlen? Die wichtigste Eigenschaft ist die Unvorhersagbarkeit durch den Angreifer. Da jedoch Computer meist sehr vorhersagbar sind, wenigstens unter Linux, werden für die Generierung der Zufallszahl meist externe Quellen genutzt. Hierbei handelt es sich zum Beispiel um die Mausbewegungen eines Benutzers und den zeitlichen Abstand zwischen zwei dem zweimaligen Drücken einer Taste. Häufig stehen derartige Informationen aber nicht in ausreichender Menge zur Verfügung, so dass die Quelle versiegen kann. Daher werden oft auch Pseudo-Zufallszahlengeneratoren eingesetzt, die auf Grund ihrer Funktion immer periodisch arbeiten. Um diese Generatoren aber auch zur Erzeugung von Zufallszahlen einsetzen zu können, werden sie mit einer Zufallszahl (random seed) geimpft. Dadurch kann der Start nicht vorhergesagt werden und pseudozufällige Zahlen sind das Ergebnis.

Linux verfügt sowohl über einen echten Zufallszahlengenerator `/dev/random` und einen Pseudozufallszahlengenerator `/dev/urandom`. Für die Erzeugung von Schlüsseln sollte darauf geachtet werden, dass die verwendete Software als Zufallszahlengenerator `/dev/random` nutzt.

4.1.2 Lebensdauer von Schlüsseln

Schlüssel sollten grundsätzlich eine beschränkte Lebensdauer besitzen. Anschließend sollten die Schlüssel vernichtet werden. Eine Archivierung darf nur unter besonderen Umständen und in einer sicheren Umgebung erfolgen.

Die Kryptoanalyse benötigt möglichst viele durch einen Schlüssel verschlüsselte Ciphertexte um den Schlüssel berechnen zu können. Je länger ein Schlüssel eingesetzt wird, desto mehr Ciphertexte kann der Angreifer sammeln und um so wahrscheinlicher den Schlüssel ermitteln.

Es muss zusätzlich möglich sein, während der Lebensdauer einen Schlüssel für ungültig zu erklären, um die Verwendung eines kompromittierten Schlüssels zu unterbinden.

Ford beschreibt den Lebenszyklus eines öffentlichen Schlüssels in *Standard Protocols and Techniques* folgendermaßen:

1. Schlüsselerzeugung (und Registratur)
2. Schlüsselverteilung
3. Schlüsselverwendung (Aktivierung/Deaktivierung)
4. Schlüsselaustausch/-update
5. Rückruf des Schlüssels
6. Vernichtung/Archivierung des Schlüssels

4.1.3 Kennwörter und symmetrische Schlüssel

Kennwörter und symmetrische Schlüssel stellen die problematischsten Schlüssel im Keymanagement dar. Diese Schlüssel müssen allen beteiligten Parteien bekannt sein, damit sie für Authentifzierungs- und Verschlüsselungszwecke eingesetzt werden können.

Häufig steht bei symmetrischen Schlüsseln kein (eingebautes) Keymanagement zur Verfügung, das in der Lage ist die Lebensdauer dieser Schlüssel zu überwachen und einen Austausch der Schlüssel zu erzwingen. Werden die Schlüssel nicht von Maschinen, sondern von lebenden Personen benutzt, so versuchen diese nicht selten vorhandene Systeme, die die Lebensdauer beschränken, zu unterlaufen. In diesem Fall werden die Personen auch nicht zufällige Schlüssel, wie sie Maschinen benutzen können, wählen, da sie nur sehr schlecht einzuprägen sind. Werden Personen gezwungen die Schlüssel regelmäßig zu wechseln und dabei starke komplexe Schlüssel zu wählen, so besteht das »Keymanagement« meist aus kleinen gelben Zetteln, die am Monitor oder unter der Tastatur kleben ;-).

Grundsätzlich ist ein Keymanagement sehr schwer zu implementieren, sobald die Aspekte menschlicher Benutzer berücksichtigt werden müssen. Dies ist umso zutreffender, je einfacher der Schlüssel ist. Kennwörter sind hier das ideale Beispiel. Meist ist es wesentlich einfacher mit einem Wörterbuch, in dem sämtliche Fussballmannschaften und ihre Spieler gespeichert sind, ein Kennwort zu ermitteln als den eigentlichen Verschlüsselungsalgorithmus zu analysieren.

Für den Einsatz eines Preshared Key zur Authentifizierung in Phase 1 des IKE-Protokolls wird dies eindrucksvoll demonstriert im Artikel *www.ernw.de/*

download/pskattack.pdf. Seit fast drei Jahren steht mit IKEcrack auch schon ein Werkzeug zur Verfügung (*http://sourceforge.net/projects/ikecrack*), das diesen Angriff durchführt. Dabei können die notwendigen Informationen für den Angriff aus dem Paketfluss extrahiert werden und mit IKEcrack in Ruhe auf großen Clustern analysiert werden. Wurde der PSK gefunden, kann die gesamte Kommunikation als Man-in-the-Middle abgehört werden.

4.1.4 Öffentliche Schlüssel

Öffentliche Schlüssel (Public Key) bieten für einige Probleme eine Lösung an. Hauptsächlich müssen die privaten Anteile nicht mehr zwischen allen beteiligten Systemen/Personen ausgetauscht werden, da es sich nicht um symmetrische Verfahren handelt. Es genügt der Austausch der öffentlichen Schlüssel.

Öffentliche Schlüssel sind meist so lang, dass ein normaler Mensch sie sich nicht einprägen kann. Dennoch müssen die privaten Schlüssel sicher aufgehoben werden. Hierzu werden sie meist mit einer Passphrase geschützt.

Dieser Schutz verhindert aber oft eine automatische Verwendung durch einen Dienst oder unabhängig arbeitende Software. Die Software ist nicht in der Lage den privaten Schlüssel durch Eingabe der Passphrase frei zu schalten, wenn die ihr nicht in Klartext übergeben wurde. Ein Reboot eines SSL-Webservers führt unweigerlich zu seinem fehlerhaften Start, da er seinen privaten Schlüssel für die SSL Verschlüsselung nicht laden kann. Daher werden die privaten Schlüssel meist in Klartext auf dem Rechner abgespeichert.

Damit ist ein Angriff gegen den Rechner über das Netzwerk oder physikalisch durch Einbruch und Diebstahl wesentlich erfolgversprechender als ein Angriff des kryptografischen Verfahrens.

Abhilfe schaffet hier nur eine Smartcard zur Speicherung des Private Key.

Weitere Probleme bei der Verwendung der öffentlichen Schlüssel ist die Überwachung der Lebensdauer und die automatische Deaktivierung der Schlüssel. Die Verteilung der Schlüssel ist ebenfalls sehr problematisch, da verhindert werden muss, dass eine dritte Person in der Lage ist den Schlüssel während der Übertragung abzufangen und gegen einen eigenen Schlüssel auszutauschen. Dies würde ansonsten einen Man-in-the-Middle Angriff ermöglichen.

Zertifikate können hier eine Lösung darstellen. Ihre Verwendung in einer Public Key Infrastruktur zur Verteilung und Überwachung der Schlüssel und der Einsatz von Smartcards werden in den nächsten Abschnitten besprochen.

4.2 X.509 Zertifikat

Wie bereits im letzten Abschnitt beschrieben, bieten Public Keys große Vorteile beim Einsatz zur Verschlüsselung und Authentifizierung. Die eingesetzten Verfahren und die gewählten Schlüssellängen sorgen mit einem zufällig erzeugten Schlüssel dafür, dass diese Verfahren mit heutigen Methoden nach menschlichem Ermessen nicht geknackt werden können. Angriffe erfolgen daher üblicherweise nicht direkt auf den Algorithmus, sondern auf die Schlüsselverwaltung.

Ein sicheres System für den Einsatz von öffentlichen Schlüsseln muss in der Lage sein, die folgenden Punkte zu leisten:

- Verteilung und Management der Schlüssel bieten (Lebensdauer)
- Sicherheit des privaten Schlüssels gewährleisten (Passphrase, Smartcard)
- Zuordnung der öffentlichen Schlüssel zu Personen oder Geräten ermöglichen.
- Vertrauensbeziehungen zwischen den Schlüsseln ermöglichen

Alle diese Funktionen können mit X.509 Zertifikaten erreicht werden. Die Erfüllung sämtlicher Funktionen bietet eine Public Key Infrastruktur, die die X.509 Zertifikate verwaltet. Zunächst sollen jedoch die Zertifikate beschrieben werden. Ihre Erzeugung mit dem Befehl `openssl` wird später beschrieben.

4.2.1 Aufbau des X.509 Zertifikates

Das X.509 Zertifikat wurde von der International Telecommunication Union (ITU, *http://www.itu.int*) 1988 als Teil des X.500 Verzeichnisdienstes entwickelt. Die aktuelle Version 3 des X.509 Standards wurde 1996 veröffentlicht. X.500 ist eine Datenbank, in der verschiedene bjekte (Entitäten) gespeichert werden können. Diese Datenbank ist für den weltweiten Einsatz konzipiert worden und bietet die Möglichkeit Personen und Ressourcen wie Computer oder Drucker zu verwalten. X.509 bot den Authentifizierungsdienst für das X.500 Verzeichnis. Das X.500 System konnte sich zwar nicht durchsetzen, aber die X.509 Zeritifikate sind heute ein allgemeiner Standard für die Verwendung von Zertifikaten.

Das X.509 Zertifikat bindet öffentliche Schlüssel an eine absolute Entität des X.500 Verzeichnisses (Distinguished Name, DN). Zusätzlich werden in einem Zertifikat der Version 3 der öffentliche Schlüssel und der verwendete Signaturalgorithmus, der Gültigkeitszeitraum und der Gültigkeitsbereich gespeichert.

Die wesentliche Eigenschaft, die das Zertifikat jedoch auszeichnet, ist die Unterschrift durch eine Zertifikatsautorität (CA). Diese Unterschrift bestätigt ähnlich einem Stempel die Echtheit des Zertifikates und seiner Gültigkeit. Die wesentliche Funktion ist jedoch die Zuordnung des Schlüssels zum Besitzer, der in dem Zertifikat angegeben wird. Diese Zuordnung wird von der CA bestätigt.

Sobald nun dieses Zertifikat eingesetzt werden soll, überprüft der Anwender, der das Zertifikat erhalten hat, ob die Signatur der CA gültig ist. Ist dies der Fall und vertraut er der CA, so kann er das Zertifikat für die Authentifzierung einsetzen.

TIPP

In Verbindung mit Virtuellen Privaten Netzwerken bieten Zertifikate sehr viele Vorteile. Der wesentliche Vorteil ist die einfache Schlüsselverwaltung. Wenn viele verschiedene Kommunikationspartner oder sogar möglicherweise eine unbekannte Menge an Partnern miteinander kommunizieren soll, so müssen entweder von Hand Preshared Keys (PSKs) oder öffentliche Schlüssel ausgetauscht werden. Werden Zertifikate eingesetzt, so müssen lediglich die Kommunikationspartner so konfiguriert werden, dass sie alle der Zertifikatsautorität vertrauen.

Ein digitales Zertifikat X.509 Version 3 enthält die folgenden Informationen:

- **Version** X.509 Version
- **Serial Number** Jede CA nummeriert die unterzeichneten Zertifikate eindeutig.
- **Signature Algorithm** Hier wird der Algorithmus eingetragen, mit dem die CA das Zertifikat unterzeichnet hat.
- **Issuer** Name der Zertifikatsautorität
- **Validity** Lebensdauer
- **Subject** Der X.500 Name des Inhabers
- **Subject Public Key Info** Informationen über den öffentlichen Schlüssel
- **X509.V3 Extensions** Erweiterungen, die es ermöglichen das Zertifikat eindeutig zu gestalten oder in seiner Gültigkeit einzuschränken. Hier können auch URLs für den Download von Rückruflisten (CRL) angegeben werden.

Im Folgenden ist ein Beispielzertifikat aufgeführt:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
CN=RootCA 2003/Email=ralf@spenneberg.net
        Validity
            Not Before: Apr 30 06:08:56 2003 GMT
            Not After : Apr 29 06:08:56 2004 GMT
        Subject: C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
CN=VPN-Gateway/Email=ralf@spenneberg.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c5:3b:9c:36:3a:19:6c:a9:f2:ba:e9:d2:ed:84:
                    33:36:48:07:b2:a3:2d:59:92:b0:86:4c:81:2c:ea:
                    5c:ed:f3:ba:eb:17:4e:b3:3a:cc:b7:5b:5d:ca:b3:
                    04:ed:fb:59:3c:c5:25:3e:f3:ff:b0:22:10:fb:de:
                    72:0a:ee:42:4b:9a:d3:27:d3:b6:fb:e9:88:10:c8:
                    47:b7:26:4f:71:40:e4:75:c4:c0:ee:6b:87:b8:6f:
                    c9:5e:66:cf:bb:e7:ad:72:68:b8:6d:fd:8f:4c:1f:
                    3a:a2:0d:43:25:06:b9:92:e7:20:6c:86:15:a0:eb:
                    7f:f7:0b:9a:99:5d:14:88:9b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                CB:5C:19:9B:E6:8A:8A:FE:0E:C4:FD:5E:DF:F7:BF:3D:A8:
18:7C:08
            X509v3 Authority Key Identifier:
                keyid:01:BB:C6:33:BE:F5:9A:5E:B0:0C:5D:BD:41:E9:78:
6C:54:AD:66:8E
                DirName:/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
CN=RootCA 2003/Email=ralf@spenneberg.net
                serial:00

    Signature Algorithm: md5WithRSAEncryption
        6f:89:2b:95:af:f1:8d:4d:b7:df:e8:6d:f7:92:fb:48:8c:c4:
        1a:43:68:65:97:01:87:a6:84:b5:a1:38:bd:62:74:70:db:9e:
        78:19:d9:0c:af:18:ad:13:77:56:7d:3f:19:61:da:ba:74:30:
        8e:c5:50:0e:e3:eb:ff:95:cd:8d:d6:7e:c3:0e:ab:5b:34:94:
        bc:16:0f:ef:dc:de:40:bb:7d:ba:a2:b8:5d:f9:74:e7:28:58:
```

```
75:a0:66:d2:8d:85:ba:38:82:08:10:33:ef:be:29:c9:31:9d:
63:a9:f7:e0:99:ea:a7:ed:b6:b5:33:1b:1c:4a:a4:05:40:6e:
40:7b
```

Die Erzeugung dieser Zertifikate mit Kommandozeilenbefehlen ist im Exkurs »Erzeugung von X.509 Zertifikaten mit OpenSSL« beschrieben. In einem weiteren Kapitel wird die Verwendung grafischer Werkzeuge beschrieben.

X.509 Zertifikate sind in der Lage die zu Beginn des Kapitels geforderten Punkte umzusetzen. Damit bieten sie ideale Vorraussetzungen für den Einsatz in VPN-Lösungen. Dennoch sollte ihr Einsatz genau geplant werden und möglicherweise über die Implementierung einer PKI nachgedacht werden.

4.3 Public Key Infrastruktur – PKI

Eine Public Key Infrastruktur bildet das Grundgerüst für den Einsatz von Zertifikaten in größeren Umgebungen. Solange sich die beteiligten Personen, die untereinander authentifiziert und verschlüsselt kommunizieren wollen, persönlich kennen, können sie direkt ihre öffentlichen Schlüssel austauschen. Sie erkennen und vertrauen einander und damit auch den direkt ausgetauschten Schlüsseln. Werden diese Gruppen jedoch größer und unübersichtlich, wie zum Beispiel in einem weltumspannenden Unternehmen, ist diese Form der Vertrauensbeziehung nicht mehr praktikabel. Ob nun E-Mails verschlüsselt oder ein VPN aufgebaut werden soll, es muss ein allgemein zugängliches und nachvollziehbares Gerüst existieren, das die Vertrauensbeziehungen aufbaut.

Dieses Gerüst wird möglich durch die Verwendung von Zertifikaten. Dabei können die Personen, die untereinander die Schlüssel austauschen möchten, Kontakt zu einer zentralen Autorität aufnehmen. Sie weisen Ihre Identität dieser Autorität gegenüber nach (Registrationsautorität) und erhalten ein digitales Zertifikat von der Zertifikatsautorität für den gleichzeitig vorgelegten öffentlichen Schlüssel in Kombination mit Ihrem Namen. Anschließend wird dieses Zertifikat zentral zur Verteilung gespeichert.

Erhält nun ein Benutzer eine unterzeichnete E-Mail, so wird entweder das Zertifikat, das für die Unterzeichnung verwendet wurde, direkt vom Absender mitgeschickt, oder der Empfänger fordert es von der zentralen Stelle an. Da die Zertifikate lediglich die öffentlichen Schlüssel enthalten, stellt dies keine Sicherheitslücke dar. Vertraut der Benutzer der Zertifikatsautorität, so überprüft er die Signatur des Zertifikates. Ist die Signatur und das Zertifikat

gültig, so vertraut er, dass dieses Zertifikat tatsächlich dem angegebenen Benutzer gehört und kann die Signatur der E-Mail überprüfen.

Die Gesamtheit des Zertifikatsmanagements und der Zertifikatsautorität bezeichnet man als PKI. Diese PKI ist verantwortlich, dass die Zertifikate erzeugt, verteilt und erneuert beziehungsweise gesperrt werden. Sie ist für die Verwaltung der Lebensdauer zuständig. Für die Anforderung von Zertifikaten (Certificate Requests) und die Verteilung von Zertifikaten und Rückruflisten (auch Sperrlisten) stellt sie geeignete Protokolle (HTTP, LDAP) zur Verfügung.

4.4 Smartcard

Selbst beim Einsatz einer PKI existiert noch ein großes Problem in der Schlüsselverwaltung: Wie und wo wird der private Schlüssel so gespeichert, dass eine Kompromittierung unmöglich oder möglichst schwer ist?

Insbesondere von kommerziellen Anbietern wird hier immer häufiger die Smartcard oder ein Security Token eingesetzt. Diese Geräte erlauben die Speicherung des privaten Schlüssels so, dass dieser anschließend nicht wiederhergestellt werden kann. Der Schlüssel kann nicht vom Benutzer und auch nicht von einem Angreifer ausgelesen werden. Damit ist der Schlüssel sicher vor Zugriff geschützt.

Werden nun Operationen benötigt, die einen Zugriff auf den Schlüssel verlangen, so verlagert die anfordernde Software die Berechnung auf die Smartcard. Dort wird die Operation mit dem privaten Schlüssel durchgeführt und das Ergebnis an die anfordernde Software zurückgeliefert.

Um die Karte bei einem Verlust oder bei Diebstahl zu schützen, wird in der Regel die Verwendung des privaten Schlüssels zusätzlich noch mit einer PIN geschützt. Auch dies ist nicht immer sicher. Es existieren unterschiedliche Kartenleser, bei denen die PIN entweder in dem Betriebssystem eingegeben wird oder direkt über eine Tastatur auf dem Kartenleser. Wird die PIN über das Betriebssystem eingegeben, so kann durch einen Keylogger diese PIN protokolliert werden. Dennoch ist durch einen Einbruch über das Netzwerk der Schlüssel nicht erreichbar. Auch kann der Schlüssel nicht ausgelesen werden. Damit besteht keine Möglichkeit andere verschlüsselte oder signierte Kommunikationen lesen oder modifizieren zu können. Es kann lediglich neu signiert oder verschlüsselt werden.

Seit dem FreeS/WAN-X.509-Patch 1.4.0 unterstützt dieser die Speicherung der Schlüssel auf einer Smartcard.

Teil II
Praktische Umsetzung

Es existieren momentan im wesentlichen zwei IPsec Implementierungen für Linux.

Zum einen existiert mit FreeS/WAN seit einigen Jahren eine relativ stabile Version, die sich sehr großer Beliebtheit erfreut. Diese wird in dem nächsten Kapitel genauer betrachtet. FreeS/WAN besteht aus einem Kernelmodul und entsprechenden Konfigurationswerkzeugen. Das Kernelmodul wurde jedoch nie von Linus Torvalds zur Aufnahme in den Linux Kernel akzeptiert. Die Gründe werden im weiteren in dem Kapitel erläutert. Dies führte auch dazu, das FreeS/WAN in einigen Distributionen (z. B. Red Hat) fehlt.

Zum anderen wurde beginnend mit dem Entwicklerkernel Version 2.5.45 von Dave Miller und Alexey Kuznetsov ein eigener IPsec Stack implementiert. Hierbei handelt es sich um einen IPsec Stack, der von dem USAGI Projekt (*http://linux-ipv6.org*) übernommen wurde. Diese IPsec Implementierung wird nun auch Einzug in alle Linux Distributionen finden. Diese wird in dem Kapitel Linux Kernel 2.6 IPsec besprochen.

5 FreeS/WAN

Das Projekt FreeS/WAN wurde 1996 von John Gilmore gestartet. Er verfolgte mit diesem Projekt ein hohes Ziel. IPsec für Linux sollte mindestens fünf Prozent des Internetverkehrs gegen Lauschangriffe schützen. Dieses sehr idealistische Ziel wurde bis heute leider nicht erreicht. Um so aktiver wird die Weiterentwicklung von FreeS/WAN vorangetrieben. Der Name S/WAN wurde von der Firma RSA zuerst verwendet, um ein sicheres Weitverkehrsnetz (Secure Wide Area Network) zu bezeichnen. Da das Projekt FreeS/WAN freie Software erzeugt, lag der Name nahe.

5.1 Einleitung

Im Jahr 1999 wurde Version 1.0 von FreeS/WAN veröffentlicht. Sämtliche Software des Projektes wird außerhalb der Vereinigten Staaten von Amerika entwickelt um mögliche Exportbeschränkungen der Programme zu verhindern. Bis heute nimmt das Projekt aus diesem Grund keine Hilfe von US amerikanischen Staatsbürgern an. Auch die Lockerung der Exportbestimmungen für starke Kryptografie Ende des Jahres 2000 hat hier keine Änderung veranlasst. Dies hat zu großen Missstimmungen und schließlich zur Entwicklung einer zweiten IPsec Implementierung geführt (siehe Kapitel 6, »IPsec mit Linux 2.6«). Diese Parallelentwicklung wurde von den Entwicklern auch mit der angeblich mangelnden Qualität des KLIPS Codes begründet. KLIPS ist der Kernelanteil von FreeS/WAN.

Wegen seines Alters ist das FreeS/WAN Projekt zu einer sehr mächtigen Implementierung gereift, die in der Vergangenheit die Interoperabilität mit vielen anderen Betriebssystemen und Netzwerkgeräten unter Beweis stellen konnte. Viele freiwillige Helfer haben das Projekt um unzählige Eigenschaften erweitert. Besonders hervorzuheben ist die hervorragende Unterstützung von X.509 Zertifikaten, die von Andreas Steffen (*http://strongsec.com/freeswan*) programmiert wird. Desweiteren existieren Patches, die zusätzliche Verschlüsselungsalgorithmen wie zum Beispiel den besonders schnellen AES Algorithmus unterstützen oder die Interoperabilität mit speziellen Betriebsystemen erhöhen.

Im Moment ist FreeS/WAN sicherlich das am häufigsten auf der Basis von Linux eingesetzte Produkt zur Erzeugung eines virtuellen Netzwerkes. Dies mag sich mit der Verbreitung des Linux Kernels 2.6 ändern, da dann eine

zweite Implementierung zur Verfügung stehen wird. Inzwischen existiert ein Patch von Herbert Xu (*http://gondor.apana.org.au/~herbert/freeswan/*, Mirror: *http://www.freeswan.ca/patches/gondor.apana.org.au/%257Eherbert/freeswan/*), mit dem FreeS/WAN auf dem aktuellem 2.6.0-Test-Kernel lauffähig ist. Die FreeS/WAN Werkzeuge und Konfigurationsdateien können dann transparent auf dem Linux Kernel 2.4 und 2.6 eingesetzt werden. Dieser Code ist in den aktuellen Snapshot-Versionen von FreeS/WAN enthalten. Zukünftige Versionen benötigen daher diesen Patch nicht mehr. Dieses Kapitel beschreibt die Installation und Konfiguration von FreeS/WAN basierend auf dem Linux Kernel 2.4, denn beim Linux Kernel 2.6 werden sich in Zukunft sicherlich noch Änderungen bezüglich der Installation ergeben. Hierbei werden sowohl die FreeS/WAN Versionen 1.9x als auch die neuen Versionen 2.x beschrieben. Die besondere Neuerung der Version 2.0 ist jedoch die direkte Unterstützung von opportunistischer Verschlüsselung und die Verwendung von Policy Groups. Da diese Policy Groups ganz neue Funktionen ermöglichen, werden diese in einem eigenen Abschnitt besprochen.

5.2 Lizenz

FreeS/WAN wird unter der GNU General Public License (GPL, siehe Anhang) vertrieben. Hierbei wurde jedoch vom Projekt die Einschränkung definiert, dass Code, der von US amerikanischen Staatsbürgern erzeugt wurde, nicht in das Projekt aufgenommen wird, um möglichen Beschränkungen durch Exportregelungen für starke Kryptografie zu verhindern. Dies führte bei Red Hat auch zum Verzicht auf die Einbindung von FreeS/WAN in die eigene Distribution.

Da FreeS/WAN unter der GNU GPL veröffentlicht wird, übernehmen die Programmierer des Projektes keinerlei Garantie für die Funktionsfähigkeit und Fehlerfreiheit der Software.

5.3 Installation

Die Installation von FreeS/WAN erfordert eine Modifikation des normalen Linux Kernels. Er enthält noch keine Unterstützung für die Verarbeitung von IPsec Paketen. In Abhängigkeit der gewählten Distribution wurde dies bereits vom Distributor durchgeführt. Es genügt dann meist die Installation der entsprechenden Pakete. Folgende Distributoren liefern FreeS/WAN bei ihren aktuellen Distributionen mit:

- SuSE Linux
- Conectiva Linux
- Mandrake Linux
- Debian
- Polish(ed) Linux
- Best Linux

Wird FreeS/WAN nicht bereits von der Distribution zur Verfügung gestellt – so zum Beispiel bei Red Hat Linux – so ist es erforderlich die Installation selbst vorzunehmen. Hierzu gibt es die Möglichkeit die Installation mit dem Red Hat Package Manager (RPM) durchzuführen oder selbst den Kernel zu patchen, zu kompilieren und anschließend zu installieren. Im Folgenden werden beide Wege beschrieben.

5.3.1 Installation mit RPM

Die Installation von FreeS/WAN mit der Unterstützung des RPM Systems ist sehr einfach. In vielen Fällen werden die entsprechenden RPM Pakete bereits von der Distribution zur Verfügung gestellt. Wenn die Pakete der Distribution sämtliche gewünschten Funktionen enthalten und eine akzeptable Aktualität aufweisen, kann FreeS/WAN mit Hilfe der RPM Pakete installiert werden.

Die Red Hat Linux Distribution enthält bisher keine FreeS/WAN RPM Pakete. Jedoch werden von verschiedenen Personen RPM Pakete für die jeweils aktuellen Kernel der Red Hat Linux Distributionen erzeugt:

- Das FreeS/WAN Projekt bietet selbst RPMs unter *ftp://ftp.xs4all.nl/pub/crypto/freeswan/binaries/RedHat-RPMs* an.
- Ken Bancoft stellt auf seiner Website *http://www.freeswan.ca* ebenfalls RPM Pakete für die Red Hat Distributionen bereit. Sie enthalten meist zusätzliche Funktionen, die über die oben genannten Pakete hinausgehen (siehe zusätzliche Patches 5.3.3, »FreeS/WAN Patches«). Es handelt sich in beiden Fällen um RPM Pakete, die lediglich die IPsec Module enthalten und zu einem vorhandenen Kernel hinzuinstalliert werden.
- Der Autor stellt auf seiner Website *http://www.spenneberg.org/VPN* ebenfalls RPM Pakete zur Verfügung. Diese Pakete enthalten komplette auf den Red Hat Linux Kerneln basierende Linux Kernel inklusive der notwendigen IPsec Module. Diese Kernel enthalten auch eine Unterstützung für die Microsoft Point-to-Point Encryption (MPPE), die für den Aufbau von PPTP VPN Lösungen benötigt wird.

Die Installation dieser Pakete ist recht einfach. Meist wird die FreeS/WAN Unterstützung in zwei Pakete aufgeteilt, zum Beispiel:

```
freeswan-1.99_2.4.18_18.8.0-0.i386.rpm
freeswan-module-1.99_2.4.18_18.8.0-0.i386.rpm
```

Listing 5.1 FreeS/WAN RPM Pakete auf http://www.freeswan.org

Dann erfolgt die Installation der Pakete mit:

```
# rpm -ivh  freeswan-module-1.99_2.4.18_18.8.0-0.i386.rpm
Warnung: freeswan-module-1.99_2.4.18_18.8.0-0.i386.rpm: V3 RSA/MD5
signature: NOKEY, key ID 5a7e4731
Preparing...                ########################################### [100%]
   1:freeswan-module        ########################################### [100%]
do not forget to install the userland utilities

# rpm -ivh freeswan-1.99_2.4.18_18.8.0-0.i386.rpm
Warnung: freeswan-1.99_2.4.18_18.8.0-0.i386.rpm: V3 RSA/MD5 signature:
NOKEY, key ID 5a7e4731
Preparing...                ########################################### [100%]
   1:freeswan               ########################################### [100%]
invoke "service ipsec start" or reboot to begin
```

Listing 5.2 Installation der RPM Pakete

Anschließend sollte überprüft werden, wo das verwendete RPM Paket seine Konfigurationsdateien abgelegt hat. Üblich sind die Verzeichnisse `/etc` und `/etc/ipsec.d`:

```
# rpm -qc freeswan
/etc/ipsec.conf
```

Hier ist nun zu erkennen, dass die wesentliche Konfigurationsdatei im Verzeichnis `/etc` zu finden ist. Dies kann jedoch von Distribution zu Distribution voneinander abweichen.

5.3.2 Kompilierung und Installation des Sourcecodes

Wenn die eingesetzte Linux Distribution keine Unterstützung für FreeS/WAN bietet und sie auch nicht von Dritten als RPM Paket zu erhalten ist, muss FreeS/WAN selbst übersetzt werden. Das ist auch nötig, wenn zusätzliche Funktionen, die momentan nur als Patches zur Verfügung stehen, eingebunden werden sollen oder die RPM Pakete der Distribution nicht die nötige Aktualität aufweisen.

Im Folgenden wird die Kompilierung und Installation des originalen Sourcecodes beschrieben. Eine Beschreibung der verfügbaren Patches folgt anschließend. Diese Patches müssen vor der Kompilierung in den Sourcecode eingebracht werden.

Für die Übersetzung von FreeS/WAN sind die Quellen des Linux Kernels erforderlich, da FreeS/WAN wesentliche Änderungen an ihm vornimmt. Hierzu können sowohl der offizielle Linux Kernel Sourcecode von *http://www.kernel.org* als auch die Quellen des Kernels der eingesetzten Linux Distribution verwendet werden. Die Distributions Kernel Quellen befinden sich meist in einem eigenen Paket: `kernel-source`. Zunächst sollte sichergestellt werden, dass dieses Paket installiert oder das Quelltextarchiv von *http://www.kernel.org* geladen und ausgepackt wurde:

```
# wget http://www.kernel.org/pub/linux/kernel/v2.4
  /linux-<version>.tar.gz
# cd /usr/src
# tar xzf /path/linux-<version>.tar.gz
# ln -s linux-<version> linux
```

Listing 5.3 Laden und Entpacken des Quelltextarchives

Wenn das Kernel Sourcecode Paket der Distribution verwendet werden soll, so ist darauf zu achten, dass es die Kernelquellen unter dem Pfad `/usr/src/linux` anbietet. Ansonsten sollte eine entsprechende Verknüpfung erzeugt werden.

ACHTUNG

Bei der Verwendung des Quelltextpaketes der Distribution kann es zu Problemen bei der Übersetzung kommen, da der Distributor möglicherweise Patches verwendet hat, die nicht kompatibel zum FreeS/WAN Patch sind.

Nun ist der Zeitpunkt gekommen die FreeS/WAN Quellen zu laden. Sie sind verfügbar unter *http://ftp.xs4all.nl/pub/crypto/freeswan/freeswan-<version>.tar.gz* und müssen geladen und anschließend entpackt werden:

```
# wget http://ftp.xs4all.nl/pub/crypto/freeswan/
  freeswan-<version>.tar.gz
# cd /usr/src
# tar xzf /root/freeswan-<version>.tar.gz
```

Listing 5.4 Laden und Entpacken von FreeSIWAN

Die Übersetzung von FreeS/WAN verlangt als Vorraussetzung einen Kernel Sourcecode, der bereits übersetzt wurde. Das soll dazu führen, dass die Benutzer den Kernel booten und testen – ohne FreeS/WAN zuvor übersetzen zu müssen. Diese Vorgehensweise wird empfohlen. Eine spätere Fehlersuche kann so vereinfacht werden. Wenn dieser Schritt übersprungen werden soll, so müssen zumindest die im Folgenden beschriebenen Schritte `make (old|-menu|x)config` und `make depend` für den Kernel ausgeführt werden. Ansonsten schlägt eine Übersetzung mit FreeS/WAN fehl.

Um die nun notwendige Konfiguration des Kernels zu vereinfachen, kann sie auf der Konfiguration des aktuell verwendeten Kernels aufbauen. Die meisten Distributionen speichern die aktuelle Kernelkonfigurationsdatei im Verzeichnis `/boot` ab. Die Datei trägt meist *config* als Namen gepaart mit der Versionsnummer des Kernels. Diese Datei sollte nun in das Kernel Verzeichnis kopiert werden:

```
# cp /boot/config-2.4.18-18.8.0 /usr/src/linux/.config
```

Listing 5.5 Kopieren der aktuellen Konfiguration (Red Hat Linux)

Um später die verschiedenen übersetzten Instanzen einer Kernelversion auseinanderhalten zu können, empfiehlt es sich den Wert `EXTRAVERSION` im Makefile (`/usr/src/linux/Makefile`) zu modifizieren. Tragen Sie dort zum Beispiel für die erste Übersetzung ohne FreeS/WAN *-noipsec1* ein.

Anschließend sollte der Befehl `make oldconfig` aufgerufen werden. Er passt die Konfigurationsdatei an den Kernel an und fragt bei neuen Optionen, ob sie aktiviert werden sollen. Wenn Sie bei neuen Optionen gefragt werden, ob sie unterstützt werden sollen, können Sie zunächst ohne Probleme meist *No* wählen. Eine Hilfe ist mit *?* verfügbar. Wenn Sie wünschen, können Sie nun weitere Änderungen am Kernel mit den üblichen Werkzeugen (`make xconfig` oder `make menuconfig`) vornehmen.

Anschließend sind die üblichen Schritte erforderlich (genauere Informationen entnehmen Sie bitte dem Kernel HowTo: *http://www.tldp.org*):

```
# make depend
# make bzImage
# make modules
# make modules_install
# make install
```

Listing 5.6 Kompilierung und Installation eines Linux Kernels

Nun sollte die Konfiguration des Bootmanagers überprüft werden und der Rechner mit dem neuen Kernel neu gestartet werden.

Beim Bootmanager Lilo muss sich in der Konfigurationsdatei ein neuer Eintrag für den neuen Kernel befinden. Dieser Eintrag kann folgendermaßen aussehen:

```
rompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32

image=/boot/vmlinuz-2.4.18-18.8.0
        label=linux
        initrd=/boot/initrd-2.4.18-14.img
        read-only
        append=\{root=LABEL=/\}

image=/boot/vmlinuz-<version>-noipsec1
        label=linux
        initrd=/boot/initrd-<version>-noipsec1.img
        read-only
        append=\{root=LABEL=/\}
```

Listing 5.7 Konfigurationsdatei /etc/lilo.conf

Nach Änderungen in dieser Datei ist ein Aufruf von Lilo erforderlich:

```
# /sbin/lilo
```

Listing 5.8 Aufruf von Lilo zur Übernahme der Konfigurationsänderung

Wenn der Bootmanager GRUB eingesetzt wird, muss die entsprechende Datei `/etc/grub.conf` angepasst werden. Dies ist üblicherweise ein Symlink auf die tatsächliche Konfigurationsdatei `/boot/grub/grub.conf`.

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.18-18.8.0)
        root (hd0,0)
        kernel /vmlinuz-2.4.18-18.8.0 ro root=LABEL=/
        initrd /initrd-2.4.18-18.8.0.img
```

```
title Kernel NoIPsec (2.4.18-noipsec1)
        root (hd0,0)
        kernel /vmlinuz-<version>-noipsec1 ro root=LABEL=/
        initrd /initrd-<version>-noipsec1.img
```

Listing 5.9 Konfigurationsdatei /etc/grub.conf

Damit der Bootvorgang erfolgreich ablaufen kann, ist bei den meisten modernen Distributionen eine initiale RAM Disk erforderlich, die wesentliche Treiber enthält. Sie wird beim Aufruf `make install` üblicherweise automatisch erzeugt und muss vor dem nächsten Bootvorgang existieren. Die einzelnen Distributionen verwenden unterschiedliche Befehle zu ihrer manuellen Erzeugung:

```
# mkinitrd /boot/initrd-<version>-noipsec1.img <version>-noipsec1
```

Listing 5.10 Red Hat Initrd

```
# mk_initrd -k vmlinuz-<version>-noipsec1 -i initrd-<version>-noipsec1
```

Listing 5.11 SuSE Initrd

Ist der Boot des neuen Kernels erfolgreich, so kann nun FreeS/WAN in den Kernel integriert werden. Hierbei sind grundsätzlich drei verschiedene Vorgehensweisen möglich:

- Integration und Installation von FreeS/WAN als fester Kernelbestandteil (empfohlen)
- Integration und Installation von FreeS/WAN als modulares Kernelmodul
- Erzeugung von zwei RPM Paketen, die getrennt die Kernelmodule und die Kommandozeilenwerkzeuge enthalten.

Alle drei Varianten haben ihre Vor- und Nachteile. Im Folgenden werden die drei verschiedenen Vorgehensweisen besprochen.

In allen Fällen erfolgt die Übersetzung aus dem FreeS/WAN Verzeichnis heraus.

TIPP

Bei der Übersetzung wird die Kernel Source im Verzeichnis `/usr/src/linux` erwartet. Wenn der Sourcecode sich an einer anderen Stelle befindet, so kann dies bei den `make` Aufrufen mit `KERNELSRC=<path>` definiert werden.

Vor der Konfiguration und Übersetzung sollten weitere erwünschte FreeS/WAN Patches eingespielt werden (siehe nächster Abschnitt) und die Variable `EXTRAVERSION` im `Makefile` des Kernels entsprechend angepasst werden.

1. Statisch im Kernel eingebunden:

```
# cd /usr/src/freeswan-<version>
# make oldgo
# make kinstall
```

2. Modular im Kernel eingebunden:

```
# cd /usr/src/freeswan-<version>
# make oldmod
# make minstall
```

3. Als RPM Paket:

```
# cd /usr/src/freeswan-<version>
# make oldrpm
```

   Die RPM Pakete befinden sich anschließend unterhalb des Verzeichnisses `./rpms/RPMS/<arch>`. Sie können anschließend mit dem Befehl `rpm -i <Paket>` installiert werden.

5.3.3 FreeS/WAN Patches

Das FreeS/WAN Softwarepaket ist in seiner Funktion recht beschränkt. Es erlaubt den automatischen Aufbau von VPN Verbindungen nur mit RSA Schlüsseln oder sogenannten Shared Secrets (oder Preshared Keys, PSK) zur Authentifizierung und mit Triple-DES zur Verschlüsselung. Weitere Authentifizierungsverfahren oder Verschlüsselungsverfahren werden nicht unterstützt. Des weiteren ist die Vergabe von IP Adressen mittels DHCP an den Client nicht möglich und eine Network Address Translation (NAT), wie sie häufig von Firewalls durchgeführt wird, verhindert oft den Aufbau von VPNs. Zur Unterstützung dieser Funktionen stehen eine Vielzahl von Patches zur Verfügung, die einzeln hinzugefügt oder im Ganzen geladen werden können. Ken Bancoft hat ein zentrales Repository zusammengestellt, wo die meisten dieser Patches heruntergeladen werden können. Sie ist unter *http://www.freeswan.ca* zu finden. Dort wird von Ken Bancoft auch ein Super FreeS/WAN Paket gepflegt, das die meisten dieser Patches bereits enthält.

X.509-Patch

Der X.509-Patch wird seit Jahren von Andreas Steffen gepflegt (*http://www.strongsec.com/freeswan*). Er befähigt FreeS/WAN die Authentifizierung auf der Basis von digitalen X.509 Zertifikaten durchzuführen. Dieser Patch existiert in zwei verschiedenen Varianten. Die Versionen 0.9.x sind für FreeS/WAN 1.9x und die Versionen 1.x.x für FreeS/WAN 2.x bestimmt.

TIPP

Sobald das aufzubauende VPN mehrere Verbindungen mit mehreren Knotenpunkten aufbauen soll, sollte der X.509-Patch verwendet werden. Die Verwendung von X.509 Zertifikaten und einer Zertifikatsautorität vereinfacht die Verwaltung des VPNs wesentlich. Zusätzlich unterstützt dieser Patch die Vergabe von dynamischen IP Adressen mittels DHCP-over-IPsec.[1] Auf dieser Webpage ist auch eine DHCP-Relay Distribution erhältlich.

Die Installation des Patches ist recht einfach. Hierzu wird der aktuelle Patch geladen, extrahiert und FreeS/WAN gepatcht.

```
# cd /usr/src
# tar xvzf x509-<version>.tar.gz
# cd freeswan-<version>
# patch -p1 < ../x509-<version>/freeswan.diff
```

Wichtig ist hierbei, dass die Version des X.509-Patches mit der Version des FreeS/WAN Patches übereinstimmt.

Anschließend wird FreeS/WAN wie oben beschrieben übersetzt.

Einige Funktionen des X.509-Patches müssen vor der Übersetzung durch Editieren der Datei `programs/pluto/Makefile` aktiviert werden. Die LDAP-Unterstützung und die Smartcard-Unterstützung sind per Default deaktiviert und können hier ab der Version 1.4 aktiviertiert werden.

Der Patch fügt eine große Zahl an weiteren Funktionen zu FreeS/WAN hinzu. Einige dieser Funktionen werden in diesem und in späteren Kapiteln besprochen. Hier soll nur kurz eine Auflistung der Funktionen und weiteren Direktiven in den Konfigurationsdateien erfolgen.

- Verwendung von Zertifikaten anstelle der RSA-Schlüssel. Zertifikate können mit der Direktive `leftcert` und `rightcert=<datei>` direkt angegeben werden. Wenn das Zertifikat der Gegenseite über das Netzwerk er-

1. Entsprechend dem Draft `draft-ietf-ipsec-dhcp-13.txt`.

halten wird, so kann dies durch die Angabe von `leftrsasigkey` und `rightrsasigkey=%cert` definiert werden.

- Für die Unterstützung von virtuellen IP Adressen, wie sie bei NAT und DHCP-over-IPsec auftreten können, unterstützt der X.509-Patch die Direktive `rightsubnetwithin|leftsubnetwithin`.
- Für die Einschränkung des im Tunnel erlaubten Verkehrs untersützt der X.509-Patch die Protokoll- und Portselektoren `leftprotoport` und `rightprotoport=<proto>/<port>`. Diese Direktiven werden auch für DHCP-over-IPsec benötigt (`leftprotoport=udp/bootpc`).
- Ab der Version 0.9.24 unterstützt der X.509-Patch die Verwendung von Wildcards in den Direktiven `leftid` und `rightid`: `leftid=»C=DE, O=Test, OU=VPN, CN=*«`.
- Es ist möglich, nicht die Zertifikate aller erlaubten CAs zu akzeptieren, sondern bestimmte Verbindungen auf bestimmte CAs zu beschränken. Hierfür existiert die Direktive `rightca|leftca=<ca-subject>`.
- Zertifikatswiderruflisten (CRL) können vom X.509-Patch automatisch geladen werden. Hierfür können die Protokolle LDAP, HTTP und FTP verwendet werden.
- Der X.509-Patch unterstützt die Speicherung der Schlüssel auf einer Smartcard (siehe Kapitel 9.11, »Smartcard Unterstützung«).

Notify/Delete SA Patch

FreeS/WAN baut im Gegensatz zu anderen IPsec Implementierungen dauerhafte Verbindungen auf. Sie werden nicht automatisch nach kurzer Zeit wieder abgebaut. Es existieren jedoch andere Implementierungen, wie zum Beispiel Windows 2000, die eine IPsec Verbindung ähnlich einer Wählverbindung bei Bedarf aufbauen und anschließend bei fehlender Benutzung nach einem Timeout abbbauen. Das erfolgt vor Ablauf der entsprechenden Zeitgeber für eine Neuverhandlung der Verbindung (Rekeying). Hierzu übermittelt Windows 2000 ein «Delete SA Announcement». Es teilt der Gegenstelle mit, dass gewisse Security Associations (SAs) nicht mehr gültig sind. FreeS/WAN unterstützt dies bisher nicht.

Um diese Unterstützung zu implementieren, hat Mathieu Lafon den Notify/Delete SA Patch geschrieben. Er bietet die folgenden Funktionen:

- Senden von Notification Nachrichten
- Senden von Delete SA Nachrichten für ISAKMP und IPsec SAs
- Empfang von Delete SA Nachrichten

Dieser Patch wurde von Mathieu Lafon bisher nur bis zur Version 1.98b gepflegt. Er funktioniert jedoch auch unverändert mit der Version 1.99. Ab der Version 2.00-pre8 sind wesentliche Teile des Patches in FreeS/WAN übernommen worden.

Der Patch ist verfügbar unter *http://open-source.arkoon.net/* oder unter *http://www.freeswan.ca/patches/*.

Die Installation des Patches ist recht einfach. Hierzu wird der aktuelle Patch geladen, extrahiert und FreeS/WAN gepatcht.

```
# cd /usr/src/freeswan-<version>
# zcat /path/notify_delete-freeswan-<version>.diff.gz | patch -p1
```

TIPP

Bevor der Patch tatsächlich eingespielt wird, sollte mit der Option `-dry-run` die Funktion getestet werden:

```
# zcat /home/spenneb/notify_delete-freeswan-<version>.diff.gz |
patch -p1 --dry-run
patching file pluto/connections.c
patching file pluto/connections.h
patching file pluto/demux.c
patching file pluto/ipsec_doi.c
patching file pluto/ipsec_doi.h
patching file pluto/spdb.c
patching file pluto/state.c
patching file pluto/state.h
```

NAT Traversal Patch

Der NAT Traversal Patch von Mathieu Lafon unterstützt die folgenden Internet Engineering Task Force (IETF) drafts:

- `draft-ietf-ipsec-nat-t-ike-01.txt`
- `draft-ietf-ipsec-udp-encaps-01.txt`
- `draft-ietf-ipsec-nat-t-ike-02.txt`
- `draft-ietf-ipsec-udp-encaps-02.txt`
- `draft-ietf-ipsec-nat-t-ike-03.txt`
- `draft-ietf-ipsec-udp-encaps-03.txt`
- `draft-ietf-ipsec-nat-t-ike-04.txt`
- `draft-ietf-ipsec-udp-encaps-04.txt`

Die Network Address Translation (NAT) erlaubt es einem Router oder einer Firewall, die Absender Adresse eines Clients gegen seine eigene auszutauschen. So ist es möglich erstens die tatsächliche Adresse eines Clients zu verbergen und zweitens, mehreren Clients einen gleichzeitigen Zugriff auf das Internet mit nur einer IP Adresse zu ermöglichen.

Damit die Firewall in der Lage ist, die Antwortpakete der Verbindungen den richtigen Clients zuzuordnen, pflegt sie eine Verbindungstabelle, die üblicherweise nach den verwendeten UDP oder TCP Ports sortiert ist. Eine Zuordnung erfolgt dann sehr einfach über den verwendeten Port. Jede nach außen gehende Verbindung nutzt hierzu einen eindeutigen Port auf dem Router oder der Firewall.

Im Fall von IPsec stößt dieses Verfahren schnell an seine Grenze. Die Protokolle AH und ESP verwenden keine Ports. Ein Router ist daher nicht in der Lage mit einfachen Mitteln die von außen kommenden AH oder ESP Pakete zwei verschiedenen Clients zuzuordnen.

Zusätzlich erlaubt das AH Protokoll nicht die Modifikation der Absenderadresse im IP Header. Bei der Berechnung des Hashes zur Integritätsüberprüfung wird die Absenderadresse mit einbezogen. Eine spätere Änderung führt zur Ablehnung des Paketes beim Empfänger, da es nicht mehr erfolgreich validiert. Lediglich das ESP Protokoll kann für einen Client über einen Router oder der Firewall, die NAT durchführt, unterstützt werden.

Es existieren unterschiedliche Verfahren, die versuchen dieses Problem zu umgehen. Die erfolgversprechenste Möglichkeit wurde unter der Federführung der Firma SSH (*http://www.ssh.com*) unter Beteiligung der Firmen F-Secure, Microsoft, Cisco Systems und Nortel Networks entwickelt. Hierbei handelt es sich um das sogenannte NAT Traversal. Das Verfahren wird in den oben aufgeführten Drafts beschrieben.

Diese Drafts können unter *http://search.ietf.org/* oder auf der beigelegten Buch-CD nachgelesen werden.

Beim NAT Traversal senden die beiden Kommunikationspartner zunächst eine spezielle NAT Traversal Vendor ID. Sie zeigt an, dass eine Unterstützung des NAT Traversals erfolgen kann. Anschließend sind die Kommunikationspartner in der Lage selbstständig das Vorhandensein und den Standort des NAT durchführenden Routers oder der Firewall zu erkennen. Sobald dies festgestellt wurde, wird der IKE Port auf den Port 4500 verlegt. Die anschließenden ESP Pakete werden dann gekapselt in UDP Paketen transportiert. Hierbei wird auch der Port verwendet, der für den IKE Austausch genutzt wurde (meist 4500).

Der NAT Traversal Patch ist verfügbar unter *http://open-source.arkoon.net/* oder unter *http://www.freeswan.ca/patches/*. Aktuell ist die Version 0.6, die auch FreeS/WAN 2.0 unterstützt.

Die Installation des Patches ist recht einfach. Hierzu wird zunächst der aktuelle Patch geladen und extrahiert. Anschließend stehen fünf verschiedene Patches zur Verfügung:

- **NAT Traversal-0.5-freeswan-1.99.diff**: Dieser Patch enthält lediglich den NAT Traversal Patch.
- **NAT Traversal-0.5-freeswan-1.99-nd.diff**: Dieser Patch kann angewendet werden, wenn zuvor der Notify/Delete SA Patch angewendet wurde.
- **NAT Traversal-0.5-freeswan-1.99-nd-x509-0.9.15.diff**: Dieser Patch kann angewendet werden, wenn zuvor der Notify/Delete SA Patch und der x509 Patch Version 0.9.15 angewendet wurden.
- **NAT Traversal-0.5-freeswan-1.99-nd-x509-0.9.15-alg.diff**: Dieser Patch kann angewendet werden, wenn zuvor der Notify/Delete SA Patch, der x509 Patch Version 0.9.15 und der Algo Patch (s.u.) angewendet wurden.
- **NAT Traversal-0.5-freeswan-1.99-nd-x509-0.9.18-alg.diff**: Dieser Patch kann angewendet werden, wenn zuvor der Notify/Delete SA Patch, der x509 Patch Version 0.9.18 und der Algo Patch (s.u.) angewendet wurden.

Wenn der richtige Patch ausgwählt wurde, kann er mit dem `patch` Kommando angewendet werden:

```
# cd /usr/src
# tar xvzf NAT Traversal-<version>.tar.gz
# cd /usr/src/freeswan-<version>
# patch -p1 < ../NAT Traversal-<version>/NAT Traversal-<version>.diff
```

Anschließend wird FreeS/WAN wie oben beschrieben übersetzt und installiert. Hierbei ist aber darauf zu achten, das die Unterstützung bei der Kernel Konfiguration aktiviert wird:

```
CONFIG_IPSEC_NAT_TRAVERSAL=yes
```

Die Unterstützung des NAT Traversal kann anschließend in der Konfigurationsdatei `/etc/ipsec.conf` mit der Direktive `nat_traversal` aktiviert werden:

```
nat_traversal = yes
```

Dieser Patch unterstützt auch die Verwendung virtueller IP Adressen. Diese Funktion kann zusammen mit der DHCP-over-IPsec des x509 Patches genutzt werden. Weitere Informationen können in Abschnitt 9.5, »NAT Traversal« nachgelesen werden.

Algo (AES) Patch

FreeS/WAN unterstützt sehr wenige Verschlüsselungsalgorithmen. Hierbei handelt es sich um 3DES, MD und SHA-1. JuanJo Ciarlante hat diese Auswahl mit dem sogenannten Algo Patch stark erweitert.

Dieser Patch umfasst die Unterstützung für:

- **AES**: Der Advanced Encryption Standard. AES ist zwei bis dreimal schneller als 3DES.
- **Twofish**: Twofish wurde von Bruce Schneier entwickelt und war Teilnehmer im AES Wettbewerb.
- **Blowfish**: Blowfish wurde ebenfalls von Bruce Schneier entwickelt.
- **Serpent**: Serpent war ein weiterer Teilnehmer beim AES Wettbewerb.
- **CAST**
- **SHA-2**: Eine Weiterentwicklung von SHA-1.
- **MD5**: Neu implementiert in Intel x86 Assemblercode für eine höhere Geschwindigkeit.
- **SHA-1**: Neu implementiert in Intel x86 Assemblercode für eine höhere Geschwindigkeit.

Der Patch wird in vielen verschiedenen Teilpatches geliefert. Hierbei werden die Patches getrennt in `common`, `klips` und `pluto`. Der `pluto` Patch existiert in zwei Versionen für FreeS/WAN mit und ohne X.509 Patch. Außerdem gibt es Patches für jeden Algorithmus:

```
freeswan-alg-0.8.0-enc-3des.diff.gz      -  P2   x86 optimized asm
freeswan-alg-0.8.0-enc-aes.diff.gz       P1 P2   128-256 bits
freeswan-alg-0.8.0-enc-blowfish.diff.gz  P1 P2   128-256 bits
freeswan-alg-0.8.0-enc-null.diff.gz      -  P2
freeswan-alg-0.8.0-enc-serpent.diff.gz   P1 P2   128-256 bits
freeswan-alg-0.8.0-enc-twofish.diff.gz   P1 P2   128-256 bits

freeswan-alg-0.8.0-auth-md5.diff.gz      -  P2   x86 optimized asm
freeswan-alg-0.8.0-auth-sha1.diff.gz     -  P2   x86 optimized asm
freeswan-alg-0.8.0-auth-sha2.diff.gz     P1 P2   sha2_256 and sha2_512
```

Die Angaben `P1` und `P2` geben an, ob der Algorithmus in der Phase 1 oder der Phase 2 eingesetzt werden kann.

Bei der anschließenden Konfiguration und Übersetzung des Kernels stehen neue Funktionen zur Verfügung:

```
IPSEC Modular Extensions (CONFIG_IPSEC_ALG) [Y/n/?] y
      AES encryption algorithm (CONFIG_IPSEC_ALG_AES) [M/n/y/?] m
      TWOFISH encryption algorithm (CONFIG_IPSEC_ALG_TWOFISH) [M/n/y/?] m
      und so weiter
```

Die entsprechenden Module können nach ihrer Installation mit dem `modprobe` Kommando geladen werden und stehen dann direkt als Algorithmus bereit. Ihre Auswahl erfolgt anschließend in der Datei `/etc/ipsec.conf` mit den Direktiven `ike`, `esp` und `ah`.

```
ike=aes128-sha
esp=aes128-sha1
```

Wenn diese Angaben in der Konfigurationsdatei fehlen, werden automatisch folgende Algorithmen gewählt:

```
ike=3des-md5-modp1536,3des-md5-modp1024,3des-sha-modp1536,
3des-sha-modp1024
esp=3des-md5,3des-sha1
```

Single DES Patch

FreeS/WAN unterstützt von Haus aus nur die 3DES Verschlüsselung. Der IPsec Standard verlangt jedoch auch eine Unterstützung von DES (1DES). Sie wurde bei der Implementierung von FreeS/WAN bewusst nicht unterstützt, da sie als unsicher angesehen wird. Die Schlüssellänge von 56 Bit ist nicht lang genug, um eine ausreichende Vertraulichkeit der übertragenen Informationen sicherzustellen.

Es existieren jedoch eine Reihe von IPsec Implementierungen, die lediglich in der Lage sind eine DES Verschlüsselung durchzuführen. Hierbei handelt es sich in erster Linie um Hardware Router und Firewalls (zum Beispiel Cisco PIX 506 mit einfacher VPN Lizenz), die zusätzlich über VPN Eigenschaften verfügen. Wenn diese Router eine VPN Verbindung mit FreeS/WAN aufbauen wollen, so muss FreeS/WAN DES unterstützen.

Matt Wright hat einen Patch erzeugt, mit dem diese Unterstützung erfolgen kann. Dieser Patch ist nicht gleichzeitig mit dem Algo Patch verwendbar.

Wenn der richtige Patch von *http://www.freeswan.ca/patches/1DES/* heruntergeladen wurde, kann er mit dem `patch` Kommando angewendet werden:

```
# cd /usr/src/freeswan-<version>
# patch -p1 < /path/freeswan-1des-<version>.diff
```

Anschließend wird FreeS/WAN wie oben beschrieben übersetzt und installiert. Hierbei ist aber darauf zu achten, das für die Verwendung von Single-DES die folgenden Parameter (`esp`) in der Datei `/etc/ipsec.conf` erforderlich sind:

```
esp = des-sha1-96
```

Listing 5.12 Aktivierung von 1DES in `/etc/ipsec.conf`

Cryptolib

FreeS/WAN implementiert sämtliche Verschlüsselungsalgorithmen in Software. Diese Implementierung ist nicht besonders effektiv. Der Algo Patch »Algo (AES) Patch« bietet hier bereits Verbesserungen. Dennoch sind sehr schnelle Prozessoren erforderlich um hohe Transfermengen zu verschlüsseln.

In solchen Fällen werden häufig Hardware Kryptoprozessoren eingesetzt. Sie unterstützen den Prozessor bei der Arbeit, in dem sie ihm die eigentliche Verschlüsselung abnehmen. So kann bereits ein einfacher Rechner der Pentium 133 Klasse mit einem zusätzlichen Hardware Kryptoprozessor bis zu 33 Mbit/s verschlüsseln.

Um diese Funktion für FreeS/WAN nutzen zu können, existiert ein Patch, der auch die HiFn 7901 und die HiFn 7811 Chips unterstützt: Cryptolib.

Cryptolib wird auf der Webpage *http://sources.colubris.com/en/projects/FreeS-WAN/* gepflegt. Dort befindet sich auch ein Dokument, das die Installation auf einem Linux Kernel der Version 2.4.x beschreibt (*http://sources.colubris.com/en/projects/FreeSWAN/7811_K2.4.txt*) und eine neuere Version des Original-Patches für FreeS/WAN Version 1.96 (*http://sources.colubris.com/en/projects/FreeSWAN/freeswan-1.96-cryptolib.patch*). Dies ist jedoch leider die aktuellste Version.

HiFn 7951 Patch

HiFn 7901 und HiFn 7811 sind nicht die einzigen verfügbaren Kryptoprozessoren. Der HiFn 7951 ist ebenfalls ein sehr beliebter Chip.

Um diese Funktion für FreeS/WAN nutzen zu können, existiert ein Patch, der auch die HiFn7951 Chips unterstützt. Dieser Chip wird zum Beispiel auf der VPN1200 PCI Karte der Firma Soekris (*http://www.soekris.com*) verwendet.

Der Treiber wurde von der Firma Security Data in Spanien (*http://www.securitydata.es*) entwickelt und unter der GPL Lizenz auf *http://sourceforge.net/projects/hifn7951/* veröffentlicht. Dieser Treiber basiert auf der oben vorgestellten Cryptolib.

Aggressive Modus Patch

Die IKE Verhandlungen der Phase 1 können in Main Modus oder Aggressive Modus ausgeführt werden. Das RFC verlangt spezifisch eine Implementierung des Main Modus und erlaubt zusätzlich optional die Implementierung des Aggressive Modus (siehe IKE Protokoll). Jedoch existieren einige IPsec Implementierungen, die auf dem Aggressive Modus bestehen, wenn eine Authentifizierung mit PSKs durchgeführt werden soll. Um sie unterstützen zu können hat Steve Harvey einen Patch für FreeS/WAN 1.5 erzeugt. Dieser Patch wurde von Henrik Nordstrom weiterentwickelt und an die Versionen 1.97 und 1.99 angepasst. Der Patch unterstützt momentan jedoch nur den Aggressive Modus als Client. Dieser Patch ist verfügbar unter *http://marasystems.com/download/freeswan/* und *http://www.freeswan.ca/patches/*. Momentan ist er nur für Super FreeS/WAN verfügbar.

Nach dem Download kann der Patch angewendet werden:

```
# cd /usr/src/freeswan-<version>
# patch -p1 < /path/super-freeswan-<version>-aggrmode.patch
```

Dieser Patch betrifft übrigens nur die FreeS/WAN Userspace Programme. Es ist also nicht erforderlich den Kernel neu zu übersetzen. Das Übersetzen und die Installation der Programme genügt. Eine Übersetzung der Programme kann erfolgen mit `make programs`.

Nach der Installation kann der Aggressive Modus mit der Direktive `aggrmode` in der `ipsec.conf` Datei angeschaltet werden:

```
aggrmode = yes
```

Die Defaulteinstellung ist `no`.

5.4 FreeS/WAN Komponenten

Das Produkt FreeS/WAN besteht aus mehreren verschiedenen Komponenten, die ineinander greifen. Hierbei handelt es sich um KLIPS, Pluto, den Kommandozeilenbefehl `ipsec` und die Konfigurationsdateien `ipsec.secrets` und `ipsec.conf`.

KLIPS ist ein Abkürzung für **K**erne**L** **IP**sec **S**upport. Es handelt sich hierbei um die Routinen im Linux Kernel, die sich um die Verschlüsselung und die Authentifizierung der IP Pakete mit den Protokollen ESP und AH kümmern. Außerdem erzeugen diese Routinen die ESP und AH IP Header für die Pakete und interpretieren bei ankommenden Paketen diese Header.

KLIPS ist kein fester Bestandteil des Linux Kernels. Wahrscheinlich wird er auch nie dazu werden, da einige namhafte Kernelprogrammierer von der Qualität des Codes nicht überzeugt sind. Daher wird KLIPS vom FreeS/WAN Projekt als Patch zur Verfügung gestellt.

Pluto ist der Internet Key Exchange (IKE) Daemon des FreeS/WAN Paketes. Er führt die Authentifizierung des Partner Gateways durch und handelt dann die Security Associations (SAs) aus. In der Phase 1 wird hierbei mit dem Main Mode die ISAKMP SA verhandelt. Basierend auf dieser SA werden in der Phase 2 mit dem Quick Mode die IPsec SAs verhandelt (siehe IKE Protokoll). Pluto unterrichtet anschließend KLIPS über die zu verwendenen ESP und AH Schlüssel und die Security Parameter Indices (SPI). Diese Angaben werden von KLIPS benötigt, um die Pakete zu erzeugen und zu empfangen. Schließlich passt Pluto auch die Routingtabellen und die Firewallregeln an die neuen Tunnel an. Die von Pluto benötigten Informationen werden aus den Konfigurationsdateien `ipsec.conf` und `ipsec.secrets` gelesen.

Das Kommandozeilenwerkzeug `ipsec` wird verwendet, um die verschiedenen Funktionen von FreeS/WAN, wie zum Beispiel das Starten oder Beenden eines Tunnels, zu steuern. Hierzu bietet das Werkzeug viele unterschiedliche Kommandomodule an, die zu einem Großteil jedoch für interne Aufgaben genutzt werden.

Sämtliche Konfigurationseinstellungen und die zu verwendenden Schlüssel für die Authentifizierung durch das IKE Protokoll werden in den Dateien `ipsec.conf` und `ipsec.secrets` gespeichert. Hierbei ist die Datei `ipsec.secrets` reserviert für die Schlüssel. Beide Dateien werden von Pluto und vom Kommandozeilenwerkzeug `ipsec` gelesen und ausgewertet.

Die Konfiguration der Dateien und die Verwendung des Kommandozeilenwerkzeuges werden in den weiteren Kapiteln genauer betrachtet.

5.5 Konfiguration von FreeS/WAN

Dieser Abschnitt beginnt mit einer Einführung in die verschiedenen Parameter und Werkzeuge des FreeS/WAN Projektes. Wenn Sie direkt ein Test VPN implementieren wollen, so steht es Ihnen frei zu den Abschnitten »Manuelle verschlüsselte Verbindungen« oder »Automatische verschlüsselte Verbindungen« vorzublättern. Jedoch denken Sie bitte daran, später hierher zurückzukehren und und die entsprechenden Grundlagen nachzulesen. Die drei folgenden Kapitel versuchen Ihnen mit einfachen Worten die Parameter, Befehle und Optionen zu erklären.

5.5.1 Allgemeine Konfigurationsparameter

FreeS/WAN verwendet in der Grundausstattung, also ohne Patches, zwei Konfigurationsdateien. Diese beiden Konfigurationsdateien sind üblicherweise unter `/etc` abgelegt und tragen die Namen `ipsec.conf` und `ipsec.secrets`. Wenn FreeS/WAN im Rahmen einer Distribution installiert wurde, so besteht die Möglichkeit, dass diese Dateien sich an einer anderen (zum Beispiel `/etc/ipsec.d`) Stelle befinden.

Die Datei `/etc/ipsec.conf` ist die zentrale Konfigurationsdatei. Sie enthält alle Startparameter, Tunneldefinitionen und so weiter. Sie wird beim Start von FreeS/WAN automatisch von den entsprechenden Werkzeugen gelesen und umgesetzt. Dazu wird diese Datei von einem automatischen Parser gelesen. Damit keine Fehler auftreten, ist die Syntax dieser Datei sehr wichtig. Die entsprechenden Werkzeuge, die diese Datei lesen, werden im nächsten Abschnitt besprochen.

Die Datei `/etc/ipsec.secrets` enthält die Schlüssel, die für die Authentifizierung beim Aufbau des Tunnels erforderlich sind. Diese Datei benötigt daher spezielle Rechte, die normalen Benutzern den Einblick verbieten. Üblicherweise besitzt die Datei die in Listing 5.13 gezeigten Eigenschaften.

```
-rw-------    1 root     root         3412 Sep  4 16:47 /etc/ipsec.secrets
```

Listing 5.13 Eigenschaften der Datei ipsec.secrets

Im Folgenden wird die Syntax der beiden Dateien genauer besprochen. Im Anhang findet sich darüber hinaus noch eine Referenztabelle die zum schnellen Nachschlagen der Parameter und der Syntax dienen kann. Die Version 2.x bietet einige zusätzliche Funktionen. Dies führt dazu, dass die Konfiguration für die Versionen 1.9x nicht unverändert übernommen werden kann. Die Unterschiede werden in einem eigenen Abschnitt besprochen.

ipsec.secrets

Die Datei `ipsec.secrets` ist üblicherweise eine sehr kurze Datei. Sie enthält meist nur einen Schlüssel, der für die Authentifizierung des FreeS/WAN Gateways bei seinem Kommunikationspartner genutzt wird. Eine typische `ipsec.secrets` Datei ist in Listing 5.14

```
: RSA {
        # RSA 2048 bits   kermit.spenneberg.de   Tue Sep  3 20:39:20 2002
        # for signatures only, UNSAFE FOR ENCRYPTION
        # pubkey=0sAQOSdvSey23TH66x1p7DaFNP/L87jv
Dv9SV+rHcZ4CgINsgS4Ku7xhbdUUO0pMIoP5Eqn8pJ7r1w871KB6RTJLL21
```

```
Mdb3GcEXLcfbnO6QLBFRq63xWz05aYwkXdtTyjqCtt6GCzn2e9b43ejxCM
vXIsaW7JaBl8T/NiUCOmsAAYgDwsk3EVyQCLARF8er2OOAk0KvMVHmx6
eBUyX6FzHtJBwtbJPvHk0pz0/iIxizGqyLm8bMzTRugkr+FoWsA+X7P8Xibz3
VzFgaizSQOOVTp995a/vt8mglf/lHZwDKMsURQKA4wbZMIgbhHpmQ3sz/C
GjE0YiCypGI1rPVk51qbxr
        #IN KEY 0x4200 4 1
AQOSdvSey23TH66x1p7DaFNP/L87jvDv9SV+rHcZ4CgINsgS4
Ku7xhbdUUO0pMIoP5Eqn8pJ7r1w87lKB6RTJLL2lMdb3GcEXLcfbnO6QLBF
Rq63xWz05aYwkXdtTyjqCtt6GCzn2e9b43ejxCMvXIsaW7JaBl8T/NiUCOms
AAYgDwsk3EVyQCLARF8er2OOAk0KvMVHmx6eBUyX6FzHtJBwtbJPvHk0
pz0/iIxizGqyLm8bMzTRugkr+FoWsA+X7P8Xibz3VzFgaizSQOOVTp995a/vt8
mglf/lHZwDKMsURQKA4wbZMIgbhHpmQ3sz/CGjE0YiCypGI1rPVk51qbxr
        # (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
        Modulus:
0x9276f49ecb6dd31faeb1d69ec368534ffcbf3b8ef0eff5257eac7719e0
280836c812e0abbbc616dd5143b4a4c2283f912a9fca49eebd70f3b94a
07a45324b2f694c75bdc67045cb71f6e73ba40b04546aeb7c56cf4e5a6
3091776d4f28ea0adb7a182ce7d9ef5be377a3c4232f5c8b1a5bb25a06
5f13fcd89408e9ac0006200f0b24dc45724022c0445f1eaf638e024d0ab
cc5479b1e9e054c97e85cc7b49070b5b24fbc7934a73d3f888c62cc6ab2
2e6f1b3334d1ba092bf85a16b00f97ecff1789bcf75731606a2cd240e39
54e9f7de5afefb7c9a095ffe51d9c0328cb14450280e306d930881b847a
66437b33fc21a31346220b2a46235acf564e75a9bc6b
        PublicExponent: 0x03
        # everything after this point is secret
        PrivateExponent:
0x186928c521e7a32ff272f91a75e6b88d54ca89ed2827fe30ea7213d
9a55c015e76add01c9f4bae7a3835f370cb06b542dc6ff70c5274e8289e
e1abf0b8861dd3c3768f4f6680ba1e853d1349b572b6367273f63cd37b9
bb2c2e93ce286d1ac79e9aeb226a4528f5093f0a0b087e4c1d9b9f30f01
0fd8aa2418ac26f200010557969e5ea64d52224915fef9a1d464f6ce03
7cf84b8ccb84c4c4681e492c15c53f65da129821b98a94c702c7039b06
951c1934df6d3d7d9a57e01b75a703235455865c4e19021a500acedc9d
5d4f7a1b580b15f3a8e7e237630addad9a22bfd23d36b32ba5af9356a9e
4fb34b9aa775ab1485e1e18d05905b3c445c50d6f9b6833
        Prime1: 0xc7480084d957ca060c47852d9cfa3801d258445366064
c0c57947e65354ba49ddc935ab1a898a65f00de0565481934daa8d2812
e63ad254618dec99508d7e022a713f3394e78fe9c97274dab1df732a4f
a6179ba7296fb96c2a501368cb8cd94610ad2daf5b4154cd1257d1c034
78d1447d3d5aefa59aff4ef7a3e89911f5719
        Prime2: 0xbc26a3dac8fb8b03b41dbfb6cc3601772376af2ee853
355c5a92b44089e64c567602857a0642c1619499dce7e229feab2f0b75
76fd1b45d69ed68b30f3ec12db4a4bf527fc1a818cbc7f43089a7ac8c8a
8cac007e7bc4e16e146892fcb93594159430a49c6086ad35571accd792
04ee5a90abbde2eba73f7d1b2797446e5f423
        Exponent1: 0x84daab033b8fdc04082fae1e68a6d00136e582
e2440432b2e50da998ce326dbe930ce7211b106eea00940398dabb7891c
5e1ab7442736e2ebb3f310e05e540171a0d4cd0defb546864c4de7213fa
21c351965126f70f526481c356245dd0890d9607373ca3cd63888b6e5
```

```
368022fb362da8d391f51911ff89fa6d45bb614e4bb
        Exponent2: 0x7d6f17e730a7b2027813d524882400fa17a474c9f03
778e83c61cd805beedd8ef95703a6aed72b9663113defec1bff1cca07a3
a4a8bcd939bf39b220a29d61e786dd4e1aa811abb32854d75b11a73085
c5dc80054528340f40d9b0ca87b790d63b8206dbd95af1e238f67333a61
589ee70b1d2941f26f7fa8bcc50f82f43f817
        Coefficient: 0x875387df7124323c389952be8a3c8a8275c83db0
e130793b7773cbb75bfeda3b8e32c94ea2a9e6b88d01f74f3683b472a1
5c268ab4e1f6a1ecf410c4a2077a45620f0d1632c96e2a16d67c771c14
3e613de4aef2be0db7e12e20be50d5a2d542f28bcd6a24f471e6579460b
0019735d470afa10dac2e0650869462715bdcf3d5
    }
```

Listing 5.14 Die Datei `ipsec.secrets`

In dieser Datei werden Preshared Keys (PSKs) und die privaten Schlüssel (Private Key) des Gateways abgespeichert. Defaultmäßig wird bei FreeS/WAN auch der öffentliche Schlüssel (Public Key) in dieser Datei als Kommentar hinterlegt. Der öffentliche Schlüssel wurde in Listing 5.13 fett unterlegt.

Die allgemeine Syntax dieser Datei ist relativ einfach. Es handelt sich um eine Tabelle mit zwei Spalten, die durch einen Doppelpunkt voneinander getrennt sind. Die erste Spalte enthält die Identitäten der Rechner, für die die entsprechenden Schlüssel in der zweiten Spalte genutzt werden dürfen. Ein Beispiel:

```
# sample /etc/ipsec.secrets file for 10.1.0.1
10.1.0.1 10.2.0.1: PSK "secret shared by two hosts"

# an entry may be split across lines,
# but indentation matters
www.xs4all.nl @www.kremvax.ru
   10.6.0.1 10.7.0.1 1.8.0.1: PSK "secret shared by 5"

# an RSA private key.

@my.com: rsa {
   Modulus: 0syXpo/6waam+ZhSs8Lt6jnBzu3C4grtt...
   PublicExponent: 0sAw==
   PrivateExponent: 0shlGbVR1m8Z+7rhzSyenCaBN...
   Prime1: 0s8njV7WTxzVzRz7AP+0OraDxmEAt1BL5l...
   Prime2: 0s1LgR7/oUMo9BvfU8yRFNos1s211KX5K0...
   Exponent1: 0soaXj85ihM5M2inVf/NfHmtLutVz4r...
   Exponent2: 0sjdAL9VFizF+BKU4ohguJFzOd55OG6...
   Coefficient: 0sK1LWwgnNrNFGZsS/2GuMBg9nYVZ...
   }
```

Listing 5.15 Syntax der Datei `ipsec.secrets` (Beispiel aus der Manpage)

Als Identitäten in der ersten Spalte sind erlaubt:

- IP Adressen: 10.0.0.1
- DNS Namen: `www.spenneberg.net`. Diese DNS Namen werden beim Einlesen der Schlüssel zu ihren entsprechenden IP Adressen aufgelöst. Diese Auflösung erfolgt also nicht regelmäßig.
- Platzhalter: `%any`, `%any6`, nichts
- E-Mail Adressen ähnliche Konstrukte: `ralf@spenneberg.net`, `@www.spenneberg.net`. Sie werden lediglich als Text aufgefasst und nicht aufgelöst.

TIPP

Ein Neueinlesen der Datei `/etc/ipsec.secrets` um die erneute Auflösung der DNS Namen zu erzwingen kann mit dem Befehl

```
ipsec auto --rereadsecrets
```

erfolgen.

Die Zuordnung der entsprechenden Identitäten zu den verschiedenen Tunneln erfolgt in der Datei `/etc/ipsec.conf` (siehe unten) mit den Parametern `leftid` und `rightid`.

```
leftid=ralf@spenneberg.net
rightid=@www.spenneberg.net
```

In der rechten Spalte befinden sich die geheimen Schlüssel für die Authentifizierung der Verbindung. Hierbei kann es sich um Preshared Keys (PSK) oder um RSA Schlüssel handeln.

Ein PSK wird üblicherweise zwischen doppelten Anführungszeichen angegeben. Vor dieser Zeichenkette muss sich die Buchstabenkombination `psk` befinden, um den Preshared Key anzuzeigen. Diese PSK darf jedes Zeichen enthalten außer dem doppelten Anführungszeichen und einem Zeilenumbruch. Alternativ kann die PSK auch als hexadezimale Zahl (mit Prefix 0x) und als Base64 kodierte Zeichensequenz (mit Prefix 0x) angegeben werden. Die Angabe als ASCII in Anführungszeichen hat sich jedoch etabliert, da die meisten anderen Betriebsysteme die gleiche Darstellung wählen.

Werden bei einem PSK Schlüssel mehrere Identitäten angegeben, so wird nach dem Schlüssel gesucht, der mindestens für die Identität des lokalen Rechners und möglicherweise zusätzlich für die Identität des Partners konfiguriert ist. Das wird am einfachsten an einem Beispiel deutlich:

```
www.xs4all.nl @www.kremvax.ru
   10.6.0.1 10.7.0.1 1.8.0.1: PSK "secret shared by 5"
www.xs4all.nl: PSK "default shared"
```

Listing 5.16 Zuordung von PSK Schlüsseln

Soll in diesem Beispiel eine Verbindung aufgebaut werden zwischen `www.xs4all.nl` und `10.6.0.1`, so wird der erste Schlüssel gewählt. Wenn jedoch der Schlüssel für eine Verbindung von `www.xs4all.nl` und `www.spenneberg.net` gesucht wird, so wird der zweite Schlüssel gewählt.

Ein privater RSA Schlüssel besteht aus acht relativ großen Zahlen. Sie werden als Liste mit geschweiften Klammern umschlossen. Sinnvoll ist es den RSA Schlüssel über mehrere Zeilen anzugeben. Dann müssen aber alle folgenden Zeilen eingerückt werden. Dies trifft selbst dann auf die letzte Zeile zu, wenn sie lediglich die schließende geschweifte Klammer enthält.

Der Schlüssel wird mit der Direktive `rsa` in der Datei eingetragen. Die Identitäten, die diesem Schlüssel zugewiesen werden, gelten alle als lokale Identitäten. Es ist bei einem RSA Schlüssel nicht möglich unterschiedliche Schlüssel in Abhängigkeit des Partners zu verwenden.

TIPP

Bei der Konfiguration der VPN Tunnel in der Datei `/etc/ipsec.conf` ist es möglich für die verschiedenen Tunnel unterschiedliche lokale Identitäten zu definieren. So können für verschiedene Tunnel unterschiedliche RSA Schlüssel zum Einsatz kommen.

Die Erzeugung der RSA Schlüssel kann mit dem Befehl `ipsec rsasigkey` erfolgen. Hierbei ist lediglich die Angabe der Schlüssellänge (zum Beispiel 2048) erforderlich (siehe unten). Um aber gleich die richtige Syntax für diese Datei zu erzeugen, kann der Befehl `ipsec newhostkey` genutzt werden.

Wenn aus organisatorischen oder administrativen Gründen die Schlüssel in unterschiedlichen Dateien aufgeführt werden sollen, so besteht die Möglichkeit diese mit der `include` Direktive zu laden. Hierbei ist es möglich auch Globbing mit Wildcards (?*) zu nutzen.

Zusammenfassung

Syntax:

1. Jede Einheit aus Identität und Schlüssel beginnt in der ersten Spalte.
2. Müssen mehrere Zeilen verwendet werden, so werden Folgezeilen eingerückt. Dies gilt auch für die letzte schließende Klammer bei RSA Schlüsseln!

3. Alle Angaben auf einer Zeile werden durch Leerzeichen voneinander getrennt. Ein Zeilenumbruch gilt als Leerzeichen.
4. Als Schlüssel können RSA Schlüssel oder PSK (Preshared Keys) Schlüssel mit den entsprechenden Direktiven `rsa` und `psk` angegeben werden. Groß-/Kleinschreibung ist hierbei unerheblich.
5. Die Funktion der Identitäten hängt von den verwendeten Schlüsseln ab. Allgemein lässt sich sagen, dass sowohl IP Adressen, DNS Namen und E-Mail-Adressen als Identitäten möglich sind. Die Platzhalter `%any` oder `0.0.0.0` treffen auf jede IPv4 und `%any6` auf jede IPv6 Adresse zu. Wenn die Identität fehlt, handelt es sich um einen Defaulteintrag, der auf jede Verbindung zutrifft.
   - **PSK:** Hierbei besteht die Möglichkeit nur die Identität des lokalen Rechners oder auch zusätzlich die Identität des Partners anzugeben.
   - **RSA:** Die aufgeführten Identitäten gelten lediglich als unterschiedliche Identitäten des lokalen Rechners. Der Partner kann bei RSA Schlüsseln nicht angegeben werden.

ipsec.conf

Die Datei `ipsec.conf` enthält die meisten Konfigurationseinstellungen bei FreeS/WAN. In der Datei `ipsec.secrets` sind lediglich die Schlüssel für die Authentifizierung beim Aufbau der Tunnel. Alle Einstellungen in der Datei `ipsec.conf` dürfen öffentlich bekannt sein und sind nicht kritisch. Die einzige Ausnahme sind die Parameter bei einer Verbindung mit manuell festgelegten statischen Schlüsseln (Manual-keyed Connection).

ACHTUNG

Die Konfiguration von FreeS/WAN unterscheidet sich zwischen der Version 1.9x und 2.x. Eine wesentliche neue Eigenschaft von Version 2.0 sind die Policy Groups. Sie ermöglichen die einfache Konfiguration der opportunistischen Verschlüsselung. Sie wird in einem späteren Kapitel besprochen. Um FreeS/WAN klassisch einzusetzen, müssen die Policy Groups deaktiviert werden. Am Ende dieses Kapitels gibt es hierzu einige Hinweise (Abschnitt 5.6, »FreeS/WAN 2.x«).

Die Datei `ipsec.conf` besteht üblicherweise aus drei oder mehr Abschnitten:

- **`config setup`** Hier werden allgemeine Parameter für den Start von FreeS/WAN definiert.

- **conn %default** Diese Default Verbindung erlaubt die Definition von Standardwerten für alle Verbindungen.
- **conn client-airport** Dies ist die Definition der VPN Verbindung *Client-Airport*. Hier können nun mehrere VPN Verbindungen nacheinander definiert werden.

Die drei Abschitte werden durch die oben angegebenen Direktiven getrennt. Alle Parameter, die innerhalb eines Abschnittes nun definiert werden sollen, müssen eingerückt werden. Listing 5.17 zeigt eine typische Datei.

```
# Wenn es sich um FreeS/WAN in der Version 2 handelt ist die
# folgende Zeile erforderlich:
# version 2

config setup
        interfaces=%defaultroute # Default ab Version 2
        klipsdebug=none
        plutodebug=none
        plutoload=%search        # Default ab Version 2, aber
                                   als Option
        plutostart=%search       # Default ab Version 2, nicht
                                   mehr vorhanden!
        uniqueids=yes            # Default ab Version 2

conn %default
        keyingtries=0            # Default ab Version 2 (%forever)
        authby=rsasig            # Default ab Version 2

conn newyork-berlin
        left=3.0.0.1
        leftid=@newyork
        leftrsasigkey=0sAQOSdvSey23TH66x....
   leftnexthop=3.255.255.254
        right=5.0.0.1
        rightid=@berlin
        rightrsasigkey=0sAQNlGOWeO66bPgi....
        rightnexthop=5.255.255.254
        auto=start
```

Listing 5.17 Die Datei `ipsec.conf`

Diese Datei – in Kombination mit einer entsprechenden `ipsec.secrets` – genügt bereits, um eine VPN Verbindung aufzubauen. Die Konfiguration ist also gar nicht kompliziert. Bevor nun jedoch eine VPN Verbindung tatsächlich aufgebaut wird, sollen kurz alle wichtigen Direktiven der Datei besprochen werden. Die folgende Erläuterung kann später im Zusammenhang mit der Referenz im Anhang als Nachschlagewerk dienen.

Ein Parameter, der an jeder Stelle genutzt werden kann, ist der `include` Parameter. Er erlaubt, weitere Dateien an der entsprechenden Stelle in die Datei `/etc/ipsec.conf` einzulesen und auszuwerten.

Setup Parameter

Im `config setup` Bereich wird definiert, wie FreeS/WAN gestartet werden soll. Hier können eine ganze Reihe Angaben gemacht werden, die anschließend das Verhalten von FreeS/WAN steuern.

Zunächst sollen nun die häufigen Parameter besprochen werden:

- **`interfaces`** Diese erforderliche Angabe definiert die zu verwendenden Netzwerkkarten. Dies erfolgt, indem die virtuellen `ipsec` Interfaces den physikalischen zugeordnet werden. Jedem physikalischen Interface, das für den Aufbau von VPN Tunneln verwendet werden soll, muss auf diese Weise ein virtuelles Interface zugewiesen werden. Hierbei stehen insgesamt vier virtuelle Interfaces (`ipsec0` – `ipsec3`) zur Verfügung. Dabei ist es wichtig, dass die Zuordnung in doppelte Anführungszeichen gefasst wird und mehrere Zuordnungen durch Leerzeichen getrennt werden, zum Beispiel:

  ```
  interfaces="ipsec0=eth0 ipsec1=eth1"
  ```

  In den meisten Fällen kann hier jedoch als Wert `%defaultroute` angegeben werden. Dann ermittelt FreeS/WAN beim Start automatisch das Interface, über das das Standard Gateway erreicht wird. Dies ist in den meisten Fällen auch das Interface, über das die verschlüsselten Pakete versendet werden sollen. Ab Version 2 ist diese Einstellung der Default-Wert.

  ```
  interfaces=%defaultroute
  ```

- **`klipsdebug`** Dieser Parameter definiert den Detailgrad des KLIPS Protokolls beim Start von FreeS/WAN. Dieser Wert kann zu einem späteren Zeitpunkt mit dem Befehl `ipsec klipsdebug -option` geändert werden. Ist er leer oder `none`, so wird von KLIPS keinerlei Debug Information ausgegeben. Dies sollte die Standardeinstellung sein, da ansonsten sehr ausführliche Protokolle erzeugt werden. Diese Debug Informationen können sehr wertvolle Tipps liefern, wenn die Kommunikation mit anderen IPsec Implementierungen fehlschlägt. Der Wert `all` aktiviert alle folgenden Debug Informationen:
  - **`tunnel`** Meldungen des Tunnelcodes
  - **`tunnel-xmit`** Meldungen der verschickten Pakete des Tunnels
  - **`pfkey`** Kommunikation mit den Kommandozeilenbefehlen

- **xform** Auswahl der Transformation der Pakete
- **eroute** Manipulation der IPsec Routen (eroute)
- **spi** Manipulation der Security Association Datenbank
- **radij** Manipulation der Radij Datenbank
- **esp** Encapsulation Security Payload Protokoll
- **ah** Authentication Header Protokoll
- **ipcomp** Kompression der IP Pakete vor der Verschlüsselung
- **verbose** liefert die meisten Informationen. **Achtung!** Diese Funktion gibt auch Authentifizierungs- und Verschlüsselungsschlüssel in den Protokolldateien aus.

Wenn mehrere Parameter bei `klipsdebug` angegeben werden sollen, so müssen sie durch Leerzeichen getrennt und in doppelte Anführungszeichen gefasst werden.

- **plutodebug** Dieser Parameter erlaubt die Einstellung der entsprechenden Debug Informationen für Pluto beim Start von FreeS/WAN. Dieser Wert kann zu einem späteren Zeitpunkt mit dem Befehl `ipsec pluto --debug-<option>` geändert werden. Er unterstützt ebenso die Einstellung `none` und `all`. Zusätzlich können die folgenden Debug Informationen einzeln aktiviert werden:
  - **control** protokolliert die von Pluto getroffenen Entscheidungen (sinnvoll bei der Fehlersuche)
  - **crypt** protokolliert die Ver- und Entschlüsselung der Nachrichten
  - **dns** zeigt die DNS Anfragen für KEY und TXT Einträge (bei `rsasigkey=%dns` und opportunistischer Verschlüsselung, sinnvoll bei der Fehlersuche)
  - **emitting** zeigt die Struktur der ausgehenden Nachrichten
  - **klips** protokolliert die Kommunikation mit KLIPS
  - **lifecycle** temporäre Option, protokolliert die Lebensdauer der Security Associations (SA)
  - **parsing** zeigt die Struktur der eingehenden Nachrichten (sinnvoll bei der Fehlersuche)
  - **private** protokolliert auch die privaten Schlüssel im Protokoll
  - **raw** zeigt sämtliche übertragenen Bytes an (raw bytes)
- **plutoload** Bevor ein Tunnel aktiviert werden kann, muss er zunächst geladen werden. Beim Start können mit `plutoload` bestimmte Tunnel automatisch geladen werden. Erlaubte Werte sind `none`, der Name einer Verbindung, eine Liste von Verbindungen in Anführungszeichen durch

Leerzeichen getrennt oder der Parameter `%search`. Der letzte Parameter bewirkt, dass beim Start sämtliche Verbindungen mit `auto=add|route|start` geladen werden. Sinnvoll ist hier der Eintrag `%search`. Diese Einstellung ist ab Version 2 der Default-Wert. Die Angabe führt in Version 2 zu einem Fehler.

- **`plutostart`** Diese Direktive `plutostart` erlaubt es automatisch beim Start von FreeS/WAN auch die entsprechenden angegebenen Tunnel zu starten. Erlaubte Werte sind `none`, der Name einer Verbindung, eine Liste von Verbindungen in Anführungszeichen durch Leerzeichen getrennt oder der Parameter `%search`. Der letzte Parameter bewirkt, dass bei dem Start sämtliche Verbindungen mit `auto=route|start` geroutet und mit `auto=start` gestartet werden. Sinnvoll ist hier der Eintrag `%search`. Lediglich für die Fehlersuche empfiehlt sich hier der Eintrag `none`, so dass die Verbindungen von Hand gestartet und die Protokolle verfolgt werden können. Diese Einstellung ist der Default-Wert ab Version 2. Die Angabe führt in Version 2 zu einem Fehler.
- **`uniqueids`** Diese Einstellung definiert, ob mit einer ID gleichzeitig Verbindungen von unterschiedlichen IP Adressen aufgebaut werden dürfen. Wenn dieser Wert auf `yes` gesetzt wird, so werden bei einer neuen Verbindung alle anderen Tunnel derselben ID und unterschiedlicher IP Adresse gelöscht. Default-Wert ist `uniqueids=no`. Sinnvoll ist hier `yes`, wenn nicht zum Beispiel Roadwarrior mit PSK zur Authentifizierung alle dieselbe ID einsetzen. Ab Version 2 ist der Default-Wert `yes`.
- **`overridemtu`** Mit diesem Wert kann die Maximum Transmission Unit (MTU) der `ipsecX` Schnittstelle definiert werden. Die MTU definiert die maximale mögliche Größe eines IP Paketes. Der Standardwert ist 16260. Der Wert `overridemtu` benötigt üblicherweise keine Anpassung.
- **`syslog`** Hiermit können die Quelle und die Wertigkeit der Syslogmeldungen von FreeS/WAN angegeben werden. So können diese Meldungen in einer anderen Datei protokolliert werden. Die Defaulteinstellung ist `syslog=daemon.error`.
- **`forwardcontrol`** Damit ein FreeS/WAN Gateway die verschlüsselten Pakete weiterreichen kann, muss die Routing (Forwarding) Funktionalität des Kernels aktiviert werden. Dies kann manuell oder durch Firewallskripte erfolgen. FreeS/WAN kann diese Aufgabe aber auch selbst übernehmen. Das wird mit dieser Option aktiviert. Wenn diese Option auf `yes` gesetzt ist, so schaltet FreeS/WAN Forwarding an, wenn es zuvor aus war, und deaktiviert es beim Beenden von FreeS/WAN auch wieder. Defaultwert ist `forwardcontrol=no`. Sinnvoll ist es, diese Funktion von einem Firewallskript genau definiert an- und abschalten zu lassen.

- **dumpdir, dump** Zu Debugzwecken können Linuxprogramme bei einem Absturz ihr Speicherabbild (`dump`) in eine Datei (`core`) schreiben. So können die Programmierer diese Datei untersuchen und feststellen, warum es zum Absturz gekommen ist. Im Fall von Pluto ist dies jedoch sicherheitstechnisch bedenklich, da alle Informationen einschließlich des privaten Schlüssels in dieser Datei vorhanden sind. Pluto schreibt sein Abbild nur, wenn in der Variable `dumpdir` ein Verzeichnis angegeben wurde. Die Direktive dump ist veraltet und erlaubte die Werte `yes` und `no`. Der Dump erfolgte dann immer in `/var/tmp`. Diese Option ist in Version 2 nicht mehr enthalten!
- **prepluto** Hier kann ein Shellkommando angegeben werden, das vor dem Start von Pluto ausgeführt wird. Dies kann zum Beispiel verwendet werden, um vor dem Start die Datei `ipsec.secrets` aus einer verschlüsselten Datei zu erstellen.
- **postpluto** Hier kann ein Shellkommando angegeben werden, das nach dem erfolgreichen Start von Pluto ausgeführt wird. Dies kann zum Beispiel eine mit `prepluto` erzeugte Datei wieder löschen.
- **manualstart** Diese Direktive ähnelt der `plutostart` Direktive. Hier können die Verbindungen aufgezählt werden, die mit manuell definierten Schlüsseln gestartet werden sollen. Die möglichen Werte sind `manualstart=none` (default), ein Name oder eine Liste von Verbindungsnamen. Die Angabe `%search` ist nicht möglich.
- **hidetos, no_eroute_pass, packetdefault, fragicmp, pluto, plutowait, plutobackgroundload** Diese Parameter benötigen üblicherweise keine Modifikation durch den Benutzer. Sie sollen hier nur der Vollständigkeit wegen erwähnt werden. Weitere Informationen bietet die Manpage `ipsec.conf`. Die Parameter `plutobackgroundload` und `no_eroute_pass` wurden in Version 2 entfernt.

Verbindungsparameter
Die Parameter, die bei den Verbindungen angegeben werden können, lassen sich in drei Gruppen einteilen.

Einige Parameter sind nur gültig in manuell verschlüsselten Verbindungen. Hierbei handelt es sich um Verbindungen, bei denen die für die eigentliche Verschlüsselung verwendeten Schlüssel vom Administrator fest definiert werden. Diese werden über die ganze Dauer der Verbindung nicht mehr verändert.

Die zweite Gruppe der Parameter sind nur in automatisch verschlüsselten Verbindungen sinnvoll. Hierbei handelt es sich um Verbindungen, bei denen der Administrator vorgibt, wie die Authentifizierung der Kommunikations-

partner erfolgt. Anschließend ermittelt Pluto mit der Gegenstelle in einem Diffie Hellmann Schlüsselaustausch mit dem IKE Protokoll, die für die Verschlüsselung zu verwendenden Schlüssel. Diese werden anschließend auch in regelmäßigen Abständen erneut ermittelt.

Die dritte Gruppe der Parameter sind sowohl bei manuell verschlüsselten als auch automatisch verschlüsselten Verbindungen erlaubt. Dies sind die allgemeinen Parameter.

TIPP

Verwenden Sie, wenn möglich, immer automatisch verschlüsselte Verbindungen. Sie sind wesentlich sicherer, da die für die eigentliche Verschlüsselung verwendeten Schlüssel automatisch regelmäßig ausgetauscht werden. Nur wenn die Gegenseite die automatische Schlüsselwahl nicht unterstützt, sollte Sie manuell verschlüsselte Verbindungen nutzen.

Die Parameter können auch als Defaultwerte gesetzt werden. Dies erfolgt durch die Angabe der Parameter in einem eigenen Block. Dieser Block trägt dann den Namen `conn %default`. Im Folgenden ist ein Beispiel dargestellt:

```
conn %default
        keyingtries=0
        authby=rsasig
```

Die Werte der hier gesetzten Parameter gelten dann für alle Verbindungen, außer sie werden bei den Verbindungen erneut auf einen anderen Wert gesetzt.

Allgemeine Parameter

Die Angabe der folgenden Parameter ist bei jeder Verbindung erlaubt beziehungsweise erforderlich.

- **`type`** Dieser Parameter legt die Art der Verbindung fest. Mögliche Werte sind `type=tunnel`, `transport` und `pass-through`. Im ersten Fall wird IPsec im Tunnel Modus und im zweiten Fall im Transport Modus betrieben. Der letzte Fall schaltet für diese Verbindung IPsec ab. Die Defaulteinstellung ist `tunnel`. Dies ist in den meisten Fällen auch die richtige Einstellung.
- **`left`** Dies ist eine erforderliche Angabe. Hiermit wird die IP Adresse einer der beiden VPN Endpunkte definiert. Wenn an dieser Stelle `left=%defaultroute` angegeben wird (und gleichzeitig `interfaces=%defaultroute`

definiert ist), so wird die aktuelle IP Adresse des Interfaces genutzt, über die das Standardgateway erreichbar ist. Das ist eine sinnvolle Einstellung für Dialup-Verbindungen oder andere Umgebungen, in denen die IP Adresse bei der Konfiguration von FreeS/WAN nicht bekannt ist. Dieser Parameter überschreibt dann auch den Parameter `leftnexthop` (s.u.).

- **`right`** Siehe `left`. Nur einer der beiden Parameter `left` oder `right` darf den Wert `%defaultroute` enthalten.
- **`leftsubnet`** Hiermit kann ein privates Netzwerk definiert werden, das sich hinter dem VPN Gateway `left` befindet. Die Angabe erfolgt in der Form *IP Adresse/Netzmaske*. Wenn dieser Parameter fehlt, so wird `left/32` angenommen. Die Angabe eines Netzwerkes ist nur sinnvoll im Tunnel Modus.
- **`rightsubnet`** Siehe `leftsubnet`.
- **`leftnexthop`** Dieser Eintrag enthält das Gateway, über das das andere Ende des VPN Tunnels erreicht werden kann. Wenn der Wert `left=%defaultroute` gesetzt ist (s.o.), so wird dieser Eintrag durch das Standardgateway überschrieben. Wenn der Parameter nicht angegeben wird, so wird angenommen, dass das andere Ende direkt erreicht werden kann.
- **`rightnexthop`** Siehe `leftnexthop`
- **`leftupdown`** Ein Skript, das automatisch gestartet wird, wenn der Zustand der Verbindung sich ändert (up/down). Hier werden die Routen und Firewallregeln angepasst. Standardeinstellung ist das Skript `/usr/lib/ipsec/_updown`. Eigene Skripte sollten basierend auf diesem Skript aufgebaut werden (siehe auch `rightupdown`).
- **`rightupdown`** Siehe `leftupdown`
- **`leftfirewall`** Dieser Parameter kann genutzt werden, um FreeS/WAN anzuzeigen, dass zusätzlich Firewallregeln vorhanden sind, die angepasst werden müssen. Dies trifft insbesondere zu, wenn Regeln vorhanden sind, die den Verkehr unterdrücken oder die IP Adressen mit NAT modifizieren würden. Die erlaubten Werte für diesen Parameter sind `yes` und `no` (default). Dieser Parameter `leftfirewall` darf nicht mit `leftupdown` gemeinsam genutzt werden. Wenn er gesetzt wird, modifiziert er die `ipfwadm` Firewallregeln mit dem Default `_updown` FreeS/WAN Skript (siehe auch `rightfirewall`).

ACHTUNG

FreeS/WAN Version unterstützt nur Linux Paketfilter mit dem Befehl `ipfwadm`. Erst FreeS/WAN mit X.509 Patch unterstützt `iptables`. Hier wird aber nicht die Option `left|rightfirewall` genutzt, sondern es muss mit dem Parameter `leftupdown=/usr/lib/ipsec/_updown.x509` das entsprechende Skript gewählt werden! Die Option `left|rightfirewall` ist nur gültig für `ipfwadm` Paketfilter!

- **`rightfirewall`** Siehe `leftfirewall`
- **`also`** Hiermit können Parameter eines anderen Abschnittes an dieser Stelle angehängt werden, als wären sie hier eingegeben worden. Dazu wird mit `also` der Name des entsprechenden Abschnittes angegeben. Die Definition des einzulesenden Abschnittes muss später in der Datei erfolgen. So können, ähnlich einem Unterprogramm, häufig benötigte Definitionen ausgelagert werden.

Die Wahl der symbolischen Namen `left` und `right` für die beiden Tunnelendpunkte ist beliebig. Beim Start ermittelt FreeS/WAN automatisch, welche der beiden Namen auf das eigene Gateway zutrifft.

Manuelle verschlüsselte Verbindungen

Bei manuell verschlüsselten Verbindungen werden die Schlüssel für die Verschlüsselung von den Administratoren festgelegt. Sie werden nicht, wie in einer automatischen Verbindung üblich, von Pluto mit dem Internet Key Exchange (IKE) Protokoll in einem Diffie Hellmann Austausch mit dem Kommunikationspartner ausgetauscht. Alle Funktionen des IKE Protokolls müssen mit entsprechenden Parametern nachgestellt werden. Dadurch entsteht ein erhöhter Administrationsaufwand. Da die manuellen Schlüssel in der Datei `/etc/ipsec.conf` gespeichert werden, ist es erforderlich diese nun vor Einblicken durch normale Benutzer zu schützen.

TIPP

Definieren Sie die manuellen Schlüssel in einer eigenen Datei. Sie können Sie dann mit der `include` Direktive einlesen. So können Sie diese Datei gesondert schützen.

Manuell verschlüsselte Verbindungen sind erforderlich, da einige IPsec Implementierungen anderer Hersteller existieren, die nicht in der Lage sind Diffie Hellmann Schlüsselaustauschverhandlungen durchzuführen.

Die folgenden Parameter stehen zur Verfügung:

- **spi** Die exakte zu verwendende Security Parameter Index (SPI) Nummer für die Verbindung in hexadezimaler Form (0x...). Empfohlen werden Werte zwischen 0x100 und 0xfff, da diese bei FreeS/WAN für manuelle verschlüsselte Verbindungen reserviert wurden. Pluto verwendet für automatisch verschlüsselte Verbindungen Werte größer 0xfff. Wichtig ist, dass jede manuell verschlüsselte Verbindung eine eindeutige Nummer erhält.
- **spibase** Der Anfangswert der zu verwendenen SPI Nummern für die Verbindung in hexadezimaler Form (0x...). Empfohlen werden Werte zwischen 0x100 und 0xff0.
- **esp** Dieser Parameter definiert, dass das Encapsulated Security Payload (ESP)Protokoll und ein bestimmter Verschlüsselungs- und Authentifzierungsmechanismus für die Verbindung zu verwenden ist. Mögliche Werte sind `3des`, `3des-sha1-96` und `3des-md5-96`. Der Algo Patch erweitert diese Liste. Es muss für jede Verbindung mindestens `ah` oder `esp` definiert sein.
- **espenckey** Der zu verwendende Verschlüsselungsschlüssel. Er kann für beide Richtungen getrennt mit `leftespenckey` und `rightespenckey` definiert werden. Für 3DES sollte hier eine 192 Bit lange hexadezimale Zahl angegeben werden. Von dieser Zahl werden lediglich 168 Bit für die Verschlüsselung verwendet. Die restlichen Parititätsbits werden nicht verwendet.
- **espauthkey** Der zu verwendende Authentifizierungsschlüssel. Er kann für beide Richtungen getrennt mit `leftespauthkey` und `rightespauthkey` definiert werden. Für MD5 sollte hier eine 128 Bit lange und für SHA-1 eine 160 Bit lange hexadezimale Zahl angegeben werden.
- **espreplaywindow** Das Protokoll Encapsulated Security Payload (ESP) bietet einen Anti Replay Service. Hierzu wird ein Schiebefenster genutzt (siehe Kapitel 3.2.3, »Anti Replay Schutz«). Die Größe kann zwischen 0 (abgeschaltet, Default) und 64 angegeben werden.
- **ah** Dieser Parameter definiert, dass das Authentication Header (AH) Protokoll und ein bestimmter Authentifizierungsmechanismus für die Verbindung zu verwenden ist. Mögliche Werte sind `hmac-md5-96` und `hmac-sha1-96`. Der Algo Patch erweitert diese Liste.
- **ahkey** Dieser Parameter definiert den Authentifzierungsschlüssel.
- **ahreplaywindow** Das Authentication Header (AH) Protokoll bietet einen Anti Replay Service. Hierzu wird ein Schiebefenster genutzt (siehe Kapitel 3.2.3, »Anti Replay Schutz«). Die Größe kann zwischen 0 (abgeschaltet, Default) und 64 angegeben werden.

Ein Beispiel für eine manuell verschlüsselte Verbindung wird im Folgenden gezeigt.

```
spi=0x200
esp=3des-md5-96
espenckey=0x01234567_89abcdef_02468ace_13579bdf_
12345678_9abcdef0
espauthkey=0x12345678_9abcdef0_2468ace0_13579bdf
```

Listing 5.18 Manuell verschlüsselte Verbindung

Automatische verschlüsselte Verbindungen

Automatisch verschlüsselte Verbindungen sind der Normalfall einer VPN Verbindung. Hierbei werden lediglich die beiden Tunnelendpunkte, die die Verbindung aufbauen und ihre Identitäts- und Authentifizierungsinformationen hinterlegt. Pluto kümmert sich dann um die Authentifizierung und die Aushandlung des zu verwendenden Protokolls, seiner Kryptografie, Algorithmen und Schlüssel. Hierzu wird das Internet Key Exchange (IKE) Protokoll eingesetzt.

Um dies zu konfigurieren, stehen folgende Parameter zur Verfügung.

- **`keyexchange`** Hiermit kann die Methode zum Schlüsselaustausch definiert werden. Der momentan einzige gültige Wert ist `ike` (default).
- **`auto`** Dieser Parameter definiert, wie die Verbindung bei dem Start von FreeS/WAN gehandhabt werden soll. Die Auswertung dieses Parameters erfordert auch, dass die Parameter `plutoload` beziehungsweise `plutostart` gesetzt sind. Erlaubte Werte sind `add`, `route` und `start`. Die Werte `add` und `start` führen dazu, dass die Verbindung geladen beziehungsweise auch sofort gestartet wird. Der Werte `route` lädt die Verbindung und setzt bereits die Route für diese Verbindung ohne jedoch den Tunnel zu starten. Dies führt dazu, dass sämtliche Pakete, die zu dieser Verbindung gehören, verworfen werden, da die Verbindung noch nicht aktiv ist.
- **`auth`** Dieser Wert definiert, ob die Authentifizierung der Pakete im Rahmen des ESP Protokolls (`esp`, default) oder separat mit dem zusätzlichen Protokoll AH (`ah`) erfolgen soll. Verbindungen, bei denen eine Network Address Translation unterwegs durchgeführt wird, können kein AH Protokoll nutzen (siehe 3.2.5, »Authentication Header – AH«). Die FreeS/WAN-Versionen ab Version 2.0 unterstützen diese Option im Moment nicht. Dort ist das AH-Protokoll im Moment nicht korrekt implementiert.
- **`authby`** Dieser Parameter erlaubt die Wahl von entweder Preshared Keys (`secret`, default) oder RSA Schlüsseln (`rsasig`) für die Authentifizierung der Gegenseite. Ab der Version 2 ist `rsasig` der Default-Wert.

- **leftid** Identifikation (ID) des linken Tunnelendpunktes. Dies kann eine IP Adresse, ein DNS Name, eine E-Mail Adresse oder eine beliebige Zeichenkette mit vorangestellten `@` sein. Wenn `leftid` nicht angegeben wird ist der Defaultwert die bei `left` angegebene IP Adresse.
- **rightid** Siehe `leftid`.
- **leftrsasigkey** Dies ist der öffentliche RSA Schlüssel (Public Key) des linken Tunnelendpunktes. Er kann als hexadezimaler Schlüssel (mit Prefix 0x) oder als Base64 kodierter Schlüssel (mit Prefix 0x) angegeben werden. Alternativ sind hier die Werte `%none`, `%dnsondemand` und `%dnsonload` möglich. Mit dem Wert `%dnsonload` lädt Pluto den Schlüssel aus dem `KEY` Eintrag des als `leftid` angegebenen DNS Namens. Bei `%dnsondemand` wird der Schlüssel erst geladen, wenn er von Pluto benötigt wird. Der X.509-Patch erweitert die Liste der möglichen Werte um `%cert`. Die Einstellung `%dnsondemand` ist ab Version 2 der Default-Wert.
- **rightrsasigkey** siehe `leftrsasigkey`.
- **leftrsasigkey2** Dies ermöglicht die Angabe eines alternativen Schlüssels, der ebenfalls anerkannt wird. Hiermit besteht die Möglichkeit einer einfachen Schlüsselmigration.
- **rightrsasigkey2** siehe `leftrsasigkey2`.
- **pfs** Hiermit kann die Perfect Forward Secrecy (PFS) der Sitzungsschlüssel aktiviert werden. Damit ist es nicht möglich, ausgehend von einem kompromittierten Sitzungsschlüssel, auf die vorher ausgehandelten Sitzungsschlüssel zurückzuschließen (siehe Kapitel »Quick Modus«). Der Defaultwert ist `yes`.
- **keylife** Hiermit kann die Lebensdauer der Sitzungschlüssel angegeben werden. Vor Ablauf dieser Zeit werden die Schlüssel im Regelfall neu ausgehandelt um die dauerhafte Verschlüsselung zu garantieren. Die Angabe kann in ganzen Sekunden (`s`) oder in Minuten (`m`), Stunden (`h`) oder Tagen (`d`) erfolgen. Der Defaultwert ist `8.0h`. Erlaubt sind Werte von `1s` bis `24.0h`. Dieser Wert erfordert teilweise Anpassungen, wenn FreeS/WAN mit anderen IPsec Implementierungen zusammenarbeitet.
- **rekey** Dieser Parameter definiert, ob Pluto vor Ablauf der `keylife` die Neuverhandlung der Schlüssel initiieren soll (`yes`, default) oder nicht (`no`).
- **rekeymargin** Dieser Wert gibt an, wie früh vor Ablauf der `keylife` Pluto mit der Neuverhandlung der Sitzungsschlüssel beginnen soll. Er akzeptiert die gleichen Werte wie `keylife`. Der Defaultwert ist `9m`.
- **rekeyfuzz** Dieser Prozentsatz definiert, wie stark der gewählte `rekeymargin` Wert vom eingestellten abweichen darf. Damit werden die Rekey

Intervalle zufälliger gewählt. Erlaubte Werte gehen von `0%` bis `100%` (default).

- **`keyingtries`** Dieser Wert definiert, wieviele Versuche für den Aufbau oder den Wiederaufbau einer Verbindung unternommen werden sollen. Der Wert `0` steht für unendlich viele Versuche (Default: `3`). Ab Version 2 ist hier der Wert `%forever` die Default-Einstellung.
- **`ikelifetime`** Hiermit kann die Lebensdauer der ISAKMP Security Association definiert werden. Der Defaultwert beträgt `1h`.
- **`compress`** Das IPsec Protokoll fügt zusätzlichen Overhead zu den übertragenen Informationen hinzu. Dies reduziert die effektiv zur Verfügung stehende Bandbreite. Aus diesem Grund bietet das IPsec Protokoll die IPcomp Kompression. Dieser Parameter erlaubt die Einstellung, ob das FreeS/WAN Gateway diese Kompression unterstützen soll. Nur wenn beide Seiten die Kompression unterstützen, wird sie auch genutzt. Der Defaultwert ist `no`.
- **`disablearrivalcheck`** Dieser Parameter aktiviert (default) einen Sicherheitscheck, bei dem von KLIPS jedes Paket, das den Tunnel verlässt, daraufhin geprüft wird ob es plausible IP Adressen trägt. Ab Version 2 ist dieser Parameter auf den Default-Wert `no` gesetzt.

Üblicherweise werden ein Großteil der aufgeführten Parameter nicht gesetzt, sondern die vordefinierten Standardwerte genutzt. In wenigen Fällen, bei denen FreeS/WAN einen Tunnel zu anderen IPsec Implementierungen aufbaut, ist es aber erforderlich die Standardwerte zu modifizieren.

Im Folgenden wird eine typische automatisch verschlüsselte Verbindung gezeigt:

```
conn newyork-berlin
        keyingtries=0
        authby=rsasig
        left=3.0.0.1
        leftnetwork=192.168.100.0/24
        leftid=vpn@newyork
        leftrsasigkey=0sAQOSdv....
        rightrsasigkey=0sAQNlGO....
        right=5.0.0.1
        rightnetwork=192.168.210.0/24
        rightid=vpn@berlin
        auto=start
```

Listing 5.19 Automatisch verschlüsselte Verbindung

5.5.2 Das Kommandozeilenwerkzeug ipsec

FreeS/WAN verwendet nur ein einziges Kommandozeilenwerkzeug, `ipsec`, das als gemeinsames Frontend für eine große Anzahl von Kommandomodulen dient. Die Manpage des Werkzeuges ist daher auf viele verschiedene Manpages aufgeteilt worden: `ipsec`, `ipsec_setup`, `ipsec_manual`, `ipsec_auto`, `ipsec_barf` und so weiter. Die Manpage eines Kommandomoduls erhält man, indem der Name des Kommandomoduls an den Befehl `ipsec` mit einem Unterstrich gehängt wird.

Im Folgenden sollen alle Kommandomodule besprochen werden. Hierbei werden die wichtigen, häufig verwendeten Module intensiver besprochen.

Kommandomodul: auto

Dieses Modul steuert die automatisch verschlüsselten Verbindungen. Mit ihm können die Verbindungen gestartet, gestoppt und betrachtet werden. Normalerweise können sämtliche benötigten Funktionen über dieses Modul gesteuert werden. Alle weiteren Module sind nur in besondereren Fällen und zur Fehlersuche erforderlich.

Das Kommandomodul `auto` kennt die folgenden Befehle:

- **`--add`** Hiermit wird eine neue Verbindung aus der Konfigurationsdatei in die Pluto Datenbank gelesen. Dies ist nur erfolgreich, wenn zuvor noch keine Verbindung mit dem entsprechenden Namen existierte. Anschließend steht diese Verbindungsdefinition für einen Aufbau zur Verfügung. Dieser Aufbau kann auf dem selben Rechner oder vom anderen Tunnelendpunkt initiiert werden.

```
# ipsec auto --add --verbose left-right
002 added connection description "left-right"
# ipsec auto --add --verbose left-right
020 attempt to redefine connection "left-right"
```

- **`--delete`** Diese Option erlaubt das Entfernen einer Verbindungsdefinition aus der Pluto Verbindungsdatenbank. Wenn entsprechende Verbindungen momentan aktiv sind, werden sie ebenfalls beendet. Existierte keine Verbindung mit dem entsprechenden Namen, wird eine Fehlermeldung ausgegeben.

```
# ipsec auto --delete --verbose left-right
002 "left-right": deleting connection
# ipsec auto --delete --verbose left-right
021 no connection named "left-right"
```

- **-replace** Um FreeS/WAN mitzuteilen, dass sich die Konfiguration einer Verbindung geändert hat, ist es erforderlich, deren Definition zuerst zu entfernen (`-delete`) und anschließend wieder hinzuzufügen (`-add`). Dies wird von `-replace` in einem Arbeitsschritt durchgeführt.

```
# ipsec auto --replace --verbose left-right
002 "left-right": deleting connection
002 added connection description "left-right"
```

- **-up** Mit diesem Kommando wird eine Verbindung gestartet. Pluto versucht nun basierend auf der geladenen Konfiguration in der internen Datenbank die entsprechende Verbindung aufzubauen.

```
# ipsec auto --up --verbose left-right
002 "left-right" #1: initiating Main Mode
104 "left-right" #1: STATE_MAIN_I1: initiate
106 "left-right" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "left-right" #1: STATE_MAIN_I3: sent MI3, expecting MR3
002 "left-right" #1: Peer ID is ID_FQDN: '@right'
002 "left-right" #1: ISAKMP SA established
004 "left-right" #1: STATE_MAIN_I4: ISAKMP SA established
002 "left-right" #2: initiating Quick Mode
RSASIG+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK
112 "left-right" #2: STATE_QUICK_I1: initiate
002 "left-right" #2: sent QI2, IPsec SA established
004 "left-right" #2: STATE_QUICK_I2: sent QI2, IPsec SA established
```

- **-down** Hiermit kann eine vorher aufgebaute Verbindung beendet werden. Dabei sendet FreeS/WAN auch eine Delete-SA Meldung an den Tunnelpartner. Handelt es sich beim Tunnelpartner auch um FreeS/WAN so kommt es zu Problemen, denn FreeS/WAN reagiert nicht auf diese Meldung und entfernt die Verbindung. Dies ist nur mit dem Delete/Notify SA Patch der Fall.

```
# ipsec auto --down --verbose left-right
002 "left-right": terminating SAs using this connection
002 "left-right": #2: deleting state (STATE_QUICK_I2)
002 "left-right": #1: deleting state (STATE_MAIN_I4)
```

- **-route** Die Route für den Tunnel wird normalerweise automatisch beim Start des Tunnels mit `-up` eingerichtet. Wenn jedoch nur die Route ohne einen Start der Verbindung eingerichtet werden soll, so kann das mit diesem Kommando geschehen. Das führt dazu, dass Pakete, die an den anderen Tunnelendpunkt gerichtet sind, nicht transportiert werden, da die Verbindung noch nicht existiert. Sie werden verworfen.

```
# ipsec auto --route --verbose left-right
```

- **-unroute** Dieser Befehl entfernt die Route wieder. Normalerweise bleibt die Route eingerichtet, wenn der Befehl `-down` ausgeführt wird oder eine automatische Neuverhandlung der Verbindung fehlschlägt. So ist sichergestellt, dass keine Pakete in Klartext übertragen werden.
- **-rereadsecrets, -ready** Diese Befehle lesen die Datei `/etc/ipsec.secrets` erneut ein. Damit ist es möglich ohne eine Unterbrechung der Tunnel die Schlüssel auszutauschen.

```
# ipsec auto --rereadsecrets --verbose
002 forgetting secrets
002 loading secrets from "/etc/ipsec.secrets"
```

- **-status** Dieser Befehl gibt den aktuellen Status aller Verbindungen aus.

```
# ipsec auto --status --verbose
000 interface ipsec0/eth0 192.168.1.2
000
000 "left-right": 192.168.1.2[@left]---192.168.1.1...192.168.2.1
---192.168.2.2[@right]
000 "left-right":   ike_life: 3600s; ipsec_life: 28800s; rekey_margin:
540s; rekey_fuzz: 100%; keyingtries: 0
000 "left-right":   policy:
RSASIG+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK; interface: eth0; erouted
000 "left-right":   newest ISAKMP SA: #1; newest IPsec SA: #2; eroute
owner: #2
000
000 #2: "left-right" STATE_QUICK_I2 (sent QI2, IPsec SA established);
EVENT_SA_REPLACE in 27991s; newest IPSEC; eroute owner
000 #2: "left-right" esp.3db65bff@192.168.2.2 esp.a3c1aeec@192.168.1.2
tun.1002@192.168.2.2 tun.1001@192.168.1.2
000 #1: "left-right" STATE_MAIN_I4 (ISAKMP SA established);
EVENT_SA_REPLACE in
2535s; newest ISAKMP
000
```

- **-show** Das Kommandomodul `auto` ist lediglich ein Frontend für die einfache Ausführung komplizierterer `ipsec` Kommandos. Um die Ausführung dieser Kommandos angezeigt zu bekommen, kann diese Option genutzt werden.

```
# ipsec auto --down --show left-right
+ ipsec whack --name left-right --terminate
+ echo = 0
```

- **-showonly** Diese Funktion führt dazu, dass die auszuführenden Kommandos lediglich angezeigt, aber nicht ausgeführt werden.

```
# ipsec auto --up --showonly left-right
ipsec whack --name left-right --initiate
```

- **`--config` Nicht implementiert!** Hiermit soll es möglich sein, eine andere Konfigurationsdatei als `/etc/ipsec.conf` anzugeben.
- **`--verbose`** Diese Funktion gibt wesentlich mehr Informationen aus. Beispiele wurden bei den anderen Parametern gezeigt.
- **`--asynchronous`** Diese Option ist nur erlaubt bei gleichzeitiger Verwendung mit `--up`. Wenn sie gesetzt ist, wird die Verbindung gestartet, aber die Abarbeitung weiterer Befehle nicht verzögert, bis die Verbindung aufgebaut wurde.

```
# ipsec auto --up --asynchronous left-right
```

Kommandomodul: manual

Dieses Modul steuert die manuell verschlüsselten Verbindungen. Mit ihm können die Verbindungen gestartet, gestoppt und betrachtet werden. Normalerweise werden sämtliche benötigten Funktionen für manuell verschlüsselte Verbindungen über dieses Modul gesteuert. Grundsätzlich sind jedoch manuell verschlüsselte Verbindungen als nicht so sicher wie automatisch verschlüsselte Verbindungen einzustufen.

Das Modul `manual` bietet wesentlich weniger Funktionen als `auto` und sollte nur genutzt werden, wenn es keine andere Möglichkeit gibt. So unterstützt `manual` zum Beispiel auch keine IP Kompression.

Das Kommandomodul `manual` kennt ähnlich dem Kommandomodul `auto` die folgenden Befehle:

- **`--up`** Dieser Befehl startet die Verbindung und setzt auch die Route.
- **`--down`** Dieser Befehl beendet die Verbindung und entfernt *nicht* die Route.
- **`--route`** Dieser Befehl setzt lediglich die Route. In Frage kommende Pakete werden nun so lange verworfen, bis die Verbindung aufgebaut wird.
- **`--unroute`** Dieser Befehl ist in der Lage eine Route zu entfernen, die bei `--down` zurückbleibt.
- **`--union`** Dieser Befehl erlaubt das Zusammensetzen der Verbindung aus mehreren Teildefinitionen. Er wurde vor der Verfügbarkeit von `also` genutzt, um die Definition von manuellen Verbindungen zu vereinfachen.
- **`--show`** Hiermit ist es möglich, sich sämtliche Befehle, die vom Kommandomodul `manual` ausgeführt werden, während dessen zu betrachten.

- **-showonly** Diese Option zeigt lediglich die auszuführenden Befehle an, aber führt sie nicht aus.
- **-other** Zur Fehlersuche erlaubt diese Option die Emulation der entsprechenden Gegenseite.
- **-iam** Hiermit kann eine spezifische IP Adresse und Netzwerkkarte für den Aufbau der manuellen Verbindung angegeben werden. Die Notation ist *Adresse@Netzwerkkarte*.
- **-config** Diese Option ermöglicht die Angabe einer alternativen Konfigurationsdatei. Sie ist in dem `manual` Kommandomodul im Gegensatz zum `auto` Kommandomodul implementiert.

Kommandomodul: setup

Dieses Kommandomodul kümmert sich um den Start und den Stop des FreeS/WAN Systems. Es schließt den KLIPS Kernel Code und den Pluto IKE Daemon ein. In vielen Distributionen wird dieses Skript auch verwendet, um beim Systemstart über die System-V rc-Skripte FreeS/WAN zu starten. Um diese Funktion zu unterstützen, bietet dieses Modul die folgenden Optionen:

- **start** Startet sowohl KLIPS als auch Pluto. Wenn KLIPS modular in den Kernel implementiert wurde, so wird auch das entsprechende Modul geladen. Durch den Start von Pluto werden automatisch die Verbindungen gestartet, die durch `plutoload` und `plutostart` in der Datei `/etc/ipsec.conf` definiert werden. Die Optionen `plutoload` und `plutostart` können ab Version 2.0 nicht mehr gesetzt werden und sind per Default aktiv.
- **stop** Hiermit wird KLIPS als auch Pluto beendet. Das führt auch zum Ende aller aufgebauten Verbindungen. Wenn KLIPS modular im Kernel implementiert wurde, so wird auch das entsprechende Modul entladen.
- **restart** Diese Funktion führt `stop` und `start` nacheinander aus.
- **status** Hiermit kann die Funktion von KLIPS und Pluto überprüft werden.

TIPP

Bei modernen Distributionen wird unter Umständen beim Start von FreeS/WAN die Warnung ausgegeben, dass "route filtering" aktiviert sei. Hierbei handelt es sich um die Kernel Variable `net.ipv4.conf.<interface>.rp_filter`. Dieser Wert sollte für das entsprechende physikalische Interface auf Null gesetzt werden. Ab Version 2.0 deaktiviert FreeS/WAN selbstständig den `rp_filter`.

```
# sysctl -w net.ipv4.conf.<interface>.rp_filter=0
```

Red Hat Linux bietet zu diesem Zweck die Datei `/etc/sysctl.conf` an. Dort können diese Werte eingetragen werden. Die Datei wird bei jedem Systemstart abgearbeitet.

Bei Red Hat Linux kann FreeS/WAN auch mit `service ipsec start` gestartet werden. Dieses Skript ruft entsprechend den Befehl `ipsec setup start` auf. SuSE Linux verwendet hierzu den Befehl `rcipsec start`.

Kommandomodul: look

Dieses Kommando gibt einen knappen Überblick über die aufgebauten Verbindungen und ihre Routen. Die Ausgabe dieses Befehls ist nicht sicherheitsrelevant.

```
# ipsec look
left.spenneberg.de Tue Feb  4 03:57:55 CET 2003
192.168.1.2/32     -> 192.168.2.2/32     => tun0x1002@192.168.2.2
esp0x3db65c02@192.168.2.2  (0)
ipsec0->eth0 mtu=16260(1500)->1500
esp0x3db65c02@192.168.2.2 ESP_3DES_HMAC_MD5: dir=out src=192.168.1.2
iv_bits=64bits iv=0xe1c3ad3ef7b58bdc ooowin=64 alen=128 aklen=128
eklen=192 life(c,s,h)=addtime(55,0,0)
esp0xce65753a@192.168.1.2 ESP_3DES_HMAC_MD5: dir=in  src=192.168.2.2
iv_bits=64bits iv=0x6502032e6c426f28 ooowin=64 alen=128 aklen=128
eklen=192 life(c,s,h)=addtime(55,0,0)
tun0x1001@192.168.1.2 IPIP: dir=in  src=192.168.2.2
life(c,s,h)=addtime(55,0,0)
tun0x1002@192.168.2.2 IPIP: dir=out src=192.168.1.2
life(c,s,h)=addtime(55,0,0)
Destination     Gateway         Genmask         Flags   MSS Window  irtt
Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG       40 0          0
eth0
192.168.1.0     0.0.0.0         255.255.255.0   U        40 0          0
eth0
192.168.1.0     0.0.0.0         255.255.255.0   U        40 0          0
ipsec0
192.168.2.2     192.168.1.1     255.255.255.255 UGH      40 0          0
ipsec0
```

Diese Ausgabe gibt einen schnellen Überblick.

Kommandomodul: barf

Für Debugging und in erster Linie zur Ferndiagnose besser geeignet ist die Ausgabe des Kommandomoduls `barf`. Es bereitet sämtliche Dateien und Protokolleinträge im Zusammenhang mit FreeS/WAN auf und gibt sie auf der Standardausgabe aus. Diese Informationen können dann in eine Datei umgelenkt werden und zur Ferndiagnose zum Beispiel per E-Mail versandt werden. Die erzeugte Datei enthält alle Informationen, die für die Analyse von FreeS/WAN erforderlich sind. Alle sicherheitskritischen Informationen werden durch Prüfsummen ersetzt, so dass eine Zuordnung möglich, aber ein Einblick verwehrt bleibt.

Kommandomodul: eroute

Dieses Modul erlaubt es, die IPsec Routingtabellen zu verwalten. Dies schließt die Erzeugung und Löschung von Routen wie die Betrachtung ein. Die meisten Funktionen dieses Moduls werden automatisch von den entsprechenden Befehlen der Module `auto`, `manual` und `whack` genutzt. Dennoch ist eine gewisse Kenntnis der Befehle bei der Fehlersuche hilfreich.

Wenn das Kommandomodul ohne weitere Option aufgerufen wird, so zeigt es den Inhalt der Datei `/proc/net/ipsec_eroute` an. Hier befinden sich die aktuellen IPsec Routen:

```
# ipsec eroute
0        192.168.1.2/32     -> 192.168.2.2/32    => tun0x1002@192.168.2.2
```

So kann leicht erkannt werden, welche Verbindungen momentan aktiv sind. Dabei ist jedoch zu beachten, dass die Routen von mit `–down` beendeten Verbindungen weiterhin bestehen bleiben.

```
# ipsec auto --down left-right
# ipsec eroute
0        192.168.1.2/32     -> 192.168.2.2/32    => %trap
```

Zusätzlich unterstützt der Befehl `ipsec eroute` weitere Optionen. Sie sollen hier nur kurz vorgestellt werden, da deren Ausführung üblicherweise automatisch erfolgt. Weitere Informationen finden sich in der Manpage `ipsec_eroute(8)`.

- **`-add`** Hiermit ist es möglich eine Route zur IPsec Routentabelle hinzuzufügen.
- **`-del`** Hiermit ist es möglich eine Route zu entfernen.
- **`-replace`** Diese Option tauscht eine Route aus.
- **`-clear`** Diese Option löscht die gesamte Routentabelle

Kommandomodul: ranbits

Dieses Modul erzeugt Zufallszahlen in ASCII Format. Diese Zufallszahlen sind sehr gut geeignet als PSK oder als Authentifizierungs- und Verschlüsselungsschlüssel für manuelle Verbindungen.

Das Modul ist in der Lage zwei verschiedene Quellen für die Erzeugung der Zufallszahl zu verwenden. Üblicherweise wird das Gerät `/dev/random` genutzt. Dies erzeugt sehr hochwertige Zufallszahlen. Zur Beschleunigung der Erzeugung kann jedoch mit der Option `-quick` der Pseudozufallszahlengenerator `/dev/urandom` genutzt werden.

Bei jedem Aufruf muss die Anzahl der zu ermittelnden zufälligen Bits angegeben werden.

```
# ipsec ranbits 128
0xd46774ea_5cd0b3d2_0594af64_8ba52fa9
```

Die zusätzliche Angabe von `-continuous` unterdrückt die Ausgabe des Unterstrichs alle 32 Bit.

Kommandomodul: rsasigkey

Mit diesem Modul können RSA Schlüsselpaare erzeugt werden. Hierzu ist es in erster Linie erforderlich, die Schlüssellänge in Bits anzugeben:

```
$ /usr/sbin/ipsec rsasigkey 2192
# RSA 2192 bits   kermit.spenneberg.de   Tue Feb  4 14:26:49 2003
# for signatures only, UNSAFE FOR ENCRYPTION
#pubkey=0sAQN0L...
#IN KEY 0x4200 4 1 AQN0LC...
# (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
Modulus: 0x742c...
PublicExponent: 0x03
# everything after this point is secret
PrivateExponent: 0x135c...
Prime1: 0xe7fcc41K...
Prime2: 0x80328b09...
Exponent1: 0x9aa882b84...
Exponent2: 0x5577075b7...
Coefficient: 0x15fcba42e...
```

Zusätzliche Optionen erlauben die Modifikation des Hostnamens in der ersten Zeile (fett, `-hostname`), die Angabe der Zufallszahlenquelle (`-random`) und die Angabe der zu verwendenden Runden (`-rounds`, default 30).

Die erzeugten Schlüssel weisen immer einen öffentlichen Exponenten von 3 auf. Diese Schlüssel sind lediglich zur Signatur gedacht und bieten keine sichere Verschlüsselung!

Kommandomodul: verify

Dies ist ein sehr wichtiges Modul für die Fehlersuche bei FreeS/WAN. Es testet eine Vielzahl von Bedingungen und meldet, ob diese erfüllt sind. Dieses Modul sollte als erstes aufgerufen werden, wenn eine FreeS/WAN Installation nicht funktioniert.

```
# ipsec verify
Checking your system to see if IPsec got installed and started correctly
Version check and ipsec on-path                                 [OK]
Checking for KLIPS support in kernel                            [OK]
Checking for RSA private key (/etc/ipsec.secrets)               [OK]
Checking that pluto is running                                  [OK]
< gekürzt >
```

Kommandomodul: showhostkey

Um eine Authentifizierung mit RSA Schlüsseln durchzuführen, benötigen die beiden Tunnelendpunkte jeweils den öffentlichen Schlüssel des Partners. Dies kann durch einen entsprechenden Eintrag in der Datei `ipsec.conf` geschehen (`leftrsasigkey`) oder durch einen KEY Record eines DNS Servers.

Diese Einträge benötigen eine spezielle Syntax. Ein Schlüssel in dieser Syntax kann sehr einfach mit dem `ipsec showhostkey` Befehl extrahiert werden.

```
# ipsec showhostkey
; RSA 2048 bits   left.spenneberg.de   Tue Feb  4 00:25:22 2003
left.spenneberg.de.     IN      KEY     0x4200 4 1
AQN2mRSuMk/ts8wP3c5b2xJS1uefK7OXl1z1SsT4xTC/jPjVv5Q+k
nO+wqxGcWS5dcns2sSnVbhza15cgQ2N8WbRxhvgGwscDYoqwvReLUumx
mL26LLr47Wg3/5gsM1jnUS9FJ1Wn7N0WJOnHS1U7Z4VreRX28Gid6FzgpS
RntDI11HhiAIA33eDaLjD2+IQhGcv/O5EE6ys2nCLBmW1Yo+5F7Hx5MKag0
Gz1Vsh+7d81MUjTr0Q0JMywgTg3YQ7sezOl5qolE9AmLbDsYo2AW0am1K
X4TK5DdRTCTK1jcsFP0ndgBbTw3oaIkxruw9e8t/CzsS42M+4w7cBA0WxOEJp
```

Normalerweise erzeugt der Befehl den Schlüssel in einem Format, das es erlaubt ihn direkt in einer DNS Zonendatei zu verwenden. Mit den Optionen `-left` und `-right` besteht die Möglichkeit den Schlüssel als `leftrsasigkey` für die Datei `ipsec.conf` zu formatieren. Mit `-id` kann hierbei der Schlüssel einer anderen Identität ausgewählt werden. Ansonsten wird der Defaultschlüssel genutzt.

Mit der Option `-txt` kann ein Format generiert werden, das für die opportunistische Verschlüsselung benötigt wird.

Kommandomodul: spi

Dieses Modul ermöglicht es die IPsec Security Associations zu verwalten. So besteht die Möglichkeit IPsec SAs zu erzeugen und zu entfernen. Dies sind jedoch interne Funktionen, die üblicherweise automatisch von den Frontends `ipsec auto` und `ipsec whack` wahrgenommen werden. Eine wichtige Funktion ist jedoch die Anzeige der vorhandenen SAs. Das ist möglich mit dem Befehl `ipsec spi`. Dieser Befehl zeigt den Inhalt der Datei `/proc/net/spi` an.

```
# ipsec spi
tun0x1002@192.168.2.2 IPIP: dir=out src=192.168.1.2
life(c,s,h)=addtime(138,0,0)tun0x1001@192.168.1.2 IPIP: dir=in
src=192.168.2.2 life(c,s,h)=addtime(138,0,0)esp0xa775b4f4@192.168.2.2
ESP_3DES_HMAC_MD5: dir=out src=192.168.1.2 iv_bits=64bits
iv=0x69ce503a2cd63b79 ooowin=64 alen=128 aklen=128 eklen=192
life(c,s,h)=addtime(138,0,0)
esp0x2e90b054@192.168.1.2 ESP_3DES_HMAC_MD5: dir=in  src=192.168.2.2
iv_bits=64bits iv=0xf5650d90d95486ae ooowin=64 alen=128 aklen=128
eklen=192 life(c,s,h)=addtime(138,0,0)
```

Hier können nun die Security Associations, ihre Lebensdauer und die verwendeten Algorithmen überprüft und überwacht werden.

Die weiteren Funktionen des Kommandos `ipsec spi` können in der Manpage nachgelesen werden.

Kommandomodul: spigrp

Damit mehrere Security Asscociations auf eine Verbindung angewendet werden können (zum Beispiel tun, esp und ah SAs), ist es erforderlich sie zu gruppieren. Ein Eintrag in der IPsec Routentabelle kann nur auf eine SA zeigen. Hierbei handelt es sich ebenso, wie beim `ipsec spi` Befehl um einen internen Befehl, der üblichweise im Hintergrund durch `ipsec auto` oder `ipsec whack` aufgerufen wird.

Interessant für die Fehlersuche, besonders bei heterogenen VPN Lösungen, ist die Anzeige der vorhandenen Gruppen.

```
# ipsec spigrp
tun0x1002@192.168.2.2 esp0xa775b4f4@192.168.2.2
tun0x1001@192.168.1.2 esp0x2e90b054@192.168.1.2
```

Der Befehl liest diese Informationen aus der Datei `/proc/net/spigrp`

Kommandomodul: tncfg

Der Befehl `ipsec tncfg` bindet eine virtuelle `ipsecX` Netzwerkkarte an die physikalische Netzwerkkarte. Hierbei besteht die Möglichkeit, die momentane Bindung zu betrachten, eine neue Bindung zu erzeugen (`-attach`) oder eine Bindung zu lösen (`-detach`).

Üblicherweise wird dieser Befehl von `ipsec setup` entsprechend der Konfigurationsdatei `ipsec.conf` aufgerufen. In der Konfigurationsdatei bestimmt die Zeile `interfaces=`, welche virtuelle Netzwerkkarte wie gebunden werden soll.

Eine Anzeige der aktuellen Bindungen erfolgt mit `ipsec tncfg`.

```
# ipsec tncfg
ipsec0 -> eth0 mtu=16260(1500) -> 1500
ipsec1 -> NULL mtu=0(0) -> 0
ipsec2 -> NULL mtu=0(0) -> 0
ipsec3 -> NULL mtu=0(0) -> 0
```

Insgesamt stehen ohne Änderungen am Quelltext nur vier virtuelle Netzwerkkarten zur Verfügung.

Kommandomodul: pluto

Pluto ist der FreeS/WAN IKE Daemon. Er ist zuständig für den Internet Key Exchange. Dieses Kommando startet Pluto. Anfragen können dann mit dem `ipsec whack` Befehl gestellt werden. Dieser Befehl wird weiter unten behandelt.

Pluto verhandelt sowohl die ISAKMP SA der Phase 1 als auch die IPsec SAs der Phase 2. Pluto implementiert in der Phase 1 nur den sogenannten Main Modus. Der Aggressive Modus wird nur mit einem entsprechenden Patch unterstützt. Die Phase 2 wird im sogenannten Quick Modus durchgeführt.

Die Richtlinien für die Annahme oder Ablehnung von Security Associations sind momentan nicht modifizierbar. Diese Richtlinien befinden sich nicht in einer Datenbank sondern hardkodiert in Pluto.

Zur Authentifizierung kann Pluto PSKs und RSA Schlüssel verwenden. Mit einem zusätzlichen Patch können auch X.509 Zertifikate genutzt und NAT Traversal unterstützt werden.

Der Aufruf von Pluto erfolgt normalerweise vollkommen automatisch durch `ipsec setup`. Das Kommando `ipsec pluto` ist daher nur zu verwenden, wenn das Kommando zum Start von FreeS/WAN umgangen werden soll.

Kommandomodul: whack

Der Befehl `ipsec whack` dient für die interne Steuerung und die Kommunikation mit Pluto. Dieser Befehl wird verwendet um Verbindungen zu laden, zu starten und zu beenden. Dabei können sämtliche Parameter, die normalerweise in der Datei `/etc/ipsec.conf` definiert werden, auf der Kommandozeile angegeben werden. Normalerweise wird dieser Befehl durch den Befehl `ipsec auto` mit den entsprechenden Parametern aufgerufen.

Der interne Befehl bietet unter anderem Optionen zur Verwaltung von Verbindungen und Routen. Diese Verwaltung erfolgt jedoch komfortabler mit den entsprechenden Frontends. Weitere Erläuterung der Optionen bietet die sehr ausführliche Manpage (`ipsec_whack(8)`).

Jedoch existiert eine sehr sinnvolle Option, mit der im laufenden Betrieb einer Verbindung deren Debugstatus verändert werden kann. So stellt `ipsec whack –name <verbindung> –debug-<all|none|raw| crypt|parsing| emitting|control|klips|dns|private>` für jede Verbindung den Status ein, ohne den Tunnel erneut starten zu müssen.

Genauso besteht die Möglichkeit ohne Neustart die verwendeten Schlüssel anzuzeigen beziehungweise sie neu einzulesen.

```
# ipsec whack --listpubkeys
# ipsec whack --rereadsecrets
```

Wenn der X.509-Patch verwendet wird, so existieren auch die folgenden Funktionen: `–listcerts`, `–listcacerts`, `–listcrls`, `–listall`, `–rereadmycert`, `–rereadcacerts`, `–rereadcrls` und `–rereadall`. Handelt es sich um die Version 1.4.x des Patches mit Smartcardunterstützung kommt weiterhin die folgende Option hinzu: `–listcards`.

Kommandomodul: pf_key

Dieses interne Kommando wird von FreeS/WAN genutzt, um die Schlüsselanfragen des Kernels abzufangen und zu beantworten. Hierzu öffnet das Kommando einen `PF_KEY` Socket und leitet alle hierüber empfangenden Anfragen weiter. Der Socket unterstützt Schlüsselanfragen für AH, ESP, Tunnel und IPcomp-Kompressions SAs.

Ein Aufruf dieses Befehls durch den Benutzer ist nie erforderlich.

Kommandomodul: ikeping

Das IKE Protokoll von FreeS/WAN unterstützt Ping-ähnliche Echo Pakete. Sie dienen zur Diagnose von Netzwerkproblemen. Die Implementierung

dieser Echo Pakete wird im Internet Draft von M. Richardson (*http://www.sandelman.ottawa.on.ca/SSW/ietf/ipsec/ikeping/ipsec-ikeping.txt*) beschrieben. Dieser Draft wurde am 22. Feburar 2002 veröffentlicht und ist am 23. August 2002 nach sechs Monaten gelöscht worden. Das ISAKMP/IKE Echo Protokoll ist nicht zum Standard erhoben worden. Daher wird die Unterstützung in vielen anderen IPsec Implementierungen fehlen.

Dennoch kann das Kommando `ipsec ikeping` in reinen FreeS/WAN Umgebungen sehr hilfreich bei der Fehlersuche sein.

Dazu übermittelt `ipsec ikeping` an alle angegebenen Adressen eine ISAKMP Nachricht vom Typ 244. Als Antwort erwartet es eine ISAKMP Nachricht vom Typ 245. So kann dieser Befehl die Verfügbarkeit und Funktion des IKE Daemons am anderen Tunnelendpunkt testen.

Kommandomodul: calcgoo

Dies ist ein rein internes Kommando, mit dem das für den aktuellen Kernel passende KLIPS Modul ermittelt wird. So wird sichergestellt, dass nur dann die Module geladen werden, wenn sie zum Kernel passen.

Kommandomodul: send-pr

Dieses Kommandomodul dient der Versendung von Fehlerberichten an ein GNU Problem Report Management System (GNATS, *http://www.gnu.org/software/gnats/*). Hierbei erzeugt der Befehl ein Template einer E-Mail und startet anschließend einen Editor, so dass der Benutzer dieser E-Mail sein Problem hinzufügen kann.

Damit ein Problembericht an das GNATS System gesendet wird, muss der sendende Benutzer zunächst eine GNATS-ID anfordern. Dies erfolgt mit dem Befehl `ipsec send-pr -request-id`.

Die Problemberichte sollten bei ihrer Versendung einer Kategorie zugeordnet werden. Die möglichen Kategorien sind: *pluto*, *klips*, *startup*, *doc*, *interop*, *source*, *admin*.

Diese Funktion kann genutzt werden, um Fehler an das FreeS/WAN Team zu melden. Jedoch kann dies auch für den Aufbau eines eigenen Support-Teams in größeren Unternehmen genutzt werden. Hierzu ist die Installation von GNU GNATS und die Anpassung der Datei `/usr/lib/ipsec/ipsec_pr.template` erforderlich. Diese Datei enthält die Einstellungen, wie und an wen der Problembericht zu senden ist.

Es ist aber zweifelhaft, ob irgend jemand diesen Befehl tatsächlich einsetzt.

Kommandomodul: showdefaults

Dieser Befehl zeigt die Werte an, die bei der Angabe von `%defaultroute` in der Datei `/etc/ipsec.conf` verwendet werden. Dies kann ebenfalls recht gut zur Fehlersuche genutzt werden. So besteht die Möglichkeit die dort angezeigten Informationen mit den gewünschten Werten zu vergleichen.

```
# ipsec showdefaults
routephys=eth0
routephys=eth0
routevirt=ipsec0
routevirt=ipsec0
routeaddr=192.168.2.2
routeaddr=192.168.2.2
routenexthop=192.168.2.1
routenexthop=192.168.2.1
defaultroutephys=eth0
defaultroutevirt=ipsec0
defaultrouteadd=192.168.2.2
defaultroutenexthop=192.168.2.1
```

Kommandomodul: newhostkey

Mit diesem Kommando kann sehr einfach ein neues RSA Schlüsselpaar erzeugt werden. Hierbei erzeugt dieser Befehl, im Gegensatz zum Befehl `ipsec rsasigkey`, die Schlüssel direkt in einem Format, das in der Datei `/etc/ipsec.secrets` erwartet wird. Um die Schlüssel zu erzeugen verwendet der Befehl den `ipsec rsasigkey –verbose –bits 2192`.

Der Befehl unterstützt die folgenden Optionen:

- **`-output`** Diese Angabe ist erforderlich und gibt die Datei an, in der der erzeugte Schlüssel abgespeichert werden soll. Die Angabe der Standardausgabe ist mit - möglich.
- **`-quiet`** Diese Angabe unterdrückt die ausführlichen Informationen des Kommandos.
- **`-bits`** Hiermit ist es möglich die Länge des Schlüssels anzugeben. Default sind 2192 Bits. Dies sollte nicht reduziert werden.
- **`-hostname`** Innerhalb des Schlüssels wird ein Rechnername angegeben. Diese Option erlaubt es einen alternativen Namen zu wählen.

Dieser Befehl ist bei der Erzeugung der Schlüssel dem Befehl `ipsec rsasigkey` vorzuziehen, da er die Schlüssel in ihrer richtigen Syntax erzeugt.

5.5.3 Manuelle Verbindung

Dieses Kapitel zeigt den Aufbau einer manuellen Verbindung. Sie wird im Rahmen einer Schritt für Schritt-Anleitung aufgebaut, so dass Sie die entsprechenden Vorgänge direkt nachvollziehen können.

Eine manuell verschlüsselte Verbindung bedeutet, dass der Administrator die für die Authentifizierung und Verschlüsselung der Pakete verwendeten Schlüssel fest vorgibt. Sie werden nicht mit dem Internet Key Exchange (IKE) Protokoll automatisch ausgehandelt. Das IKE Protokoll tauscht diese Schlüssel auch in regelmäßigen Abständen aus und kann Perfect Forward Secrecy garantieren. Das bedeutet, dass durch das Knacken eines Schlüssels nicht die vorher verwendeten Schlüssel ermittelt werden können. Diese Sicherheit ist bei einer manuell verschlüsselten Verbindung nicht gegeben. Hier ist es die Aufgabe des Administrators, die Sicherheit der Schlüssel zu gewährleisten und für ihren regelmäßigen Austausch zu sorgen. Je länger für einen Tunnel identische Schlüssel verwendet werden, desto größer ist die Gefahr, dass ein Angreifer in der Lage ist durch eine Kryptoanalyse, durch einen Einbruch auf einem der Gateways oder durch Social Engineering die Schlüssel zu ermitteln. Er kann dann sämtliche Nachrichten (auch vorher verschlüsselt mitgeschnittene Informationen) zu entschlüsseln und zu lesen.

Es soll hier nochmal darauf hingewiesen werden, dass automatisch verschlüsselte Verbindungen erstens wesentlich sicherer sind und zweitens wesentlich weniger administrativen Aufwand verursachen. Die Verwendung manuell verschlüsselter Verbindungen ist nur dann sinnvoll, wenn die Gegenseite nicht in der Lage ist mit dem IKE Protokoll umzugehen.

Der hier verwendete Testaufbau ist im Kapitel Testumgebungen ausführlich beschrieben. Dort befinden sich auch Hinweise, wie mit VMware oder User Mode Linux diese Umgebungen virtuell aufgebaut werden können.

Erzeugung der Datei /etc/ipsec.conf

Zunächst soll die Konfigurationsdatei `/etc/ipsec.conf` erstellt werden. Sie enthält die gesamte Konfiguration und die Schlüssel für manuell verschlüsselte Verbindungen. Aus diesem Grund ist der Sicherheit dieser Datei besonderes Augenmerk zu widmen. Ich werde weiter unten darauf zurückkommen.

Die Konfigurationsdatei wird zunächst aus zwei Abschnitten bestehen. Der erste Abschnitt definiert die Parameter für den Start von FreeS/WAN. Der zweite Abschnitt definiert dann die manuell verschlüsselte Verbindung.

Der `config setup` Abschnitt sollte bei einer manuell verschlüsselten Verbindung die in Listing 5.20 aufgeführten Einträge enthalten.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        pluto=no
        manualstart=""
```

Listing 5.20 Manuell verschlüsselte Verbindung: `config setup`

Da es sich um manuell verschlüsselte Verbindungen handelt, wird nicht das IKE Protokoll verwendet. Daher ist es auch nicht erforderlich, dass der IKE Daemon Pluto gestartet wird. Die Angabe `interfaces` dient der Zuordnung der virtuellen `ipsecX` Netzwerkkarte zur entsprechenden physikalischen Netzwerkkarte. Hier ist häufig auch die Verwendung des Eintrages `%defaultroute` möglich. Für einen ersten Test ist es häufig sinnvoll das Debugging zunächst auszuschalten. Wenn die Verbindung nicht aufgebaut werden kann, kann das Debugging immer noch aktiviert werden. Der Parameter `plutodebug` kann jedoch sowieso auf `none` gesetzt oder gleich weggelassen werden, da Pluto nicht gestartet wird. Sobald die Verbindungen funktionieren, ist es auch möglich den Namen der Verbindung bei der Direktive `manualstart` einzutragen. Damit wird dann sichergestellt, dass die Verbindung beim Start von FreeS/WAN gestartet wird.

Anschließend kann nun die Verbindung definiert werden. Sie wird in diesem Beispiel `man-newyork-berlin` genannt. Der gewählte Name für die Verbindung ist vollkommen willkürlich. Es sollte sich jedoch zu Dokumentationszwecken um einen sinnvollen, nachvollziehbaren Namen handeln.

```
        # leftsubnet bleibt in diesem Beispiel leer
        # So wird ein Host-Host Tunnel aufgebaut.
conn man-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
        # Die SPI kann frei in dem Bereich von 0x000-0xfff gewählt werden
        spi=0x200
        # Wir wünschen das ESP Protokoll mit 3DES
        # Verschlüsselung und MD5 HMAC Digests
        esp=3des-md5-96
```

Listing 5.21 Manuell verschlüsselte Verbindung: `conn man-newyork-berlin`

Die Parameter `left` und `right` definieren die IP Adressen der VPN Gateways. Die Parameter `leftnexthop` und `rightnexthop` geben die jeweiligen Standardgateways an (siehe Kapitel 11, »Testumgebungen«)

Dieser Verbindung fehlen nun noch die Schlüssel. Sie sollten möglichst zufällig erzeugt werden. Dies stellt sicher, dass sie nicht einfach erraten werden können. Zusätzlich soll hier nochmals der Hinweis angebracht werden, dass die Schlüssel regelmäßig ausgetauscht werden sollten.

Die Erzeugung der Schlüssel kann sehr komfortabel mit dem Befehl `ipsec ranbits` erfolgen (siehe Listing 5.22). Es wird ein 128 Bit Schlüssel für MD5 und ein 192 Bit Schlüssel für 3DES benötigt.

```
# ipsec ranbits 192
0x3f0b868a_d03e68ac_c6e4e464_4ac8bb80_ecea3426_d3d30ada
# ipsec ranbits 128
0xbf9a081e_7ebdd4fa_824c822e_d94f5226
```

Listing 5.22 Erzeugung der manuellen Schlüssel

TIPP

Ein DES Schlüssel ist 64 Bit lang. Hierbei handelt es sich bei 8 Bits um Paritätsbits. Die für die Verschlüsselung effektive Schlüssellänge beträgt also 64-8=56 Bit. Ein 3DES Schlüssel ist analog 192 Bits lang. Auch dieser Schlüssel enthält 24 Paritätsbits. Die effektive Länge beträgt also 192-24=168 Bits.

Die so ermittelten Schlüssel können nun in der Datei `/etc/ipsec.conf` eingetragen werden. Die fertige Datei ist in Listing 5.23 dargestellt.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        pluto=no
        manualstart=""

conn man-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
```

```
# Die SPI kann frei in dem Bereich von 0x000-0xfff gewählt werden
spi=0x200
# Wir wünschen das ESP Protokoll mit 3DES
  Verschlüsselung und
# MD5 HMAC Digests
esp=3des-md5-96
espenckey=0x3f0b868a_d03e68ac_c6e4e464_
          4ac8bb80_ecea3426_d3d30ada
espauthkey=0xbf9a081e_7ebdd4fa_824c822e_d94f5226
```

Listing 5.23 Manuell verschlüsselte Verbindung: fertige `/etc/ipsec.conf`

Diese Konfiguration muss nun auf beiden FreeS/WAN Gateways, die den Tunnel aufbauen sollen, durchgeführt werden. Angenehm bei FreeS/WAN ist die Tatsache, dass hierzu die Datei lediglich einmal erzeugt werden muss. Anschließend kann die identische Datei auf das zweite Gateway kopiert werden. Denken Sie bei der Kopieraktion aber daran, dass Sie die Geheimhaltung der Schlüssel sicherstellen!

Start der Verbindung

Wenn die Konfigurationsdatei auf beiden Systemen abgespeichert wurde, kann ipsec gestartet werden. Hierzu ist es erforderlich, dass auf beiden Systemen der Befehl `ipsec setup start` eingegeben wird.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
ipsec_setup: WARNING: eth0 has route filtering turned on, KLIPS may not
work
ipsec_setup:  (/proc/sys/net/ipv4/conf/eth0/rp_filter = `1', should be 0)
```

Die Warnmeldung kann in diesem Fall ignoriert werden. Wenn diese Meldung stört, so kann `rp_filter` abgeschaltet werden.

```
# sysctl -w net.ipv4.conf.eth0.rp_filter=0
oder
# echo "0" > /proc/sys/net/ipv4/conf/eth0/rp_filter
```

Dauerhaft ist diese Einstellung in der Datei `/etc/sysctl.conf` möglich.

Anschließend muss auf beiden Gateways die Verbindung gestartet werden.

```
# ipsec manual --up man-newyork-berlin
```

Nun kann mit `ping` und `tcpdump` der Tunnel getestet werden. Hierzu wird auf dem Rechner *Newyork* ein Ping auf *Berlin* abgesetzt: `ping 5.0.0.1`. Dieser Ping sollte erfolgreich sein. Auf dem Router, der sich zwischen *Newyork* und *Berlin* befindet, kann nun mit `tcpdump` der Verkehr verfolgt werden. Die Listings 5.24 und 5.25 zeigen den Ping Verkehr in Klartext und verschlüsselt.

```
15:10:58.018732 5.0.0.1 > 3.0.0.1: icmp: echo request (DF)
15:10:58.026198 3.0.0.1 > 5.0.0.1: icmp: echo reply
15:10:59.053507 5.0.0.1 > 3.0.0.1: icmp: echo request (DF)
15:10:59.060959 3.0.0.1 > 5.0.0.1: icmp: echo reply
```

Listing 5.24 Ping Verkehr unverschlüsselt

```
14:43:41.575897 5.0.0.1 > 3.0.0.1: ESP(spi=0x00000200,seq=0x22)
14:43:41.579359 3.0.0.1 > 5.0.0.1: ESP(spi=0x00000200,seq=0xd)
14:43:42.611291 5.0.0.1 > 3.0.0.1: ESP(spi=0x00000200,seq=0x23)
14:43:42.618982 3.0.0.1 > 5.0.0.1: ESP(spi=0x00000200,seq=0xe)
```

Listing 5.25 Ping Verkehr verschlüsselt

Verbesserungen und Anmerkungen

Diese manuelle Verbindung sichert nun den Verkehr zwischen *NewYork* und *Berlin*. Wenn sich hinter den Gateways weitere Netze befinden, die sich ebenfalls verschlüsselt unterhalten sollen, so werden hierfür weitere eigene Tunnel benötigt. Die Anzahl der aufgesetzten Tunnel ist relativ unkritisch. Es ist möglich mehrere Hundert Tunnel zu definieren.

Hier soll daher nur ein weiterer Tunnel beschrieben werden, der zwei Subnetze hinter diesen Gateways verbindet. Die verwendeten IP Adressen und den Aufbau der Testumgebung können Sie wieder dem entsprechenden Kapitel 11, »Testumgebungen« entnehmen.

Der Tunnel `man-newyorknet-berlinnet` wird im Listing 5.26 beschrieben.

```
conn man-newyorknet-berlinnet
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightsubnet=10.0.2.0/24
        # Die SPI kann frei in dem Bereich von 0x000-0xfff gewählt werden
        spi=0x200
        # Wir wünschen das ESP Protokoll mit 3DES Verschlüsselung und
        # MD5 HMAC Digests
        esp=3des-md5-96
        espenckey=0x3f0b868a_d03e68ac_c6e4e464_4ac8bb80_
                  ecea3426_d3d30ada
        espauthkey=0xbf9a081e_7ebdd4fa_824c822e_d94f5226
```

Listing 5.26 Manuell verschlüsselte Verbindung: Subnet-Subnet

Hierbei fällt unangenehm auf, dass die Schlüssel erneut definiert werden müssen. Um derartige häufig wiederkehrende Angaben abzukürzen gibt es zwei mögliche Lösungsansätze. Die Angaben können als Standardwerte in einer Verbindung `conn %default` definiert werden. Sie gelten dann für alle Verbindungen, außer sie werden dort wieder neu definiert. Die zweite Variante, die hier gewählt wird, ist die Definition in einer eigenen Verbindung und das anschließende Einlesen mit dem Schlüsselwort `also` (siehe Listing 5.27).

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        pluto=no
        manualstart=""

conn man-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
        # Die SPI kann frei in dem Bereich von 0x000-0xfff gewählt werden
        spi=0x200
        also=man_keys

conn man-newyorknet-berlinnet
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightsubnet=10.0.2.0/24
        # Die SPI kann frei in dem Bereich von 0x000-0xfff
          gewählt werden
        spi=0x200
        also=man_keys

conn man_keys
        # Wir wünschen das ESP Protokoll mit 3DES
          Verschlüsselung und
        # MD5 HMAC Digests
        esp=3des-md5-96
        espenckey=0x3f0b868a_d03e68ac_c6e4e464_4ac8bb80_
                  ecea3426_d3d30ada
        espauthkey=0xbf9a081e_7ebdd4fa_824c822e_d94f5226
```

Listing 5.27 Auslagern der Schlüssel in eine eigene Verbindung

Dies vereinfacht die Verwaltung der Schlüssel. Wenn nun die Schüssel ausgetauscht werden sollen, so genügt der Austausch an einer Stelle.

Da jedoch die Geheimhaltung der Schlüssel höchste Wichtigkeit hat, sollte überlegt werden, ob die Schlüssel in einer eigenen Datei vorgehalten werden. Dies ist mit FreeS/WAN sehr einfach möglich. Hierzu werden die Schlüssel in einer eigenen Datei, zum Beispiel `/etc/ipsec.d/conn.man_keys.conf` abgespeichert (siehe Listing 5.28).

```
conn man_keys
        # Wir wünschen das ESP Protokoll mit 3DES
          Verschlüsselung und
        # MD5 HMAC Digests
        esp=3des-md5-96
        espenckey=0x3f0b868a_d03e68ac_c6e4e464_4ac8bb80_
                  ecea3426_d3d30ada
        espauthkey=0xbf9a081e_7ebdd4fa_824c822e_d94f5226
```

Listing 5.28 Die Datei `/etc/ipsec.d/conn.man_keys.conf`

Diese Datei sollte dann mit sehr eingeschränkten Rechten versehen werden.

```
-r--------    1 root     root          267 Feb  6 18:42
/etc/ipsec.d/conn.man_keys.conf
```

Die eigentliche Datei `/etc/ipsec.conf` enthält dann keine Schlüssel mehr. Damit FreeS/WAN dennoch die Schlüssel findet, wird die Datei `/etc/ipsec.d/conn.man_keys.conf` mit der `include` Direktive eingelesen. Die fertige Datei `/etc/ipsec.conf` sieht dann so aus, wie in Listing 5.29 dargestellt.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        pluto=no
        manualstart="man-newyork-berlin
        man-newyorknet-berlinnet"

conn man-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
        # Die SPI kann frei in dem Bereich von 0x000-0xfff gewählt werden
```

```
        spi=0x200
        also=man_keys

conn man-newyorknet-berlinnet
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightsubnet=10.0.2.0/24
        # Die SPI kann frei in dem Bereich von 0x000-0xfff gewählt werden
        spi=0x200
        also=man_keys

include /etc/ipsec.d/conn.*.conf
```

Listing 5.29 Die Datei `ipsec.conf` mit `include` für die Datei `/etc/ipsec.d/conn.man_keys.conf`

Wichtig bei der Verwendung der `also` Direktive ist, dass die Verbindung, auf die sich die Direktive bezieht, nicht vorher sondern später in der Datei definiert werden muss. In diesem Fall ist es also wichtig, dass der `include` Parameter sich am Ende der Datei befindet, da er bei der Auswertung durch die entsprechende Datei ersetzt wird. Vorteilhaft bei der Anwendung ist die Möglichkeit des Fileglobbing mit Wildcards wie zum Beispiel `*?`.

Fazit

Manuelle Verbindungen sind sehr einfach zu konfigurieren. Jedoch werden die Schlüssel nicht mit dem IKE Protokoll ausgehandelt und in regelmäßigen Abständen ausgetauscht. Bei automatisch verschlüsselten Verbindungen wird dies bei FreeS/WAN von Pluto durchgeführt. Wenn ein Angreifer die verwendeten manuell festgelegten Schlüssel ermitteln kann, kann er sämtliche übertragenen Daten entschlüsseln. Ein großes Problem stellt auch der erste sichere Austausch der Schlüssel dar. Da zu diesem Zeitpunkt noch keine verschlüsselte Verbindung existiert, steht der Administrator vor einem Henne-Ei Problem: Die Schlüssel müssen geheim übertragen werden, jedoch steht noch kein VPN zur Verfügung. Daher müssen andere Kanäle genutzt werden um diese Schlüssel zu übertragen. Hier können Disketten, SSH Verbindungen, PGP verschlüsselte E-Mails und persönliche Kommunikation genutzt werden. Dies ist jedoch immer auch mit Sicherheitsproblemen unterschiedlicher Art verbunden.

Wenn möglich sollte daher immer eine automatisch verschlüsselte Verbindung mit dem Pluto IKE Daemon gewählt werden (siehe nächster Abschnitt).

5.5.4 Automatische Verbindung mit dem Pluto IKE Daemon

Dieses Kapitel zeigt den Aufbau einer automatischen Verbindung. Sie wird im Rahmen einer Schritt für Schritt-Anleitung aufgebaut, so dass Sie die entsprechenden Vorgänge direkt nachvollziehen können.

Bei einer automatisch verschlüsselten Verbindung gibt der Administrator lediglich die zu verwendende Authentifizierung für den Aufbau der Verbindung fest vor. Die tatsächlich verwendeten Sitzungsschlüssel für die Verschlüsselung und Authentifizierung der Pakete werden mit dem Internet Key Exchange (IKE) Protokoll automatisch ausgehandelt, nachdem die Authentifizierung erfolgreich durchgeführt wurde. Das IKE Protokoll tauscht diese Schlüssel auch in regelmäßigen Abständen aus und kann dabei Perfect Forward Secrecy (PFS) garantieren. Das bedeutet, dass durch das Knacken eines Schlüssels nicht die vorher verwendeten Schlüssel ermittelt werden können. Eine manuell verschlüsselte Verbindung kann keine PFS bieten, da die Schlüssel während der gesamten Verbindung nicht verändert werden. Dies ist die Aufgabe des Administrators. Er muss die Sicherheit der Schlüssel gewährleisten und auch für ihren regelmäßigen Austausch sorgen. Je länger für einen Tunnel identische Schlüssel verwendet werden, desto größer ist die Gefahr, dass ein Angreifer in der Lage ist, durch Kryptoanalysen, durch einen Einbruch auf einem der Gateways oder durch Social Engineering die Schlüssel zu ermitteln. Er ist dann in der Lage bei einer manuell verschlüsselten Verbindung, sämtliche Nachrichten (auch vorher verschlüsselt mitgeschnittene Informationen) zu entschlüsseln und zu lesen. Bei einer automatisch verschlüsselten Verbindung mit PFS kann er lediglich in dem Zeitrahmen die Nachrichten entschlüsseln, in dem dieser geknackte Schlüssel gültig ist.

Es soll hier nochmal darauf hingewiesen werden, dass automatisch verschlüsselte Verbindungen erstens wesentlich sicherer sind und zweitens wesentlich weniger administrativen Aufwand verursachen. Die Verwendung manuell verschlüsselter Verbindungen sollte daher, wenn möglich, vermieden werden.

Der hier verwendete Testaufbau ist im Kapitel Testumgebungen ausführlich beschrieben. Dort befinden sich auch Hinweise, wie mit VMware oder User Mode Linux diese Umgebungen virtuell aufgebaut werden können.

Konfiguration

Zunächst soll auch für die automatisch verschlüsselte Verbindung die Konfigurationsdatei `/etc/ipsec.conf` erstellt werden. Sie enthält die gesamte

Konfiguration des Tunnels. Im Gegensatz zur manuell verschlüsselten Verbindung enthält sie nicht die Schlüssel. Die für die Authentifizierung der Kommunikationspartner verwendeten Schlüssel werden in der Datei `/etc/ipsec.secrets` abgespeichert. Diese sehr wichtige Datei wird weiter unten behandelt. Darum ist lediglich sicherzustellen, dass niemand außer dem Administrator die Datei `/etc/ipsec.conf` modifizieren darf. Das Lesen dieser Datei ist sicherheitstechnisch unkritisch.

Die Konfigurationsdatei wird zunächst aus zwei Abschnitten bestehen. Der erste Abschnitt definiert die Parameter für den Start von FreeS/WAN. Der zweite Abschnitt definiert dann die automatisch verschlüsselte Verbindung.

Der `config setup` Abschnitt sollte bei einer automatisch verschlüsselten Verbindung die in Listing 5.30 aufgeführten Einträge enthalten.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        # In Version 2 sind die folgenden Parameter nicht mehr erlaubt
        plutoload=%search
        plutostart=%search
```

Listing 5.30 Automatisch verschlüsselte Verbindung: `config setup`

Diese Werte definieren das Startverhalten von FreeS/WAN. Der Parameter `interfaces` ordnet die virtuelle Netzwerkkarte `ipsecX` der entsprechenden physikalischen Netzwerkkarte zu. Wird nur ein virtueller Netzwerkadapter `ipsec0` genutzt, genügt meist auch die Angabe `%defaultroute`. FreeS/WAN ermittelt dann selbst die Netzwerkkarte, über die das Standardgateway erreicht wird.

Das Debugging sollte zunächst deaktiviert werden. Im Falle eines Fehlers kann es dann immer noch aktiviert werden. Wenn die Parameter `plutoload` und `plutostart` auf den Wert `%search` gesetzt werden, so bedeutet das, dass Pluto bei seinem Start in den Verbindungen nach dem Parameter `auto` suchen wird.Diese Einstellung ist Default bei Version 2.x. Der Parameter `auto` definiert dann, ob die Verbindung automatisch geladen, geroutet oder sogar gestartet werden soll. Alle nicht definierten Parameter werden mit ihren entsprechenden Standardwerten genutzt. So wird Pluto per Default gestartet (`pluto=yes`).

Die Konfiguration der eigentlichen Verbindung unterscheidet sich nun in Abhängigkeit der gewählten Authentifizierungsmethode. Im Folgenden soll die Authentifizierung mit einem Preshared Key (PSK), öffentlichen RSA Schlüsseln, und X.509 Zertifikaten besprochen werden. Die Verwendung der X.509 Zertifikate benötigt einen zusätzlichen Patch, da diese Funktionalität bisher nicht fester Bestandteil von FreeS/WAN ist. Jedoch ist deren Verwendung im Zusammenhang mit FreeS/WAN heute sehr üblich, da dies heterogene Lösungen zulässt.

Die einzelnen Methoden und die daraus resultierenden Konfigurationsdateien werden in den folgenden Abschnitten besprochen.

Authentifizierung mit Preshared Keys (PSK)

Das IKE Protokoll erlaubt die Authentifizierung der Kommunikationspartner mit einem Kennwort. Dieses Kennwort muss vorher zwischen den beiden Partnern ausgetauscht werden. Daher spricht man von einem *Pre Shared Key* (PSK). Die Geheimhaltung dieses Schlüssels ist von immenser Wichtigkeit, da ein Dritter, der Zugang zu diesem Schlüssel erhält, dann in der Lage ist, eine verschlüsselte Verbindung aufzubauen und einen Man-in-the-Middle Angriff zu starten.

Aus diesem Grund sind öffentliche RSA Schlüssel oder X.509 Zertifikate einem PSK vorzuziehen. Leider unterstützen viele kommerzielle Implementierungen diese Authentifizierungen nicht. In diesen Fällen stellt die Authentifizierung mit PSKs die einzige Möglichkeit dar, eine automatisch verschlüsselte Verbindung aufzubauen.

Die Definition einer Verbindung in der Datei `ipsec.conf` für die Authentifizierung mit einem PSK ist in Listing 5.31 dargestellt.

```
conn psk-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
        authby=secret
        auto=""
```

Listing 5.31 Automatisch verschlüsselte Verbindung mit PSK

Die Parameter `left` und `right` definieren die IP Adressen der VPN Gateways. Die Parameter `leftnexthop` und `rightnexthop` geben die jeweiligen Standardgateways an (siehe Kapitel 11, »Testumgebungen«)

Der Parameter `authby` wird hier zur Verständlichkeit noch einmal explizit angegeben. Der Defaultwert bei der Version 1.99 ist `secret`. Ab Version 2.x ist der Defaultwert `rsasig`.

Nun muss noch das Kennwort in der Datei `/etc/ipsec.secrets` angegeben werden. Bei automatisch verschlüsselten Verbindungen werden im Gegensatz zu manuell verschlüsselten Verbindungen die Schlüssel in Datei `/etc/ipsec.secrets` abgespeichert. Diese Datei sollte dann mit besonderen Rechten vor den Einblicken anderer geschützt werden.

Das gewählte Kennwort sollte möglichst nicht nur aus Buchstaben bestehen sondern eine zufällige Abfolge von Zeichen darstellen. Da diese PSKs jedoch nur schwer einzuprägen sind, werden häufig Wörter oder Sätze aus dem normalen Sprachgebrauch verwendet. Dies reduziert jedoch die Stärke des PSKs ähnlich dem Problem bei Anmeldekennwörtern. Ein Wörterbuchangriff wird recht schnell derartige PSKs ermitteln.

Um nun eine starke PSK zu erzeugen kann der Befehl `ipsec ranbits` verwendet werden.

```
# ipsec ranbits --continuous 256
0xe10bd52b0529b54aac97db63462850f354a
bdd885c19ea6d65d6f4efaeab0222
```

Dieser PSK kann nun in die Datei `/etc/ipsec.secrets` eingetragen werden. Dies ist in Listing 5.32 dargestellt.

```
3.0.0.1 5.0.0.1 : PSK 0xe10bd52b0529b54aac97db63462850f354ab
dd885c19ea6d65d6f4efaeab0222

: PSK "kennwort" # schlechte PSK
```

Listing 5.32 Die Datei `ipsec.secrets` mit PSK

Die abgebildete Datei enthält zwei Schlüssel. Der erste Schlüssel wurde mit `ipsec ranbits` generiert und wird nur verwendet, wenn eine Verbindung zwischen den Gateways 3.0.0.1 und 5.0.0.1 aufgebaut werden soll. Der zweite Schlüssel ist ein universeller Default Schlüssel, der in allen anderen Fällen genutzt wird. Das Schlüsselwort `PSK` definiert, dass in beiden Fällen ein Preshared Key folgt.

Wichtig sind nun die Rechte dieser Datei. Es sollte sichergestellt werden, dass nur *root* in der Lage ist die Datei zu lesen und natürlich zu schreiben.

Hiermit ist die Konfiguration abgeschlossen. Nun kann der Tunnel aufgebaut werden.

Aufbau des Tunnels

Um diesen Tunnel aufzubauen, ist es erforderlich, dass beide Gateways über identische Tunneldefinitionen und Schlüssel verfügen. Im einfachsten Fall werden die entsprechenden Dateien `/etc/ipsec.conf` und `/etc/ipsec.secrets` auf die zweite Maschine kopiert.

Ist der Austausch der Konfigurationen erfolgt, so kann zunächst FreeS/WAN gestartet werden. Anschließend kann der Tunnel geladen und gestartet werden. Hierbei ist es wichtig, dass der Tunnel auf beiden Gateways momentan von Hand geladen wird.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
# ipsec auto --add --verbose psk-newyork-berlin
002 added connection description "psk-newyork-berlin"
# ipsec auto --up --verbose psk-newyork-berlin
002 "psk-newyork-berlin" #1: initiating Main Mode
104 "psk-newyork-berlin" #1: STATE_MAIN_I1: initiate
106 "psk-newyork-berlin" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "psk-newyork-berlin" #1: STATE_MAIN_I3: sent MI3, expecting MR3
002 "psk-newyork-berlin" #1: Peer ID is ID_IPV4_ADDR: '5.0.0.1'
002 "psk-newyork-berlin" #1: ISAKMP SA established
004 "psk-newyork-berlin" #1: STATE_MAIN_I4: ISAKMP SA established
002 "psk-newyork-berlin" #2: initiating Quick Mode
PSK+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK
112 "psk-newyork-berlin" #2: STATE_QUICK_I1: initiate
002 "psk-newyork-berlin" #2: sent QI2, IPsec SA established
004 "psk-newyork-berlin" #2: STATE_QUICK_I2: sent QI2, IPsec
                            SA established
```

Die in der Ausgabe definierten Zustände beziehen sich übrigens direkt auf die gesendeten Pakete beim Aufbau der ISAKMP SA und der IPsec SA. Ein `I` im State beschreibt den Initiator und ein `R` den Responder.

Wird nun wieder ein Ping von *NewYork* nach *Berlin* gestartet, so werden diese Pakete verschlüsselt übertragen (Listing 5.33). Zum Vergleich können Sie sich den unverschlüsselten Ping Verkehr in Listing 5.34 ansehen.

```
15:10:58.018732 5.0.0.1 > 3.0.0.1: icmp: echo request (DF)
15:10:58.026198 3.0.0.1 > 5.0.0.1: icmp: echo reply
15:10:59.053507 5.0.0.1 > 3.0.0.1: icmp: echo request (DF)
15:10:59.060959 3.0.0.1 > 5.0.0.1: icmp: echo reply
```

Listing 5.33 Unverschlüsselter Ping Verkehr

```
12:39:15.232200 3.0.0.1 > 5.0.0.1: ESP(spi=0x11363a61,seq=0x1)
12:39:15.239036 5.0.0.1 > 3.0.0.1: ESP(spi=0xe214defa,seq=0x1)
12:39:16.283861 3.0.0.1 > 5.0.0.1: ESP(spi=0x11363a61,seq=0x2)
12:39:16.286828 5.0.0.1 > 3.0.0.1: ESP(spi=0xe214defa,seq=0x2)
```

Listing 5.34 Verschlüsselter Ping Verkehr

Beim verschlüsselten Verkehr fallen im Gegensatz zur manuell verschlüsselten Verbindung (Beispiel 5.25) die hohen Security Parameter Indices (SPI) auf. Pluto verwendet für die automatisch verschlüsselten Verbindungen Werte ab 0x1000.

Verbesserungen und Erweiterungen

Wenn nun zusätzlich zwei Netzwerke, die sich hinter den Gateways befinden, über das VPN kommunizieren sollen, so ist ein weiterer Tunnel erforderlich. Die Parameter, die bei beiden Verbindungen benötigt werden, können in eine `conn %default` ausgelagert werden. Hier wurde das für den Parameter `authby=secret` genutzt. Die Aufnahme weiterer Parameter geht meiner Meinung nach auf Kosten der Lesbarkeit. Die fertige Konfigurationsdatei `/etc/ipsec.conf`, die dies einschließt, ist in Listing 5.35 abgebildet.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        # In Version 2 sind die folgenden beiden Parameter
          nicht mehr erlaubt
        plutoload=%search
        plutostart=%search

conn %default
        authby=secret
        keyingtries=0

conn psk-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftid=vpn@newyork
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightid=vpn@berlin
        auto="start"
```

```
conn psk-newyorknet-berlinnet
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        leftid=vpn@newyork
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightsubnet=10.0.2.0/24
        rightid=vpn@berlin
        auto="add"
```

Listing 5.35 Komplette Datei: `/etc/ipsec.conf`

Zusätzlich wurde in dieser Datei der Wert des Parameters `auto` modifiziert. So wird die Verbindung `psk-newyork-berlin` direkt beim Start von FreeS/WAN aufgebaut. Die zweite Verbindung `psk-leftnet-rightnet` wird lediglich geladen. So besteht die Möglichkeit, dass diese Verbindung von der anderen Seite aufgebaut werden kann.

Außerdem wurde der Parameter `keyingtries=0` aufgenommen. Er stellt sicher, dass FreeS/WAN unendlich lange versuchen wird, die Verbindung `psk-newyork-berlin` aufzubauen. Bei aufgebauten Verbindungen führt dieser Parameter dazu, dass FreeS/WAN auch hier unendlich lange versuchen wird, einen neuen Schlüsselaustausch durchzuführen beziehungsweise ausgefallene Verbindungen wieder herzustellen. Dies kann sich auf die Performanz auswirken. Diese Einstellung ist nur dann sinnvoll, wenn die IP Adressen der Gegenstelle bekannt sind und diese dauerhaft verfügbar ist. Wenn es sich bei der Gegenstelle um sogenannte Roadwarrior mit wechselnden IP Adressen handelt, sollte `keyingtries=1` gesetzt werden. Um eine einfache Lesbarkeit der Tunnel und der Protokolldateien zu erreichen, wurden den Gateways symbolische Namen mit den Parametern `leftid` und `rightid` zugewiesen. Werden diese Parameter nicht genutzt, so wird die IP Adresse an ihrer Stelle genutzt.

Fazit

Die Konfiguration einer automatisch verschlüsselten Verbindung mit PSKs ist recht einfach. Hierbei werden die Schlüssel zur Authentifizierung in einer eigenen getrennten Datei (`/etc/ipsec.secrets`) gespeichert. Die Dateien `/etc/ipsec.conf` können sehr einfach zwischen den Tunnelendpunkten ausgetauscht werden. Die tatsächlichen Sitzungsschlüssel für die Authentifizierung und Verschlüsselung der Pakete werden von Pluto mit dem IKE Protokoll automatisch ausgehandelt und regelmäßig ausgetauscht.

Leider muss jedoch der Preshared Key (PSK) zwischen den beiden Gateways ausgetauscht werden. Dieser Schlüssel wird für die erste Authentifizierung

zwischen den beiden Gateways genutzt. Er dient als Grundlage des anschließenden Diffie Hellmann Schlüsselaustausches. Wenn dieser Schlüssel in die Hände Dritter gerät, so können diese einen Man-in-the-middle Angriff ausführen. Die Geheimhaltung des PSKs hat also höchsten Vorrang. Dies ist jedoch nur sehr schwer zu gewährleisten, da dieser Schlüssel auf beiden Systemen identisch sein muss. Das bedeutet, er muss vor dem Aufbau des VPNs auf einem alternativen sicheren Weg ausgetauscht werden.

Automatisch verschlüsselte Verbindungen, die öffentliche RSA Schlüssel oder X.509 Zertifikate verwenden, weisen dieses Problem nicht auf.

Automatisch verschlüsselte Verbindung mit öffentlichen RSA Schlüsseln

Der Aufbau automatisch verschlüsselter Verbindungen mit öffentlichen RSA Schlüsseln stellt die sinnvollste Variante bei dem ungepatchten FreeS/WAN dar. Hierbei wird für jedes VPN Gateway ein RSA Schlüsselpaar erzeugt, das dieses eindeutig auszeichnet. Für die Authentifizierung der VPN Gateways untereinander ist lediglich der Austausch der öffentlichen RSA Schlüssel erforderlich. Dieser Austausch kann als Klartext erfolgen. Hierbei ist eine Geheimhaltung vollkommen überflüssig. Jedoch muss hier sichergestellt werden, dass der Ursprung des Schlüssels nachvollzogen werden kann. Hier ist also nicht die Vertraulichkeit des Schlüssels, sondern seine Authentizität und Integrität wichtig. Dies kann zum Beispiel gewährleistet werden, indem der Schlüssel von einer Person auf beiden VPN Gateways installiert oder der Fingerabdruck des Schlüssels telefonisch abgeglichen wird.

Hier soll nun zunächst die Erzeugung der Schlüssel vorgestellt werden. Es handelt sich dabei normalerweise um RSA Schlüssel mit 2192 Bits Länge. Sie können mit den Befehlen `ipsec rsasigkey` oder `ipsec newhostkey` erzeugt werden.

Relativ einfach ist die Erzeugung der Schlüssel mit dem Befehl `ipsec newhostkey`. Dieser Befehl erwartet die Angabe einer Datei und erzeugt die Schlüssel in dieser Datei.

```
# ipsec newhostkey --output /etc/ipsec.secrets
```

Dabei werden die Daten an eine vorhandene Datei angehängt. Wenn dies nicht gewünscht wird, sollte die Datei vorher gelöscht oder umbenannt werden.

Die Erzeugung der Schlüssel muss nun auf beiden Gateways erfolgen. Anschließend sollten die erzeugten Dateien ähnlich aussehen wie im Listing 5.36.

```
: RSA   {
        # RSA 2192 bits   newyork.spenneberg.de   Fri Feb  7 14:09:21 2003
        # for signatures only, UNSAFE FOR ENCRYPTION
        #pubkey=0sAQ
NuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKiNrLgyNfBo9npxZdOmo
lWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2DCPO5yS2OEa6HrsOjt
i+tyIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+HjJGbcpPEBL
NHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qdbuh
o9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529N
pBHqX69oQVsncnuFderIKp7zrP4HUMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH5h3
iG/EDUofMl9TjqJ
R7L4x22H0sdV6txoRC4R0LT
        #IN KEY 0x4200 4 1
AQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKiNrLgyNfBo9npxZdOmol
WaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2DCPO5yS2OEa6HrsOjti+t
yIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+HjJGbcpPEBLNHrB1fOzIQBgS1
+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qdbuho9pnjPP4sCsx
o9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBHqX69oQVsncnuFder
IKp7zrP4HUMWHhMjz0DWzL3
WQWyvdfMp9qGLJwoH5h3iG/EDUofMl9TjqJR7L4x22H0sdV6txoRC4R0LT
        # (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
        Modulus: 0x6e2e938b5cd435472599c7a0873d678d18645e5effcab93424107
64a88dacb83235f068f67a7165d3a6a2559a0cda4f2f3ca19002d67a0ff4
194
29648eec64582d27d5b80685d8308f3b9c92d8e11ae87aec3a3b62fadc88
cb62f14954fbf29f781558b1b00a09c988e73b21ac65c1be1e32466d
ca4f1012cd1eb0757cecc8401812d7e46dad87d64a54fcec12d84704778
dc5755a5b56158a1719c948dea75bba1a3da678cf3f8b02b31a3dc7e
cdec68c3fbbb0e2142859b82603b4a29e39a52605d941e76f4da411e
a5faf68415b27727b8575eac82a9ef3acfe0750c58784c8f3d035b32f7
5905b2bdd7cca7da862c9c281f9877886fc40d4a1f325f538ea251ecbe
31db61f4b1d57ab71a110b
84742d3
        PublicExponent: 0x03
        # everything after this point is secret
        PrivateExponent:
0x125d18973a235e3686444bf0168a3becd9660fba7ff71ede0602be6
1c179cc95db3a8117e69bd90f89bc5b8ef02246287df7042ab23bf02a
8aee06e617d210b95cdbf8f401164eb2c289ef6dced02f26bf275f09e5d
47a16cc907d8c38d4a86fe958e41d9d5701a196d134859cbba04a5a5d
b667a1b7d803222fc81394d22160040323fb679cebf90c2b64375dff53
826420479eea2288d0de619d645e8d55c60972d6d8242ee9cab976615
6b7f4441b7c54bab064d3610eeff8b269740018e6af706b521845ef55d
a0788f55937a2057900f3ae2a852d7c8b4b2bac65d2767ef4fecdd531b
4db89881b9df4237c2bbda00e577c568bb2d0196c0553622f041229c3b
099da1c1f52b0c72f1bf37219c57707
        Prime1: 0xb37ee25438be25702dfaf1a723ac15c97909d5c7f8fb822dd776323a
ea71ad07127b2575d2a91691a9680b7fa63162304caf38202f0fa18b00
7e52c02b04f55edbfb86b296a7b0c07ad0a4661d2c9239eeed1c7f8c
```

```
f0c5dcdfe1289a60f5a500cb6028ef037280e2f4cc9ac914b75f
d2d477427495c4e86507b26618fd18030cb0b35a1a3bb068b03f
        Prime2: 0x9d24bd8aa393d0ae9e1cca3668751a8a545f9949
95a9355391be6364cb531d
a2da571002609d0b1099f3830579b09de8052603ca6e949b7e62d7029
5a0acab96e12e689e5ce755dbfafebc858bed2175fb0ae06e49cfc52e1b
1c0dee915ab229bac99b06912f702677a874b71d9ebb
d5780abc64455b94c580f5e1e8104b57ebfbdd6fd2286d3dc86d
        Exponent1:
0x77a9ec3825d418f573fca11a17c80e8650b1392ffb5256c93a4ecc2746f
6735a0c5218f9371b64611b9ab255197641758874d01574b5165caafe
e1d572034e3f3d5259cc646fcb2afc8b184413730c269f48bdaa5df5d9
3dea961b1195f918ab32401b4a024c55eca33311db632
4ea8c8da4d6f863d89aee05219965fe10020875cce6bc27caf0757f
        Exponent2:
0x68c3290717b7e074696886cef04e11b18d9510dbb91b78e261299798878
cbe6c918f600195be076066a25758fbcb13f0036ead319f0dbcfeec8f57
0e6b1dc7b9eb7445bee89a393d51ff285907f36ba3fcb1eaf431352e1eb
cbd5e9f0b91cc1bd1dbbcaf0b74f56efa704dcf691
47d38fab1d2ed8392632e55f9414560323a9d52939fe17048d3daf3
        Coefficient:
0xa79b3e5711eb1daf64ca38598d416665896decb9ac1d839cf71fbbf49
4d9bc9136a8325950162c56d4e676a259f071807cb3e648e5607612df
cbc204a5b16f32832b3061208f180a920eb4bc83547794ff524c245f22
65edca68ed1320363ec2dcbc0e628808fb68adf868f48
8565524e71a8a6d2b5f3013391fcb996eacc8414192df513e76a22a7f
        }
```

Listing 5.36 Die Datei `/etc/ipsec.secrets` mit RSA Schlüsseln

Für die Definition der Verbindung in der Datei `/etc/ipsec.conf` werden die öffentlichen Schlüssel dieser RSA Schlüssel benötigt. Der öffentliche Schlüssel ist in Listing 5.36 fett markiert. Anstatt jedoch den Schlüssel von Hand per Copy&Paste in der Konfigurationsdatei einzufügen, kann er elegant extrahiert werden. Hierzu ist es erforderlich auf dem Gateway *NewYork* den folgenden Befehl einzugeben:

```
# ipsec showhostkey --left
        # RSA 2192 bits   newyork.spenneberg.de   Fri Feb  7 14:09:21 2003
        leftrsasigkey=0sAQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKi
        NrLgyNfBo9npxZdOmolWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2
        DCPO5yS2OEa6HrsOjti+tyIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+HjJGb
        cpPEBLNHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qdbu
        ho9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBHqX69
        oQVsncnuFderIKp7zrP4HUMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH5h3iG
        /EDUofMl9TjqJR7L4x22H0sdV6txoRC4R0LT
```

Analog ist der Befehl auf dem Gateway *Berlin* einzugeben.

Die Definition einer Verbindung für die Authentifizierung mit RSA Schlüsseln ist in Listing 5.37 dargestellt.

```
conn rsa-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftrsasigkey=0sAQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKi
        NrLgyNfBo9npxZdOmolWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2
        DCPO5yS2OEa6HrsOjti+tyIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+H
        jJGbcpPEBLNHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3
        qdbuho9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBH
        qX69oQVsncnuFderIKp7zrP4HUMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH
        5h3iG/EDUofMl9TjqJR7L4x22H0sdV6txoRC4R0LT
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightrsasigkey=0sAQOLr0P0yVDh3SETo7p2J+RC6PvbnR7
        nI85QKVUlfkJGODbJF6ZhrxdFWAjx6gF0oh9BM9BpJoSxJdkwxMTW00Q
        vGSZ9OSPxogb7m52RqJwfYaQVka6xjf7f/6xXIful4dWfUFOlaemZf8L+u
        ox5RT7RR8ZU7w8uDqV38bbh6Odo3hoBDFkdZ4yplW+HKqScqTVtTZgSP
        cT7rhSt6XqUbBdUMgpTNEeMNKNxUyE6Gabl+ejpULlN7KSebRti6o6bUU1ai
        qahk64oAPL03B1ivqqDSB+iMVSJM1NnltwThlZ0FNobbTXXL3BaccQrUDb
        Qxm/DCxn1SFN64i633/Q384+uxI92rQ8hCxLJL/YPVTSwc9S/
        authby=rsasig
        auto=""
```

Listing 5.37 Automatisch verschlüsselte Verbindung mit RSA Schlüsseln

Am einfachsten ist die Konfiguration des Tunnels, wenn beide öffentlichen Schlüssel zunächst in der Datei eingetragen werden. Dann kann diese Datei anschließend auf beide Gateways kopiert werden. Es sind dann keine Modifikationen der Datei mehr erforderlich.

Wichtig sind nun die Rechte der Datei `/etc/ipsec.secrets`. Da diese Datei die privaten Schlüssel der VPN Gateways enthält, sollte sichergestellt werden, dass nur `root` in der Lage ist die Datei zu lesen und natürlich zu schreiben.

Sind die Rechte angepasst, so ist die Konfiguration abgeschlossen. Nun kann der Tunnel aufgebaut werden.

Aufbau des Tunnels

Um diesen Tunnel aufzubauen, ist es erforderlich, dass beide Gateways über identische Tunneldefinitionen verfügen und ihre öffentlichen Schlüssel ausgetauscht haben. Wenn Sie so vorgegangen sind, wie bisher beschrieben, ist dies der Fall, da die öffentlichen Schlüssel beider Gateways sich in der Konfigurationsdatei befinden. Die Datei `/etc/ipsec.secrets` enthält den privaten RSA Schlüssel und darf *nicht* ausgetauscht werden.

Ist der Austausch der Konfigurationen erfolgt, so kann zunächst FreeS/WAN gestartet werden. Anschließend kann der Tunnel geladen und gestartet werden. Hierbei ist es wichtig, dass der Tunnel auf beiden Gateways momentan von Hand geladen werden muss.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
# ipsec auto --add --verbose rsa-newyork-berlin
002 added connection description "rsa-newyork-berlin"
# ipsec auto --up --verbose rsa-newyork-berlin
002 "psk-newyork-berlin" #1: initiating Main Mode
104 "psk-newyork-berlin" #1: STATE_MAIN_I1: initiate
106 "psk-newyork-berlin" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "psk-newyork-berlin" #1: STATE_MAIN_I3: sent MI3, expecting MR3
002 "psk-newyork-berlin" #1: Peer ID is ID_IPV4_ADDR: '5.0.0.1'
002 "psk-newyork-berlin" #1: ISAKMP SA established
004 "psk-newyork-berlin" #1: STATE_MAIN_I4: ISAKMP SA established
002 "psk-newyork-berlin" #2: initiating Quick Mode
PSK+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK
112 "psk-newyork-berlin" #2: STATE_QUICK_I1: initiate
002 "psk-newyork-berlin" #2: sent QI2, IPsec SA established
004 "psk-newyork-berlin" #2: STATE_QUICK_I2: sent QI2, IPsec SA
     established
```

Wird nun wieder ein Ping von *NewYork* nach *Berlin* gestartet, so werden diese Pakete verschlüsselt übertragen (Listing 5.39). Zum Vergleich können Sie sich den unverschlüsselten Ping Verkehr in Listing 5.38 ansehen.

```
15:10:58.018732 5.0.0.1 > 3.0.0.1: icmp: echo request (DF)
15:10:58.026198 3.0.0.1 > 5.0.0.1: icmp: echo reply
15:10:59.053507 5.0.0.1 > 3.0.0.1: icmp: echo request (DF)
15:10:59.060959 3.0.0.1 > 5.0.0.1: icmp: echo reply
```

Listing 5.38 Unverschlüsselter Ping Verkehr

```
12:39:15.232200 3.0.0.1 > 5.0.0.1: ESP(spi=0x12473a66,seq=0x1)
12:39:15.239036 5.0.0.1 > 3.0.0.1: ESP(spi=0xe236cef4,seq=0x1)
12:39:16.283861 3.0.0.1 > 5.0.0.1: ESP(spi=0x12473a66,seq=0x2)
12:39:16.286828 5.0.0.1 > 3.0.0.1: ESP(spi=0xe236cef4,seq=0x2)
```

Listing 5.39 Verschlüsselter Ping Verkehr

Beim verschlüsselten Verkehr fallen im Gegensatz zur manuell verschlüsselten Verbindung (Listing 5.25) genauso wie bei der PSK basierten Verbindung die hohen Security Parameter Indices (SPI) auf. Pluto verwendet für die automatisch verschlüsselten Verbindungen Werte ab 0x1000.

Verbesserungen und Erweiterungen

Wenn nun zusätzlich zwei Netzwerke, die sich hinter den Gateways befinden, über das VPN kommunizieren sollen, so ist ein weiterer Tunnel erforderlich. Die Parameter, die bei beiden Verbindungen benötigt werden, können in eine `conn %default` ausgelagert werden. Hier wurde das für den Parameter `authby=rsasig` und die öffentlichen RSA Schlüssel genutzt. Die Aufnahme weiterer Parameter geht meiner Meinung nach auf Kosten der Lesbarkeit. Die fertige Konfigurationsdatei `/etc/ipsec.conf`, die dies einschließt, ist in Listing 5.40 abgebildet.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        # In Version sind die beiden folgenden Parameter nicht
          mehr erlaubt
        plutoload=%search
        plutostart=%search

conn %default
        authby=rsasig
        keyingtries=0
        leftrsasigkey=0sAQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKi
        NrLgyNfBo9npxZdOmolWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2
        DCPO5yS2OEa6HrsOjti+tyIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+HjJGb
        cpPEBLNHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qdbuh
        o9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBHqX69o
        QVsncnuFderIKp7zrP4HUMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH5h3iG/
        EDUofMl9TjqJR7L4x22H0sdV6txoRC4R0LT
        rightrsasigkey=0sAQOLr0P0yVDh3SETo7p2J+RC6PvbnR7nI85QKVUlfkJG
        ODbJF6ZhrxdFWAjx6gF0oh9BM9BpJoSxJdkwxMTW00QvGSZ9OSPxogb
        7m52RqJwfYaQVka6xjf7f/6xXIful4dWfUFOlaemZf8L+uox5RT7RR8ZU7
        w8uDqV38bbh6Odo3hoBDFkdZ4yplW+HKqScqTVtTZgSPcT7rhSt6XqUbB
        dUMgpTNEeMNKNxUyE6Gabl+ejpULlN7KSebRti6o6bUU1aiqahk64oAPL03
        B1ivqqDSB+iMVSJM1NnltwThlZ0FNobbTXXL3BaccQrUDbQxm/DCxn1SF
        N64i633/Q384+uxI92rQ8hCxLJL/YPVTSwc9S/

conn rsa-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
        auto="start"
```

```
conn rsa-newyorknet-berlinnet
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=5.0.0.1
        rightnexthop=5.255.255.254
        leftsubnet=10.0.1.0/24
        rightsubnet=10.0.2.0/24
        auto="add"
```

Listing 5.40 Komplette Datei: `/etc/ipsec.conf`

Zusätzlich wurde in dieser Datei der Wert des Parameters `auto` so modifiziert, dass die Verbindung `rsa-newyork-berlin` direkt beim Start von FreeS/WAN aufgebaut wird und die zweite Verbindung `rsa-leftnet-rightnet` lediglich geladen wird. Damit kann diese Verbindung von der anderen Seite aufgebaut werden.

Hinzugekommen ist der Parameter `keyingtries=0`. Er stellt sicher, dass FreeS/WAN unendlich lange versuchen wird, die Verbindung `rsa-newyork-berlin` aufzubauen. Bei aufgebauten Verbindungen führt dieser Parameter dazu, dass FreeS/WAN auch hier unendlich lange versuchen wird, einen neuen Schlüsselaustausch durchzuführen, beziehungsweise ausgefallene Verbindungen wieder herzustellen. Dies kann sich jedoch negativ auf die Performanz auswirken. Diese Einstellung ist nur dann sinnvoll, wenn die IP Adressen der Gegenstelle bekannt sind und diese dauerhaft verfügbar ist. Wenn es sich bei der Gegenstelle um sogenannte Roadwarrior mit wechselnden IP Adressen handelt, sollte `keyingtries=1` gesetzt werden.

Fazit

Die Konfiguration einer automatisch verschlüsselten Verbindung mit RSA Schlüsseln zur Authentifizierung ist recht einfach. Hierbei werden die privaten Schlüssel in einer eigenen getrennten Datei (`/etc/ipsec.secrets`) gespeichert. Die Dateien `/etc/ipsec.conf` können sehr einfach zwischen den Tunnelendpunkten ausgetauscht werden und benötigen lediglich die Angabe der öffentlichen Schlüssel. Diese dürfen jedoch ohne Bedenken ausgetauscht werden. Die tatsächlichen Sitzungsschlüssel für die Authentifizierung und Verschlüsselung der Pakete werden von Pluto mit dem IKE Protokoll automatisch ausgehandelt und regelmäßig ausgetauscht.

Im Gegensatz zu einer automatischen Verbindung mit Preshared Key (PSK) existiert nicht ein Schlüssel, der zwischen den beiden Gateways ausgetauscht werden muss. Jedes Gateway verwendet einen eigenen, der Gegenstelle unbekannten privaten Schlüssel. Lediglich der öffentliche RSA Schlüssel muss

dem Partner mitgeteilt werden. Dadurch besteht keine Möglichkeit, dass der Schlüssel bei seinem Transport kompromittiert wird. Wenn dieser Schlüssel in die Hände Dritter gerät, können diese nicht wie bei PSK Verbindungen einen Man-in-the-middle Angriff ausführen.

Dennoch ist es erforderlich, die öffentlichen Schlüssel aller VPN Gateways, die untereinander kommunizieren möchten, auszutauschen. Bei lediglich einem Tunnel mit zwei Gateways stellt dies kein Problem dar. Wenn jedoch eine Vielzahl von Gateways verwaltet und administriert werden soll, wird dies recht schnell ein administratives und logistisches Problem. Automatisch verschlüsselte Verbindungen, die X.509 Zertifikate und eine Zertifikatsautorität verwenden, weisen dieses Problem nicht auf.

Automatisch verschlüsselte Verbindung mit X.509 Zertifikaten

FreeS/WAN unterstützt von Haus aus lediglich den Aufbau von automatisch verschlüsselten Verbindungen mit PSKs und RSA Schlüsseln. Die Unterstützung von X.509 Zertifikaten ist nur mit einem zusätzlichen Patch möglich. Dieser Patch steht sowohl für die FreeS/WAN Version 1.9x als auch für die FreeS/WAN Versionen 2.x zur Verfügung.

Der wesentliche Unterschied bei der Verwendung von X.509 Zertifikaten besteht darin, dass nicht sämtliche öffentliche Schlüssel aller beteiligten VPN Gateways untereinander ausgetauscht werden müssen. Dies ist bei der Verwendung von reinen RSA Schlüsseln notwendig und erzeugt einen hohen zusätzlichen Administrationsaufwand. Zertifikate vereinfachen die zentrale Verwaltung durch eine Zertifikatsautorität. Alle FreeS/WAN Gateways müssen dieser Zertifikatsautorität vertrauen. Dies wird umgesetzt, indem alle Gateways über das Zertifikat der CA verfügen.

Baut nun ein VPN Gateway eine Verbindung zu einem weiteren auf, so überträgt es zunächst sein X.509 Zertifikat. Der Empfänger prüft, ob dies von der richtigen RootCA unterzeichnet wurde. Wenn dies der Fall ist, wird dieses Zertifikat genutzt, um anschließend den Partner zu authentifizieren.

Dieses Verfahren wird von den meisten IPsec Implementierungen unterstützt. So können mit FreeS/WAN bei der Verwendung von X.509 Zertifikaten heterogene Netzwerke mit einer Großzahl weiterer freier und kommerzieller Implementierungen aufgebaut werden.

Exkurs: Erzeugung von X.509 Zertifikaten mit OpenSSL

Es existieren viele verschiedene Möglichkeiten um X.509 Zertifikate zu erzeugen. Das Kapitel 8, »Aufbau einer Public Key Infrastruktur« stellt zwei weitere Methoden vor, die bei häufiger Verwendung den Umgang mit Zertifikaten stark vereinfachen können.

Linux bietet jedoch mit dem OpenSSL Paket alle für die Erzeugung einer Zertifikatsautorität und das Signieren von Zertifikaten notwendigen Funktionen. Daher soll hier kurz vorgestellt werden, wie mit dem Kommando `openssl` und `CA.pl` diese Funktionen umgesetzt werden können.

Dieser Exkurs kann kein OpenSSL Handbuch sein und jeden Befehl bis in seine letzte Funktion erläutern. Bei weitergehenden Fragen wird das OpenSSL Handbuch des DFN-CERT empfohlen (*http://www.dfn-pca.de/certify/ssl/handbuch/*)

Das Kommando `CA.pl` erlaubt einen einfacheren Umgang mit der Zertifikatsautorität als das eigentliche `openssl` Kommando. Das Kommando `CA.pl` ist bei den Distributionen in unterschiedlichen Paketen zu finden. Bei Red Hat Linux befindet es sich im Paket *openssl-perl*. Stellen Sie sicher, dass Sie das richtige Paket installiert haben.

Die Zertifikatsautorität (Certificate Authority, CA) sollte in einem eigenen Verzeichnis untergebracht werden. Aus Sicherheitsgründen sollte der Rechner, auf dem die CA erzeugt wird, keine Verbindung zum Netzwerk aufweisen. Wenn dies nicht möglich ist, sollte überlegt werden, ob die CA auf einer Diskette, einer Zip-Disk oder einem USB Memory Stick angelegt wird. Dieses Medium kann dann bei Nichtgebrauch entfernt und so sicher gelagert werden.

Zunächst sollte das Verzeichnis erzeugt werden und die OpenSSL Konfigurationsdatei (Red Hat Linux 8.0: `/usr/share/ssl/openssl.cnf`) und das Perl Skript `CA.pl` hineinkopiert werden.

```
# mkdir /ca
# cp /usr/share/ssl/openssl.cnf /ca
# cp /usr/share/ssl/misc/CA.pl /ca
```

Listing 5.41 Vorbereitung des CA Verzeichnisses

Anschließend ist es sinnvoll, die Konfigurationsdatei zu editieren. Modifizieren Sie die folgenden Einträge entsprechend Ihren Wünschen.

```
default_days                    = 365   # how long to certify
                                          for
default_crl_days                = 30    # how long before next
                                          CRL
countryName_default             = DE
stateOrProvinceName_default     = NRW
localityName_default            = Steinfurt
0.organizationName_default      = Spenneberg.com
organizationalUnitName_default  = Wireless-VPN
```

Listing 5.42 Modifikation der Konfigurationsdatei

Nun kann die CA erzeugt werden. Dies ist sehr einfach mit dem Kommando `CA.pl` möglich. Hierbei wird automatisch auch die richtige Verzeichnisstruktur angelegt. Dazu ist der Aufruf `./CA.pl -newca` erforderlich. Hierbei muss jedoch mit der Variablen `OPENSSL_CONF` der Ort der angepassten Konfigurationsdatei mitgeteilt werden.

```
# export OPENSSL_CONF=./openssl.cnf
# ./CA.pl -newca
CA certificate filename (or enter to create)<enter>

Making CA certificate ...
Using configuration from ./openssl.cnf
Generating a 1024 bit RSA private key
.............++++++
.........................................................
...........++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:cakenn
Verifying password - Enter PEM pass phrase:cakenn
-----
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:<enter>
State or Province Name (full name) [NRW]:<enter>
Locality Name (eg, city) [Steinfurt]:<enter>
Organization Name (eg, company) [Spenneberg.com]:<enter>
Organizational Unit Name (eg, section) [Wireless-VPN]:<enter>
Common Name (eg, your name or your server's hostname) []:RootCA (c)
2003 Ralf Spenneberg
Email Address []:ralf.spenneberg@mut.de
```

```
# ls demoCA/
cacert.pem  certs  crl  index.txt  newcerts  private  serial
```

Listing 5.43 Erzeugung der CRL

Nun wurde ein Verzeichnis `demoCA` angelegt, in dem die Schlüssel der CA und die Datenbank für die Zertifikate gespeichert werden. Das Unterverzeichnis `certs` enthält später die signierten Zeritifikate, `crl` die Rückruflisten (Certificate Revocation Lists), `newcerts` enthält zu unterzeichnende Zertifikate und `private` den privaten Schlüssel der CA. Die Datei `cacert.pem` enthält das selbstsignierte Zertifikat der CA. Die Dateien `index.txt` und `serial` dienen zur Verwaltung der Zertifikatsdatenbank.

Nun sollte zunächst die Lebensdauer des RootCA Zertifikates verlängert werden. Dieses Zertifikat wurde lediglich für 365 Tage ausgestellt. Um es auf 3650 Tage, also zehn Jahre, zu verlängern ist ein Aufruf von `openssl` erforderlich

```
# cd demoCA
# openssl x509 -in cacert.pem -days 3650 -out cacert2.pem -signkey \
> ./private/cakey.pem
Getting Private key
Enter PEM pass phrase:cakenn
# mv cacert2.pem cacert.pem
# cd ..
```

Listing 5.44 Verlängerung des RootCA Zertifikates

Jetzt ist die Zertifikatsautorität vorbereitet und kann endlich genutzt werden um Zertifikatsanfragen zu erzeugen und diese zu signieren.

Die Zertifikatsanfrage kann mit dem Werkzeug `CA.pl` erzeugt werden. Hierzu ist es erforderlich, dass Kommando mit der Option `-newreq` für *New Request* aufzurufen.

```
# ./CA.pl -newreq
Using configuration from ./openssl.cnf
Generating a 1024 bit RSA private key
............++++++
..............++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:certkenn
Verifying password - Enter PEM pass phrase:certkenn
-----
```

```
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [NRW]:
Locality Name (eg, city) [Steinfurt]:
Organization Name (eg, company) [Spenneberg.com]:
Organizational Unit Name (eg, section) [Wireless-VPN]:
Common Name (eg, your name or your server's hostname) []:NewYork
Email Address []:ralf@spenneberg.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:<enter>
An optional company name []:<enter>
Request (and private key) is in newreq.pem
```

Listing 5.45 Erzeugung der CRL

Diese Signaturanfrage wurde in der Datei `newreq.pem` gespeichert und kann nun signiert werden. Der zu dieser Anfrage passende private Schlüssel wurde ebenfalls in dieser Datei gespeichert, jedoch vorher mit dem eingegebenen Kennwort verschlüsselt. Die Signatur wird aufgerufen mit dem Befehl `./CA.pl -sign`.

```
# ./CA.pl -sign
Using configuration from ./openssl.cnf
Enter PEM pass phrase:cakenn
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'NRW'
localityName          :PRINTABLE:'Steinfurt'
organizationName      :PRINTABLE:'Spenneberg.com'
organizationalUnitName:PRINTABLE:'Wireless-VPN'
commonName            :PRINTABLE:'NewYork'
emailAddress          :IA5STRING:'ralf@spenneberg.net'
Certificate is to be certified until Feb 12 18:32:06 2004 GMT (365
days)
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
```

Listing 5.46 Erzeugung der CRL

Anschließend sollten die beiden Dateien `newreq.pem` und `newcert.pem` sinnvoll umbenannt werden.

```
# mv newreq.pem newyork_req.pem
# mv newcert.pem newyork_cert.pem
```

Dieser Vorgang kann nun für jeden Rechner wiederholt werden, für den Zertifikate benötigt werden.

Zusätzlich sollte nun jedoch noch die Widerrufliste (Certificate Revocation List, CRL) erzeugt werden. Diese Listen enthalten Zertifikate, die bereits während ihrer Gültigkeitsdauer nicht mehr verwendet werden dürfen, weil sie zum Beispiel verloren gingen oder in falsche Hände gerieten. Diese Rückrufliste hat eine Lebensdauer von 30 Tagen. Der Wert wird in der OpenSSL Konfigurationsdatei vorgegeben. Sie sollte daher alle 30 Tage neu erzeugt werden und auf die FreeS/WAN Gateways kopiert werden. FreeS/WAN 1.99 unterstützt noch nicht die automatische Abholung der CRL von einem Server. Diese Unterstützung befindet sich bereits in FreeS/WAN 2.00 mit x509 Patch. Bei Verwendung von FreeS/WAN 1.99 kann dieses Verhalten mit einem Cronjob relativ einfach nachgestellt werden.

```
# openssl ca -gencrl -out demoCA/crl/crl.pem
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase:cakenn
```

Listing 5.47 Erzeugung der CRL

Die so erzeugte RootCA, die Zertifikate und die CRL können nun für die Authentifzierung mit FreeS/WAN genutzt werden.

Im Folgenden soll zunächst die Erzeugung der Konfigurationsdatei `ipsec.conf` beschrieben werden. Sie ähnelt der entsprechenden Datei für RSA Schlüssel. Die Definition einer Verbindung in dieser Datei ist in Listing 5.48 dargestellt.

```
conn x509-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftrsasigkey=%cert
        leftid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net"
#       leftcert=certs/newyork_cert.pem
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightrsasigkey=%cert
        rightid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net"
#        rightcert=certs/berlin_cert.pem

        authby=rsasig
        auto=""
```

Listing 5.48 Automatisch verschlüsselte Verbindung mit x509 Zertifikaten

Die angegebene Identifikation (ID) kann aus dem Zertifikat ermittelt werden. Hierzu kann der OpenSSL Befehl genutzt werden.

```
# openssl x509 -in newyork_cert.pem -noout -subject
subject= /C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/OU=Wireless-VPN/
CN=NewYork/Email=ralf@spenneberg.net
```

Der so erhaltene Distinguished Name (DN) kann in der Datei als `leftid` und `rightid` eingegeben werden. Wurde bei der Erzeugung des Zertifikates ein `subjectAltName` definiert, so kann dieser auch hier angegeben werden, zum Bespiel `leftid=ralf@spenneberg.net`

Nun benötigt FreeS/WAN die entsprechenden Zertifikate und Schlüssel an den richtigen Stellen. Hierzu wird ein Verzeichnis `/etc/ipsec.d` mit den Unterverzeichnissen `private`, `certs`, `cacerts` und `crls` benötigt. Diese Verzeichnisstruktur wird bei der Installation automatisch erzeugt. Die folgenden Dateien müssen nun hier hinterlegt werden:

- **cacert.pem** Das Zertifikat der Root-CA muss in das Verzeichnis `/etc/ipsec.d/cacerts` kopiert werden.
- **crl.pem** Die Widerrufliste (CRL) der RootCA muss in das Verzeichnis `/etc/ipsec.d/crls` kopiert werden.
- **newyork_cert.pem** Das Zertifikat des Gateways New York muss auf diesem Gateway in das Verzeichnis `/etc/ipsec.d/certs` kopiert werden. Entsprechend ist auf Berlin die Datei `berlin_cert.pem` zu kopieren. Zu-

sätzlich ist es erforderlich, mit der Direktive `leftcert=certs/newyork_cert.pem` den Ort bekannt zu geben. Die entgültige Datei kann daher nicht einfach kopiert werden.

- **newyork_req.pem** Der Schüssel des Gateways New York muss auf diesem Gateway in das Verzeichnis `/etc/ipsec.d/private` kopiert werden. Entsprechend ist auf Berlin die Datei `berlin_req.pem` zu kopieren.

Die Tunneldefinition aus Beispiel 5.48 enthält sowohl den Parameter `leftcert=` als auch den Parameter `rightcert=`. Beide sind in diesem Beispiel auskommentiert. Bitte aktivieren Sie nur den Parameter `leftcert` auf dem linken Gateway (NewYork) und `rightcert` auf dem rechten Gateway (Berlin). Ansonsten werden die Zertifikate nicht über das Netzwerk übertragen, sondern die lokalen Dateien verwendet.

Wichtig bei dieser Tunneldefinition ist auch die Definition des Parameters `authby=rsasig`. Es handelt sich bei X.509 Zertifikaten um öffentliche RSA Schlüssel. Wenn dieser Parameter nicht angegeben wird, verwendet FreeS/WAN standardmäßig PSKs aus Gründen der Abwärtskompatibilität zu alten Versionen. Diese Option ist ab FreeS/WAN 2.0 als Standardwert auf `rsasig` gesetzt.

Nun sind auf beiden Rechnern noch die Dateien `/etc/ipsec.secrets` zu erzeugen. In älteren Versionen von FreeS/WAN war es erforderlich, den privaten RSA Schlüssel aus der Datei `gateway_req.pem` zu extrahieren und in dieser Datei einzutragen. Hierfür wurde das Werkzeug `fswcert` zur Verfügung gestellt.

Dies ist nun nicht mehr erforderlich. Seit der Version 0.9.8 des X.509 Patches ist FreeS/WAN in der Lage die PEM Datei direkt zu lesen. Dafür muss der Name der PEM Datei mit dem privaten Schlüssel in `/etc/ipsec.secrets` angegeben werden.

Wenn die PEM Datei bei der Erzeugung mit einer Passphrase geschützt wurde, ist es erforderlich sie in der Datei `/etc/ipsec.secrets` anzugeben. Listing 5.49 zeigt die fertige Datei.

```
: RSA newyork_req.pem \}certkenn\{
```

Listing 5.49 Die Datei `ipsec.secrets` bei x509 Zertifikaten

Wurde die Datei nicht mit einem Kennwort geschützt, so unterbleibt auch die Angabe des Kennwortes in der Datei.

TIPP

Ab Version 1.4.5 des X.509 Patches besteht die Möglichkeit, die Passphrase bei Bedarf vom Benutzer anzufordern. Hierzu wird die folgende Syntax verwendet:

```
: RSA newyork_req.pem %prompt
```

Soll die Passphrase sofort angefordert werden, kann der Befehl `ipsec auto –rereadsecrets` verwendet werden.

Nun kann der Tunnel aufgebaut werden.

Aufbau des Tunnels

Um nun den X.509 authentifizierten Tunnel aufzubauen, ist es notwendig, dass beide Gateways über identische Tunnelkonfigurationen verfügen (bis auf die Angabe `leftcert=/rightcert=`). Zusätzlich müssen beide Gateways der gleichen CA vertrauen. Dies erfolgt durch die Kopie des Zertifikates der RootCA in das Verzeichnis `/etc/ipsec.d/cacerts`. FreeS/WAN vertraut allen CAs, deren Zertifikate in diesem Verzeichnis abgespeichert werden. Die Zertifikate der Rechner (newyork_cert.pem und berlin_cert.pem) müssen nicht ausgetauscht werden. Sie werden von FreeS/WAN automatisch übertragen.

Wurden so die Vorbereitungen getroffen, kann FreeS/WAN gestartet werden. Anschließend sollte in der Protokolldatei `/var/log/secure` der erfolgreiche Start überprüft werden. Die richtige Protokolldatei kann in Abhängigkeit der Linux Distribution unterschiedliche Namen aufweisen. Bei Red Hat ist es die Datei `/var/log/secure`.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
# tail -13 /var/log/secure
Feb 12 21:18:00 left ipsec__plutorun: Starting Pluto subsystem...
Feb 12 21:18:01 left pluto[566]: Starting Pluto (FreeS/WAN Version 1.99)
Feb 12 21:18:01 left pluto[566]: including X.509 patch with traffic
                                 selectors (Version 0.9.21)
Feb 12 21:18:01 left pluto[566]: Changing to directory
                                 '/etc/ipsec.d/cacerts'
Feb 12 21:18:01 left pluto[566]: loaded cacert file 'cacert.pem'
                                 (1419 bytes)
Feb 12 21:18:01 left pluto[566]: Changing to directory '/etc/ipsec.d/crls'
Feb 12 21:18:01 left pluto[566]: loaded crl file 'crl.pem' (560 bytes)
Feb 12 21:18:01 left pluto[566]: could not open my default X.509 cert
                                 file '/etc/x509cert.der'
Feb 12 21:18:01 left pluto[566]: OpenPGP certificate file
                                 '/etc/pgpcert.pgp' not found
```

```
Feb 12 21:18:03 left pluto[566]: listening for IKE messages
Feb 12 21:18:03 left pluto[566]: adding interface ipsec0/eth0 3.0.0.1
Feb 12 21:18:03 left pluto[566]: loading secrets from "/etc/ipsec.secrets"
Feb 12 21:18:03 left pluto[566]: loaded private key file '/etc/ipsec.d/
                                 private/newyork_req.pem' (1675 bytes)
```

Listing 5.50 Start von FreeS/WAN und Kontrolle in der Datei `/var/log/secure`

Das Protokoll zeigt, dass FreeS/WAN mit X.509 Patch gestartet wurde. Anschließend wurden erfolgreich das RootCA Zertifikat `/etc/ipsec.d/cacerts/cacert.pem` und die CRL `/etc/ipsec.d/crls/crl.pem` geladen. Die Dateien `/etc/x509cert.der` und `/etc/pgpcet.pgp` wurden nicht gefunden. Dies ist kein Fehler. In früheren Versionen musste das Rechnerzertifikat in der Datei `x509cert.der` im DER Format abgespeichert werden. Zusätzlich unterstützt der X.509-Patch auch PGP Schlüssel für die Authentifizierung. Sie müssten in der Datei `/etc/pgpcert.pgp` abgelegt werden. Die letzten beiden Zeilen der Protokolldatei bestätigen das erfolgreiche Laden des privaten Schlüssels aus der Datei `/etc/ipsec.d/private/newyork_req.pem`.

Das erfolgreiche Laden der Zertifikate kann auch mit dem folgenden Befehl überprüft werden.

```
# ipsec auto --listall
000
000 List of Public Keys:
000
000

000 List of User/Host Certificates:
000
000
000 List of CA Certificates:
000
000 Feb 12 21:32:01 2003, count: 1
000        subject: 'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
           OU=Wireless-VPN, CN=RootCA (c) 2003 Ralf Spenneberg,
           E=ralf.spenneberg@mut.de'
000        issuer:  'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
           OU=Wireless-VPN, CN=RootCA (c) 2003 Ralf Spenneberg,
           E=ralf.spenneberg@mut.de'
000        pubkey:   1024 RSA Key AwEAAb71S
000        validity: not before Feb 12 18:58:29 2003 ok
000                  not after  Feb 09 18:58:29 2013 ok
000
000 List of CRLs:
000
000 Feb 12 21:32:01 2003, revoked certs: 0
```

```
000        issuer:  'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
           OU=Wireless-VPN, CN=RootCA (c) 2003 Ralf Spenneberg,
           E=ralf.spenneberg@mut.de'
000        updates:  this Feb 12 21:10:33 2003
000                  next Mar 14 21:10:33 2003 ok
```

Listing 5.51 Anzeige der Schlüssel und Zertifikate

Sobald die Verbindung x509-newyork-berlin geladen und gestartet wurde, können die entsprechenden Schlüssel und Zertifikate ebenfalls angezeigt werden. Dies sollte auch dann dementsprechend überprüft werden.

```
# ipsec auto --add --verbose x509-newyork-berlin
002 loaded host cert file '/etc/ipsec.d/certs/berlin_cert.pem' (3804 bytes)
002 added connection description "x509-newyork-berlin"
# ipsec auto --up --verbose x509-newyork-berlin
002 "x509-newyork-berlin" #1: initiating Main Mode
104 "x509-newyork-berlin" #1: STATE_MAIN_I1: initiate
106 "x509-newyork-berlin" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "x509-newyork-berlin" #1: STATE_MAIN_I3: sent MI3, expecting MR3
002 "x509-newyork-berlin" #1: Peer ID is ID_DER_ASN1_DN: 'C=DE, ST=NRW,
                            L=Steinfurt, O=Spenneberg.com,
                            OU=Wireless-VPN, CN=NewYork,
                            E=ralf@spenneberg.net'
002 "x509-newyork-berlin" #1: ISAKMP SA established
004 "x509-newyork-berlin" #1: STATE_MAIN_I4: ISAKMP SA established
002 "x509-newyork-berlin" #2: initiating Quick Mode RSASIG+ENCRYPT+
                            TUNNEL+PFS+DISABLEARRIVALCHECK
112 "x509-newyork-berlin" #2: STATE_QUICK_I1: initiate
002 "x509-newyork-berlin" #2: sent QI2, IPsec SA established
004 "x509-newyork-berlin" #2: STATE_QUICK_I2: sent QI2, IPsec SA
                            established
# ipsec auto --listcerts
000
000 List of User/Host Certificates:
000
000 Feb 12 21:58:53 2003, count: 1
000        subject:  'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
                     OU=Wireless-VPN, CN=NewYork, E=ralf@spenneberg.net'
000        issuer:   'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
                     OU=Wireless-VPN, CN=RootCA (c) 2003 Ralf Spenneberg,
                     E=ralf.spenneberg@mut.de'
000        pubkey:   1024 RSA Key AwEAAcSOW, has private key
000        validity: not before Feb 12 20:18:01 2003 ok
000                  not after  Feb 12 20:18:01 2004 ok
# ipsec auto --listpubkeys
000
000 List of Public Keys:
000
```

```
000 Feb 12 21:59:12 2003, 1024 RSA Key AwEAAcll8, until Feb 12 20:15:06
2004 ok
000        ID_DER_ASN1_DN 'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
           OU=Wireless-VPN, CN=Berlin, E=ralf@spenneberg.net'
000 Feb 12 21:58:53 2003, 1024 RSA Key AwEAAcSOW, until
    Feb 12 20:18:01 2004 ok
000        ID_DER_ASN1_DN 'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
           OU=Wireless-VPN, CN=NewYork, E=ralf@spenneberg.net'
```

Die Verbindung wurde erfolgreich aufgebaut. Anschließend zeigt `ipsec auto –listcerts` die eigenen geladenen Zertifikate und der Befehl `ipsec auto –listpubkeys` die über das Netz geladenen und durch die RootCA zertifizierten öffentlichen Schlüssel an.

Der erfolgreiche Aufbau und die Verschlüsselung der Verbindung kann mit einem Ping getestet werden. Der unverschlüsselte Pingverkehr wurde bereits mehrfach gezeigt. Hier sollen daher nur zwei Pakete als Ausgabe des Kommandos `tcpdump` dargestellt werden.

```
# tcpdump -ni eth0
tcpdump: listening on eth0
22:29:19.731295 arp who-has 3.0.0.1 tell 3.255.255.254
22:29:19.731740 arp reply 3.0.0.1 is-at fe:fd:0:0:0:0
22:29:19.731749 5.0.0.1 > 3.0.0.1: ESP(spi=0x6dd02e94,seq=0x1)
22:29:19.732494 3.0.0.1 > 5.0.0.1: ESP(spi=0xce9b108a,seq=0x1)
```

Verbesserungen und Erweiterungen

Befindet sich nun hinter jedem Gateway ein weiteres Netzwerk, und sollen diese über das VPN miteinander kommunizieren, so ist eine weitere Tunneldefinition erforderlich. Außerdem können einige, in beiden Tunneldefinitionen benutzte Parameter, in einer `conn %default` Verbindung zusammengefasst werden. Die fertige `ipsec.conf` ist in Listing 5.52 dargestellt.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        # Die beiden folgenden Parameter werden von 2.0 nicht unterstützt
        plutoload=%search
        plutostart=%search

conn %default
        authby=rsasig
        leftrsasigkey=%cert
```

```
        rightrsasigkey=%cert
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftcert=certs/newyork_cert.pem
        leftid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net"
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=Berlin/Email=ralf@spenneberg.net"
        keyingtries=0

conn x509-newyork-berlin
        auto="start"

conn x509-newyorknet-berlinnet
        leftsubnet=10.0.1.0/24
   rightsubnet=10.0.2.0/24
        auto="start"
```

Listing 5.52 Komplette Datei `/etc/ipsec.conf`

Diese Datei wurde nun sehr stark umgestellt. Es wurde eine zweite Verbindung definiert, die die Parameter `leftsubnet` und `rightsubnet` nutzt. Hiermit wird ein Tunnel erzeugt, der diesen beiden Netzwerken den Austausch von Informationen erlaubt. Für jede Form eines Tunnels (Gateway-Gateway, Subnetz-Subnetz, Gateway-Subnetz, Subnetz-Gateway) ist ein eigener Tunnel erforderlich. Das stellt jedoch kein Problem dar, da ein Großteil der für die Definition des Tunnels benötigten Informationen als Standardwert definiert werden können. Hierzu werden die Daten in der `conn %default` eingetragen. Es wäre aber auch genau so gut möglich, die Daten in einer eigenen benannten Verbindung zu definieren, und diese dann mit `also` einzulesen. Diese Methode würde es relativ einfach erlauben mehrere Defaultwerte zu definieren, und so recht einfach auch weitere Tunnel mit anderen FreeS/WAN Gateways aufzubauen. Listing 5.53 zeigt eine derartige Struktur.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        # Die beiden folgenden Zeilen werden von 2.x nicht unterstützt
        plutoload=%search
        plutostart=%search
```

```
conn %default
        authby=rsasig
        leftrsasigkey=%cert
        rightrsasigkey=%cert
        keyingtries=0

conn x509-newyorknet-berlinnet
   also=x509-newyork-berlin
        leftsubnet=10.0.1.0/24
   rightsubnet=10.0.2.0/24

conn x509-newyork-berlin
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftcert=certs/newyork_cert.pem
        leftid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net"
        right=5.0.0.1
        rightnexthop=5.255.255.254
        rightid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=Berlin/Email=ralf@spenneberg.net"
        auto="start"
```

Listing 5.53 Verwendung der `also` Direktive

Wichtig bei der Verwendung des Parameters `also` ist die Tatsache, dass die Verbindung, die mit `also=` geladen wird, später in der Konfigurationsdatei definiert wird.

Fazit

Der Aufbau eines einzelnen Tunnels mit einer x509 zerifikatsgestützten Authentifizierung scheint zunächst einmal komplizierter zu sein, als bei einfachen RSA Schlüsseln. Der Vorteil wird erst dann deutlich, wenn viele VPN Gateways miteinander kommunizieren wollen, oder Schlüssel ausgetauscht werden müssen. Bei der Verwendung von RSA Schlüsseln ist es erforderlich, dass die öffentlichen RSA Schlüssel allen beteiligten FreeS/WAN Gateways bekannt gegeben werden. Sie müssen als `left/rightrsasigkey` in der Konfigurationsdatei angegeben werden. Das ist bei der Verwendung von x509 Zertifikaten nicht der Fall. Die öffentlichen x509 Zertifikate werden von den Clients selbstständig und automatisch über das Netzwerk übertragen. Alle beteiligten VPN Gateways benötigen lediglich ihren eigenen Schlüssel, ihr eigenes Zertifikat und das Zertifikat der CA, die alle weiteren Zertifikate signiert hat. Dann können automatisch alle Gateways sich gegenseitig authentifizieren. Muss der Schlüssel eines Gateways ausgetauscht werden, so muss dieser lediglich neu durch die CA signiert werden. Der neue Schlüssel muss

nicht von Hand auf alle Kommunikationspartner übertragen werden. Geht ein Schlüssel verloren oder gerät ein Laptop in falsche Hände, so kann mit der CRL sehr einfach dieser Schlüssel bereits während seiner Gültigkeitsdauer gesperrt werden. Wenn die CRL Listen von den Gateways automatisch (zum Beispiel per Cronjob) geladen werden, ist keine weitere Konfiguration erforderlich.

Der nächste Abschnitt beschäftigt sich mit der Verwaltung von Roadwarriors. Dabei handelt es sich um Benutzer, die mit einer dynamischen IP Adresse auf ein VPN zugreifen möchten. Wenn hier eine Vielzahl von Benutzern zu verwalten sind, die gleichzeitig zugreifen möchten, wird der Vorteil der x509 Zertifikate noch deutlicher.

Schließlich soll nicht unerwähnt bleiben, dass FreeS/WAN mit seiner Unterstützung reiner RSA Schlüssel für die Authentifizierung eine Außenseiter-Rolle einnimmt. Die meisten anderen IKE Implementierungen unterstützen PSKs und x509 Zertifikate, aber keine RSA Schlüssel. Daher ist die Verwendung von x509 Zertifikaten auch für heterogene VPN Lösungen anzuraten.

5.5.5 Roadwarrior

Der Begriff »Roadwarrior« bezeichnet Personen beziehungsweise Rechner, die mit unbestimmter IP Adresse auf ein VPN Gateway zugreifen wollen. Typischerweise handelt es sich hierbei zum Beispiel um Außendienstmitarbeiter, die von unterwegs Zugriff auf die Datenbanken ihres Mutterunternehmens benötigen. Aber auch alle anderen Konstellationen, bei denen Rechner mit dynamischen IP Adressen eine VPN Verbindung mit einem VPN Gateway aufbauen möchten, sind denkbar. Hierbei ist die Anzahl der Roadwarriors nicht beschränkt. Theoretisch sind mehrere Hundert gleichzeitiger Tunnel möglich.

Mögliche Roadwarrior Szenarien sind:

- Außendienstmitarbeiter, die abends aus dem Hotelzimmer die am Tag gesammelten Aufträge mit der zentralen Datenbank des Unternehmens abgleichen müssen.
- Heimarbeiter mit einem Telearbeitsplatz, die von zu Hause aus ihre Arbeit verrichten. Dazu benötigen sie den Zugriff auf die Daten und Dateien der Firma, Zugang zu ihrem E-Mail Konto und Kommunikationsmöglichkeiten mit ihren Mitarbeitern.
- Abteilungsleiter, die im Urlaub oder am Wochenende ihre E-Mail und den Projektfortschritt überwachen müssen.

- Filialen, die an das zentrale Unternehmensnetz angeschlossen werden sollen, jedoch nicht über eine dauerhafte Standleitung mit fester IP Adresse verfügen. Möglicherweise erfolgt der Verbindungsaufbau nur zu bestimmten Zeiten mit einem einfachen Internet Provider wie zum Beispiel T-Online oder Freenet.

Bei allen vorgestellten Szenarien wird davon ausgegangen, dass ein Endpunkt des Tunnels über eine feste IP Adresse verfügt und ständig erreichbar ist. Der Tunnel wird dann in den vorgestellten Fällen vom Roadwarrior aufgebaut. Es entsteht eine Art Client/Server Struktur, bei der das Unternehmens-VPN-Gateway ständig diesen Dienst anbietet und der Roadwarrior bei Bedarf sich als Client verbindet.

TIPP

Wenn beide Endpunkte über eine dynamische IP Adresse verfügen, ist der Aufbau eines Tunnels wesentlich problematischer. Dies ist zum Beispiel der Fall, wenn zwei Netzwerke, die jeweils über ISDN oder ADSL mit dem Internet verbunden sind, einen VPN Tunnel aufbauen sollen. Hierbei muss nun sichergestellt werden, dass ein Endpunkt die IP Adresse des anderen Endpunktes erhält. Eine Lösung wird im Kapitel 9, »Fortgeschrittene Konfiguration« vorgestellt.

Eine mögliche Testumgebung, in der die Verbindung von Roadwarriors getestet werden kann, ist in Kapitel 11, »Testumgebungen« beschrieben. Hier soll eine zusätzliche Abbildung Klarheit schaffen (Abbildung 5.1). Diese Abbildung wurde gegenüber der Testumgebung II leicht verändert, da in diesem Kapitel nicht über NAT Traversal gesprochen werden soll. Dies ist Thema des Kapitels 9, »Fortgeschrittene Konfiguration«.

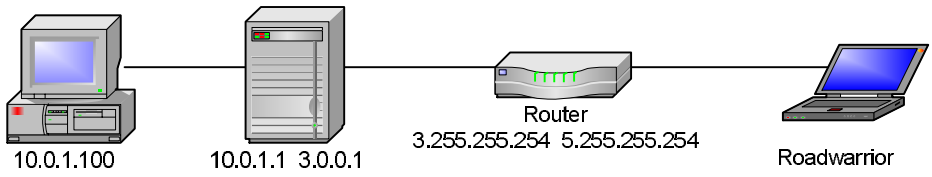


Abbildung 5.1 Ein Roadwarrior Szenario

Im Folgenden werden die FreeS/WAN Konfigurationen für dieses Szenario vorgestellt. Dabei werden separat die Konfiguration für die Authentifizierung mit PSKs, öffentlichen RSA Schlüsseln und x509 Zertifikaten vorgestellt.

Die Diskussion der PSKs erfolgt in diesem Zusammenhang nur der Vollständigkeit halber. Möglicherweise möchten Sie ein VPN aufbauen, bei dem die Clients nur in der Lage sind diese PSKs für die Authentifizierung zu verwenden. Aus Sicherheitsgründen sollte jedoch immer auf RSA Schlüssel oder x509 Zertifikate ausgewichen werden. x509 Zertifikate bieten im Zusammenhang mit Roadwarriors die höchste Sicherheit und den höchsten Komfort bei ihrer Administration.

Wenn es sich um eine größere Zahl von Roadwarriors handelt, die Sie administrieren müssen, ist der Einsatz einer komfortablen Zertifikatsautorität (zumindest komfortabler als das `openssl` Kommando) anzuraten. Hier werden in Kapitel 8, »Aufbau einer Public Key Infrastruktur« die TinyCA und die OpenCA vorgestellt. Die TinyCA ist eine einfache grafische Anwendung, die einer Person die Erzeugung und Signatur von Zertifikaten erlaubt. Bei der OpenCA handelt es sich um eine komplexe PKI mit Unterstützung eines Web und LDAP Servers, mit denen die Zertifikate erzeugt werden und auf denen sie gespeichert werden.

Roadwarrior mit Preshared Key (PSK)

Die Implementierung von Roadwarriors mit Preshared Keys ist eine sehr undankbare Aufgabe mit FreeS/WAN. FreeS/WAN unterstützt von Haus aus nur den Main Mode in der Phase 1 der IKE Verhandlungen. In dieser Phase wird die Authentifizierung der Rechner durchgeführt und eine ISAKMP SA erzeugt. Der Main Mode unterstützt jedoch nur eine einzige PSK Roadwarrior Verbindung, denn bei der Verwendung von Shared Secrets werden lediglich die IP Adressen für die Auswahl des richtigen Schlüssels genutzt. Da die IP Adresse des Roadwarriors unbekannt ist, kann nicht je Roadwarrior ein PSK definiert werden. Alle Roadwarrior müssen daher denselben PSK verwenden.

Dies stellt ein großes Sicherheitsproblem dar. Sobald der PSK in falsche Hände gerät, müssen auf allen Systemen die Schlüssel ausgetauscht werden!

Abhilfe schafft der Aggressive Modus in der Phase 1. Hierbei werden vor Beginn der Verschlüsselung bereits die Identitätsinformationen übertragen. So kann das Gateway, den zur Identifikation passenden Schlüssel auswählen. FreeS/WAN unterstützt diesen Modus nicht. Kommerzielle VPN Lösungen verlangen für einen Roadwarrior mit PSKs meist diesen Modus. Auch wenn FreeS/WAN lediglich als Roadwarrior Client auf diese kommerziellen Systeme mit einer PSK authentifiziert zugreifen soll, wird der Aggressive Modus verlangt.

Es existiert ein Patch, der FreeS/WAN in die Lage versetzt, die Verbindung erfolgreich aufzubauen. Dieser Patch ist momentan jedoch leider nur in der Lage die Client Funktion zur Verfügung zu stellen. Der Aggressive Modus wird noch nicht als VPN Gateway in voller Funktion unterstützt (Stand Super FreeS/WAN 1.99_kb2). Dann kann FreeS/WAN als Roadwarrior eigene Verbindungen mit PSKs aufbauen. Kommerzielle VPN Gateways, die den Aggressive Mode bei PSKs verlangen, sind zum Beispiel Checkpoint VPN-1 und Netscreen.

Dieser Abschnitt wird zunächst die klassische FreeS/WAN Variante vorstellen. Anschließend wird die Konfiguration von FreeS/WAN im Aggressive Mode mit Patch beschrieben. Hierzu muss FreeS/WAN entsprechend gepatcht worden sein. Der Abschnitt 5.3.3, »FreeS/WAN Patches« beschreibt die Anwendung des Patches.

Der wesentliche Unterschied bei einer Roadwarrior Verbindung liegt in der Tatsache, dass das VPN Gateway nicht die IP Adresse des Clients kennt. Daher ist es nicht möglich, die IP Adresse für die Authentifizierung zu verwenden. Das VPN Gateway kann daher die Verbindung auch nicht aufbauen. Auch ein Wiederaufbau der Verbindung nach einem Abbruch schlägt meistens fehl, da der Client sich häufig neu über seinen Internet Provider einwählt und dabei eine andere IP Adresse erhält.

VPN-Gateway

Um diesem Umstand Rechung zu tragen sollte die im Kapitel 10, »Fehlersuche« vorgestellte Konfiguration für PSKs abgeändert werden. Eine entsprechende Konfiguration für das VPN-Gateway ist in Listing 5.54 abgebildet. Die Konfiguration für den Roadwarrior folgt weiter unten.

Wenn Sie noch keine Erfahrung mit dem Aufbau von automatisch verschlüsselten Verbindungen mit PSKs besitzen, sollten Sie zunächst den entsprechenden Abschnitt im letzten Kapitel lesen.

```
config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search

conn %default
        authby=secret
        keyingtries=1
```

```
conn psk-newyork-roadwarrior
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=%any
        auto=add

conn psk-newyorknet-roadwarrior
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        right=%any
        auto=add
```

Listing 5.54 Roadwarrior Konfiguration mit PSKs (VPN Gateway)

Die neu hinzugekommenen Parameter sollen hier kurz erklärt werden. Zunächst ist der Wert `keyingtries` auf 1 gesetzt worden. Dies ist erforderlich, damit das VPN Gateway bei Abbruch der Verbindung nicht selbst versucht, die Verbindung wiederherzustellen. Da der Partner möglicherweise eine neue IP Adresse verwendet, sollte dieser die Verbindung auch bei Abbruch aufbauen. Aus diesem Grund werden die Verbindungen auf den VPN Gateway auch nur mit `auto=add` geladen und nicht sofort gestartet. Da das VPN Gateway die IP Adresse des Partners nicht kennt, kann diese Verbindung nicht gestartet werden.

Schließlich bleibt noch die Angabe `right=%any`. Hiermit wird sichergestellt, dass das VPN Gateway von jeder Quell IP Adresse Anfragen entgegen nimmt. So ist es möglich mit jeder beliebigen Absenderadresse bei Kenntnis des Kennwortes eine Verbindung aufzubauen.

Das Kennwort wird in der Datei `ipsec.secrets` gespeichert. Sie muss für den Roadwarrior leicht angepasst werden.

```
3.0.0.1 %any : PSK
"0xe10bd52b0529b54aac97db63462850f354abdd885c19ea6d65d6f4e
faeab0222"
```

Listing 5.55 VPN Gateway mit PSKs: `ipsec.secrets`

Auch hier muss berücksichtigt werden, dass die IP Adresse des Roadwarriors zum Zeitpunkt der Konfiguration noch nicht bekannt ist. Daher wird hier als Platzhalter `%any` eingetragen.

Nun kann FreeS/WAN auf dem VPN Gateway gestartet werden und anschließend geprüft werden, ob die entsprechenden Tunneldefinitionen geladen wurden.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
# ipsec auto --status
000 interface ipsec0/eth0 3.0.0.1
000
000 "psk-newyork-roadwarrior": 3.0.0.1---3.255.255.254...%any
000 "psk-newyork-roadwarrior":   ike_life: 3600s; ipsec_life: 28800s;
rekey_margin: 540s; rekey_fuzz: 100%; keyingtries: 1
000 "psk-newyork-roadwarrior":   policy:
PSK+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK; interface: eth0; unrouted
000 "psk-newyork-roadwarrior":   newest ISAKMP SA: #0; newest IPsec SA:
#0; eroute owner: #0
000 "psk-newyorknet-roadwarrior":
10.0.1.0/24===3.0.0.1---3.255.255.254...%any
000 "psk-newyorknet-roadwarrior":   ike_life: 3600s; ipsec_life: 28800s;
rekey_margin: 540s; rekey_fuzz: 100%; keyingtries: 1
000 "psk-newyorknet-roadwarrior":   policy:
PSK+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK; interface: eth0; unrouted
000 "psk-newyorknet-roadwarrior":   newest ISAKMP SA: #0; newest IPsec SA:
#0; eroute owner: #0
000
000
```

Listing 5.56 Start von FreeS/WAN auf dem VPN Gateway

Die Ausgabe von `ipsec auto –status` zeigt an, dass das VPN Gateway NewYork zwei Tunnel anbietet. Der Tunnel psk-newyork-roadwarrior ist konfiguriert für die IP Adressen `3.0.0.1—3.255.255.254...%any`. Bisher wurden für diesen Tunnel keine ISAKMP SA (0) und keine IPsec SA ausgehandelt. Ebenso wird ein Tunnel psk-newyorknet-roadwarrior angeboten, der für die IP Adressen `10.0.1.0/24===3.0.0.1—3.255.255.254... %any` konfiguriert wurde. Diese Angaben lassen sich lesen als ein Tunnel von 10.0.1.0/24 über das VPN Gateway 3.0.0.1 und den Router 3.255.255.254 zu einer beliebigen (`%any`) IP Adresse.

Roadwarrior

Auch die Konfiguration des Roadwarriors muss nun angepasst werden. Siehe Listing 5.57:

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

```
conn %default
        authby=secret
        keyingtries=0

conn psk-newyork-roadwarrior
        left=3.0.0.1
        leftnexthop=3.255.255.254
        right=%defaultroute
        auto=start

conn psk-newyorknet-roadwarrior
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        right=%defaultroute
        auto=start
```

Listing 5.57 Roadwarrior mit PSKs (Roadwarrior)

Die wesentlichen Änderungen beim Roadwarrior betreffen die Parameter `interfaces` und `right`. Hier wurde bei beiden Parametern der Wert `%defaultroute` angegeben. Dadurch wird Pluto bei seinem Start aufgefordert die korrekte Netzwerkkarte, die eigene IP Adresse und das Standardgateway selbst zu erkennen und entsprechend einzutragen. Dies ist erforderlich, da ein Roadwarrior möglicherweise über ein Modem, eine ISDN Verbindung oder über ein lokales LAN die Verbindung aufbaut. Hierbei werden dem Roadwarrior unterschiedliche, vorher unbekannte IP Adressen und Standard Gateways zugewiesen. Durch Angabe des Parameters `%defaultroute` ist es nicht erforderlich die Konfigurationsdatei von Hand anzupassen.

ACHTUNG

Der Parameter `right=%defaultroute` wird übrigens nur dann ausgewertet, wenn auch `interfaces=%defaultroute` gesetzt ist.

Die Datei `ipsec secrets` muss auch den veränderten Verhältnissen angepasst werden. Da nun die IP Adresse des Roadwarriors nicht mehr bekannt ist, kann sie auch in dieser Datei nicht angegeben werden. Daher wird hier folgende Konfiguration verwendet:

```
: PSK "0xe10bd52b0529b54aac97db63462850f354abdd885c19ea6d65d6f4efaeab0222"
```

Die Konfiguration in Datei `ipsec.secrets` stellt sicher, dass der PSK für alle Verbindungen vom Roadwarrior genutzt wird.

Da lediglich der Roadwarrior in der Lage ist die Verbindung aufzubauen (nur er kennt den anderen Tunnelendpunkt) wurden beide Verbindungen mit dem Parameter `auto=start` ausgestattet. So wird Pluto direkt bei seinem Start die Verbindung aufbauen. Hierzu passt dann auch die Angabe `keyingtries=0`. Pluto soll unaufhörlich versuchen die Verbindung aufzubauen und bei einem Abbruch derselben sie sofort wieder herstellen.

Wird nun FreeS/WAN auf dem Roadwarrior gestartet, wird dieser direkt beide Verbindungen aufbauen.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
# ipsec eroute
0       5.0.1.199/32    -> 3.0.0.1/32    => tun0x1004@3.0.0.1
0       5.0.1.199/32    -> 10.0.1.0/24   => tun0x1002@3.0.0.1
```

Listing 5.58 Start der Verbindungen auf dem Roadwarrior

Die Ausgabe von `ipsec eroute` zeigt an, dass zwei Tunnel von 5.0.1.199 (dem Roadwarrior) sowohl zu 3.0.0.1 (VPN Gateway NewYork) und 10.0.1.0/24 (Subnetz hinter NewYork) aufgebaut wurde.

Roadwarrior in Aggressive Modus

Wie bereits mehrfach erwähnt, bietet der Main Modus in Phase 1 des IKE Protokolls nicht die Möglichkeit bei einem Roadwarrior mit PSK die Identifikation mit zu übertragen. Daher müssen alle Roadwarriors bei der Verwendung von PSKs dieselbe PSK verwenden. Es besteht nicht die Möglichkeit die Roadwarriors bereits bei der Authentifizierung zu unterscheiden.

Der Aggressive Modus der Phase 1 bietet diese Möglichkeit. Dazu überträgt er die Identifikation des Roadwarriors vor der Authentifizierung und Verschlüsselung der Verbindung im Klartext an das VPN Gateway. (**Achtung, die Information wird im Klartext gesendet!**). FreeS/WAN unterstützt nur mit einem Patch diesen Aggressive Modus. Der Patch unterstützt dann auch nur den Aggressive Modus als VPN Client. FreeS/WAN kann diesen Aggressive Modus nicht als VPN Gateway nutzen (Stand super-freeswan-1.99_kb2-aggrmode.patch). Der `isakmpd` und der `racoon` können dies.

Dennoch kann diese Funktion genutzt werden, wenn FreeS/WAN als Client auf kommerzielle VPN Gateways zugreifen soll, die die Authentifizierung mit PSKs durchführen und den Aggressive Modus verlangen. Im Folgenden soll kurz die Konfiguration erläutert (Listing 5.59) werden. Ein Test dieser Funktion ist nur möglich, wenn Sie über ein Checkpoint VPN-1, Netscreen oder ähnliches VPN Gateway verfügen, welches dementsprechend konfiguriert wurde.

```
conn aggrmode-roadwarrior
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftsubnet=10.0.1.0/24
        authby=secret
        right=%defaultroute
        rightid=meine@id
        aggrmode=yes
        auto=start
```

Listing 5.59 FreeS/WAN mit Aggressive Mode

Wichtig bei der Definition des Tunnels ist die Angabe `aggrmode`. Diese Funktion wird durch den Patch bereitgestellt. Zusätzlich muss nun die Identifikation als `rightid` angegeben werden.[2] Damit nun FreeS/WAN auch den richtigen PSK für die Verbindung findet, muss dieser in der Datei `ipsec.secrets` eingetragen werden. Das folgende Beispiel zeigt einen derartigen Eintrag:

```
meine@id 3.0.0.1: "Die von meine@id gewählte PSK bei dem VPN Gateway"
```

Mit dieser Konfiguration ist es nun möglich mit FreeS/WAN den Aggressive Mode als Client zu unterstützen.

Fazit

Der Aufbau von Roadwarriorverbindungen mit Preshared Keys ist grundsätzlich möglich. Hierbei ist es unerheblich, wieviele Roadwarriors sich mit dem VPN Gateway verbinden sollen, denn durch eine Einschränkung des Main Modes müssen alle Roadwarriors, die die Authentifizierung mit einem PSK durchführen, denselben PSK nutzen. Dies ist vom Standpunkt der Sicherheit her kritisch. Die Verwendung des Aggressive Mode schafft hier zwar Abhilfe, jedoch wird dieser Modus von FreeS/WAN nur nach einem Patch und nur als Client unterstützt.

Daher sollten Roadwarriorszenarien immer mit RSA Schlüsseln oder x509 Zertifikaten aufgebaut werden.

Roadwarrior mit RSA Schlüsseln

Eine Roadwarriorkonfiguration, die für die Authentifizierung öffentliche RSA Schlüssel einsetzt, weist eine wesentlich höhere Sicherheit auf als PSK basierte Roadwarriorkonfigurationen. Jedoch ist zunächst der Administrationsaufwand um einiges höher. Hier sollen aber ein paar Tipps und Kniffe vorgestellt werden, die die Konfiguration stark vereinfachen. Zunächst wird

2. In dieser Konfiguration ist `left` das VPN Gateway. Die Wahl `left` oder `right` ist jedoch frei.

wieder die Konfiguration des VPN Gateways für einen Roadwarrior vorgestellt. Anschließend erfolgt die Konfiguration des Roadwarriors. Abschließend wird dann die Konfiguration eines VPN Gateways für mehrere verschiedene Roadwarrior dargestellt. Hier sind dann einige wesentliche Vereinfachungen möglich.

VPN Gateway

Der Aufbau eines VPN Gateways, das öffentliche Schlüssel für die Authentifizierung einsetzt, ist relativ einfach. Im Grunde kann die normale Konfiguration aus Listing 5.37 fast unverändert übernommen werden. Es muss lediglich berücksichtigt werden, dass das andere Ende des Tunnels zum Zeitpunkt der Konfiguration unbekannt ist. Die resultierende Konfigurationsdatei für das VPN Gateway ist in Listing 5.60 abgebildet. Dabei wurden zur erhöhten Lesbarkeit einige Parameter umgestellt.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces=\}ipsec0=eth0\{
        klipsdebug=none
        plutodebug=none
        # Die beiden folgenden Zeilen werden von 2.x nicht
          unterstützt
        plutoload=%search
        plutostart=%search

conn %default
        authby=rsasig
        keyingtries=1
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftrsasigkey=0sAQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKi
        NrLgyNfBo9npxZdOmolWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2
        DCPO5yS2OEa6HrsOjti+tyIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+HjJGb
        cpPEBLNHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qdbu
        ho9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBHqX69
        oQVsncnuFderIKp7zrP4HUMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH5h3iG/
        EDUofM19TjqJR7L4x22H0sdV6txoRC4R0LT
        right=%any

conn rsa-newyork-rwberlin
        rightid=roadwarrior@berlin
        rightrsasigkey=0sAQOLr0P0yVDh3SETo7p2J+RC6PvbnR7nI85QKVUlfkJG
        ODbJF6ZhrxdFWAjx6gF0oh9BM9BpJoSxJdkwxMTW00QvGSZ9OSPxogb
        7m52RqJwfYaQVka6xjf7f/6xXIful4dWfUFOlaemZf8L+uox5RT7RR8ZU7
        w8uDqV38bbh6Odo3hoBDFkdZ4yplW+HKqScqTVtTZgSPcT7rhSt6XqUbB
```

```
        dUMgpTNEeMNKNxUyE6Gabl+ejpULlN7KSebRti6o6bUU1aiqahk64oAPL03
        B1ivqqDSB+iMVSJM1NnltwThlZ0FNobbTXXL3BaccQrUDbQxm/DCxn1SF
        N64i633/Q384+uxI92rQ8hCxLJL/YPVTSwc9S/
        auto=add

conn rsa-newyorknet-rwberlin
        rightid=roadwarrior@berlin
        leftsubnet=10.0.1.0/24
        rightrsasigkey=0sAQOLr0P0yVDh3SETo7p2J+RC6PvbnR7nI85QKVUlfkJG
        ODbJF6ZhrxdFWAjx6gF0oh9BM9BpJoSxJdkwxMTW00QvGSZ9OSPxogb
        7m52RqJwfYaQVka6xjf7f/6xXIful4dWfUFOlaemZf8L+uox5RT7RR8ZU7
        w8uDqV38bbh6Odo3hoBDFkdZ4yplW+HKqScqTVtTZgSPcT7rhSt6XqUbB
        dUMgpTNEeMNKNxUyE6Gabl+ejpULlN7KSebRti6o6bUU1aiqahk64oAPL03
        B1ivqqDSB+iMVSJM1NnltwThlZ0FNobbTXXL3BaccQrUDbQxm/DCxn1SF
        N64i633/Q384+uxI92rQ8hCxLJL/YPVTSwc9S/
        auto=add
```

Listing 5.60 Roadwarrior Konfiguration mit RSA Schlüsseln (VPN Gateway)

Im Folgenden soll nun diese Datei erläutert werden. Zunächst wurden die Daten, die lediglich die linke Seite (NewYork, VPN Gateway) des Tunnels betreffen, in die Verbindung `conn %default` ausgelagert. Dadurch gelten diese Informationen für alle Verbindungen und müssen nicht bei jeder Verbindung erneut definiert werden. Hierbei handelt es sich um die IP Adresse, das Standard Gateway und den öffentlichen RSA Schlüssel des Gateway NewYork. Außerdem wurde der Wert `keyingtries` auf den Wert Eins gesetzt, damit das VPN Gateway nicht versucht eine unterbrochene Verbindung wiederaufzubauen.

Die eigentlichen Verbindungen enthalten hauptsächlich nur noch die Informationen des Roadwarriors. Damit das VPN Gateway in der Lage ist verschiedene Roadwarrior unterscheiden zu können, wird hier eine Identifikation definiert (`rightid=roadwarrior@berlin`). Der Roadwarrior wird sich später mit dieser Identifikation (ID) beim Gateway anmelden.

Da das VPN Gateway die Verbindungen nicht starten kann, weil es die IP Adresse des Roadwarriors nicht kennt, werden diese nur mit `auto=add` geladen.

Die Datei `ipsec.secrets` muss nicht angepasst werden Sie kann direkt aus dem Listing 5.36 übernommen werden.

Nun kann FreeS/WAN auf dem Gateway gestartet (`ipsec setup start`) und der erfolgreiche Ladevorgang der Verbindungen in der Protokolldatei `/var/log/secure` beziehungsweise mit dem Befehl `ipsec auto –status` kontrolliert werden.

Roadwarrior

Die Konfiguration des Roadwarriors beim Einsatz öffentlicher Schlüssel für die Authentifizierung ist nicht schwierig. Es ist lediglich erforderlich die normale Konfiguration aus Listing 5.37 anzupassen, da die eigene IP Adresse und das eigene Standard Gateway zum Zeitpunkt der Konfiguration unbekannt sind. Die resultierende Konfigurationsdatei für den Roadwarrior ist in Listing 5.61 abgebildet. Dabei wurden zur erhöhten Lesbarkeit wieder einige Parameter umgestellt.

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

conn %default
        authby=rsasig
        keyingtries=0
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftrsasigkey=0sAQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKi
        NrLgyNfBo9npxZdOmolWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2
        DCPO5yS2OEa6HrsOjti+tyIy2LxSVT78p94FVixsAoJyYjnOyGsZcG+HjJGb
        cpPEBLNHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qdbu
        ho9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBHqX69
        oQVsncnuFderIKp7zrP4HUMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH5h3iG
        /EDUofMl9TjqJR7L4x22H0sdV6txoRC4R0LT
        right=%defaultroute

conn rsa-newyork-rwberlin
        rightid=roadwarrior@berlin
        rightrsasigkey=0sAQOLr0P0yVDh3SETo7p2J+RC6PvbnR7nI85QKVUlfkJG
        ODbJF6ZhrxdFWAjx6gF0oh9BM9BpJoSxJdkwxMTW00QvGSZ9OSPxogb
        7m52RqJwfYaQVka6xjf7f/6xXIful4dWfUFOlaemZf8L+uox5RT7RR8ZU7
        w8uDqV38bbh6Odo3hoBDFkdZ4yplW+HKqScqTVtTZgSPcT7rhSt6XqUbB
        dUMgpTNEeMNKNxUyE6Gabl+ejpULlN7KSebRti6o6bUU1aiqahk64oAPL03
        B1ivqqDSB+iMVSJM1NnltwThlZ0FNobbTXXL3BaccQrUDbQxm/DCxn1SF
        N64i633/Q384+uxI92rQ8hCxLJL/YPVTSwc9S/
        auto=start

conn rsa-newyorknet-rwberlin
        rightid=roadwarrior@berlin
        leftsubnet=10.0.1.0/24
        rightrsasigkey=0sAQOLr0P0yVDh3SETo7p2J+RC6PvbnR7nI85QKVUlfkJG
        ODbJF6ZhrxdFWAjx6gF0oh9BM9BpJoSxJdkwxMTW00QvGSZ9OSPxogb
        7m52RqJwfYaQVka6xjf7f/6xXIful4dWfUFOlaemZf8L+uox5RT7RR8ZU7
        w8uDqV38bbh6Odo3hoBDFkdZ4yplW+HKqScqTVtTZgSPcT7rhSt6XqUbB
```

```
dUMgpTNEeMNKNxUyE6Gabl+ejpULlN7KSebRti6o6bUU1aiqahk64oAPL03
B1ivqqDSB+iMVSJM1NnltwThlZ0FNobbTXXL3BaccQrUDbQxm/DCxn1SF
N64i633/Q384+uxI92rQ8hCxLJL/YPVTSwc9S/
auto=start
```

Listing 5.61 Roadwarrior Konfiguration mit RSA Schlüsseln (Roadwarrior)

Um einen einfachen Vergleich der beiden Konfigurationsdateien aus Listing 5.60 und 5.61 zu ermöglichen, wurde hier keine weitere Umstellung mehr vorgenommen. Dabei wäre es sehr wohl möglich auch die öffentlichen RSA Schlüssel des Roadwarriors und seine Identifikation mit in die `conn %default` zu übernehmen. Die Unterschiede der beiden Dateien wurden **fett** markiert.

Der wesentliche Unterschied liegt in der Verwendung des Wertes `%defaultroute` für die zu verwendene Netzwerkkarte (`interfaces=`) und die eigene IP Adresse (`right=`). Hiermit wird FreeS/WAN angewiesen, die richtige IP Adresse und die aktuell zu verwendene Netzwerkkarte automatisch beim Start zu ermitteln. Dabei kann es sich um eine Ethernet Karte (eth0), ein (DSL-)Modem (ppp0) oder eine ISDN Karte handeln (ippp0).

Außerdem wird der Roadwarrior angewiesen, die Verbindung direkt beim Start von FreeS/WAN aufzubauen (`auto=start`) und bei einem Abbruch der Verbindung deren Neustart zu veranlassen (`keyingtries=0`).

Die Datei `ipsec.secrets` kann unverändert aus dem letzten Kapitel übernommen werden. Diese Datei enthält den privaten RSA Schlüssel des Roadwarriors. Roadwarrior und VPN Gateway verfügen hier über unterschiedliche private Schlüssel. Diese Schlüssel können mit dem Befehl `ipsec newhostkey` erzeugt werden. Weitere Hinweise enthält das Kapitel »Automatisch verschlüsselte Verbindung mit öffentlichen RSA Schlüsseln«.

Wird nun die FreeS/WAN auf dem Roadwarrior gestartet, so kann der erfolgreiche Aufbau der Tunnel getestet werden.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
# ipsec eroute
0          5.0.0.1/32          -> 3.0.0.1/32          => tun0x1004@3.0.0.1
0          5.0.0.1/32          -> 10.0.1.0/24         => tun0x1002@3.0.0.1
```

VPN Gateway mit mehreren Roadwarriors

Wenn das VPN Gateway mehrere Roadwarriors unterstützen soll, so benötigt es von jedem einzelnen dieser Rechner die öffentlichen RSA Schlüssel. Damit diese Schlüssel auseinander gehalten werden können, ist es erforderlich, dass jeder Roadwarrior über eine eigene Identität (`rightid`) verfügt

und die `ipsec.conf` Datei des VPN Gateway für jeden dieser Roadwarrior eine eigene Verbindung enthält. So kann es zu sehr langen und unübersichtlichen Konfigurationsdateien auf dem VPN Gateway kommen.

Es ist daher sinnvoll, die Konfiguration der Roadwarrior auf dem VPN Gateway bei der Verwendung öffentlicher RSA Schlüssel in einzelne Dateien auszugliedern. Die fertige Konfigurationsdatei `ipsec.conf` auf dem VPN Gateway ist in Listing 5.62 dargestellt.

```
config setup
        interfaces=\}ipsec0=eth0\{
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search

conn %default
        authby=rsasig
        keyingtries=1
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftrsasigkey=0sAQNuLpOLXNQ1RyWZx6CHPWeNGGReXv/KuTQkEHZKi
        NrLgyNfBo9npxZOmolWaDNpPLzyhkALWeg/0GUKWSO7GRYLSfVuAaF2D
        CPO5yS2OEa6HrsOjti+tyIy2LxSVT78p94FVisAoJyYjnOyGsZcG+HjJGbcp
        PEBLNHrB1fOzIQBgS1+RtrYfWSlT87BLYRwR3jcV1WltWFYoXGclI3qduho
        9pnjPP4sCsxo9x+zexow/u7DiFChZuCYDtKKeOaUmBdlB529NpBHqX69o
        QVsncnuFderIKp7zrP4UMWHhMjz0DWzL3WQWyvdfMp9qGLJwoH5h3iG/E
        DUofMl9TjqJR7L4x22H0sdV6txoRC4R0LT
        right=%any

include /etc/ipsec.d/ipsec.*.conf
```

Listing 5.62 Verwaltung der Roadwarrior in einzelnen Dateien (`ipsec.conf`)

Diese Datei enthält lediglich noch die Standardeinstellungen für alle Roadwarrior Verbindungen. Wichtig ist die letzte Zeile. Hiermit werden nun weitere Dateien eingelesen. Dabei ist ein Fileglobbing mit `*` möglich. Die Konfiguration für die Roadwarriors erfolgt nun in einzelnen Dateien. So können die Dateien `/etc/ipsec.d/ipsec.rwberlin.conf` und `/etc/ipsec.d/ipsec.rwlissabon.conf` angelegt werden, in denen die entsprechenden öffentlichen Schlüssel und Identitäten der Roadwarriors gespeichert werden. Listing 5.63 zeigt ein Beispiel.

```
conn rsa-newyork-rwlissabon
        rightid=roadwarrior@lissabon
        rightrsasigkey=0sAQNG0jnSNzz2mAAZ0ftBD/cX2Kz4CnhveajWkNJe5J7V
        IVPCjTEr2aL+wFJOKT525u1Ctp3LUoMKVG1jrH85vVQjo2aI9XFVrVaa2
```

```
        pQqv2JuPNCucgnHTm9vJFFfVLliVoQD3Ve/XtviaGxY6Uy9gUneooPJlu1g
        qGY10MM1JLKK9sA9Tis7/PWXxLaN8IyBmGkGI3GEGAcJn90yNwvrggRH
        BFgGc8Fs+jV9uRKvo0OQrD1O5t11RmUGZNfPi/MugXKWNlEqFmBkCSET
        UJgnYmDq0tbGFkOzr8yxoUz2NWI1d7usc39KQz0euJPO/jSrIg5I9+wW5AO
        Ro8G7tN+M/p356uBFHstRGbZMYR+dRb/RVXOn
        auto=add

conn rsa-newyorknet-rwlissabon
        rightid=roadwarrior@lissabon
        leftsubnet=10.0.1.0/24
        rightrsasigkey=0sAQNG0jnSNzz2mAAZ0ftBD/cX2Kz4CnhveajWkNJe5J7V
        IVPCjTEr2aL+wFJOKT525u1Ctp3LUoMKVG1jrH85vVQjo2aI9XFVrVaa2
        pQqv2JuPNCucgnHTm9vJFFfVLliVoQD3Ve/XtviaGxY6Uy9gUneooPJlu1g
        qGY10MM1JLKK9sA9Tis7/PWXxLaN8IyBmGkGI3GEGAcJn90yNwvrggRH
        BFgGc8Fs+jV9uRKvo0OQrD1O5t11RmUGZNfPi/MugXKWNlEqFmBkCSET
        UJgnYmDq0tbGFkOzr8yxoUz2NWI1d7usc39KQz0euJPO/jSrIg5I9+wW5AO
        Ro8G7tN+M/p356uBFHstRGbZMYR+dRb/RVXOn
        auto=add
```

Listing 5.63 Verwaltung der Roadwarriors in einzelnen Dateien (`ipsec.rwlissabon.conf`)

Nun können die Roadwarriors relativ einfach in einzelnen Dateien administriert werden.

Fazit

Die Verwendung von RSA Schlüsseln in einem Roadwarrior Szenario ist von Vorteil. Hier kann jeder Roadwarrior eindeutig identifiziert werden und einen eigenen Schlüssel verwenden. Es ist nicht erforderlich, dass alle Roadwarriors die selben Anmeldeinformationen verwenden, wie bei PSK authentifizierten Roadwarrriorn.

Leider ist es jedoch erforderlich, die öffentlichen Schlüssel aller Roadwarriors auf dem VPN Gateway zu speichern. Dies erzeugt einen hohen Verwaltungsaufwand. Der Verwaltungsaufwand ist dabei um so größer je mehr Roadwarriors unterstützt werden sollen und je mehr Fluktuation bei diesen Roadwarriors existiert. Darüber hinweg täuscht auch nicht die Administration der Roadwarrior in einzelnen Dateien. Die Verwendung von X.509 Zertifikaten für die Authentifizierung von Roadwarriors kann hier eine große Hilfe sein.

Roadwarrior mit X.509 Zertifikaten

Beim Aufbau eines VPNs mit vielen Teilnehmern wie zum Beispiel Roadwarriors kann die Verwendung von X.509 Zertifikaten große Erleichterung bringen. Es ist so nicht nur möglich, andere IPsec Implementierungen mit FreeS/WAN kommunizieren zu lassen, sondern es besteht auch die Möglich-

keit diese Roadwarriors einfach zu verwalten. Es ist nicht erforderlich sämtliche öffentlichen Schlüssel der Roadwarriors auf dem VPN Gateway zu speichern. Nicht mehr benötigte Zertifikate können sehr einfach mit einer Widerrufliste (CRL) aufgehoben werden.

Dieses Kapitel gibt nun die entsprechenden Hinweise für den Aufbau einer derartigen Lösung. Dabei wird zunächst die Konfiguration des VPN Gateways besprochen. Anschließend wird die Konfiguration des Roadwarriors vorgestellt und erklärt. Die Konfiguration von heterogenen Clients, zum Beispiel Windows 2000, wird in dem Kapitel 9, »Fortgeschrittene Konfiguration« besprochen. Ihre Konfiguration des VPN Gateways unterscheidet sich jedoch beim Einsatz von Windows 2000 als Roadwarrior nicht.

Dieses Kapitel geht davon aus, dass Sie bereits in der Lage sind, x509 Zertifikate zu erzeugen. Dies kann mit dem Kommando `CA.pl` erfolgen (siehe Exkurs »Exkurs: Erzeugung von X.509 Zertifikaten mit OpenSSL«) oder mit einer CA (siehe Kapitel 8, »Aufbau einer Public Key Infrastruktur«).

Außerdem wird davon ausgegangen, dass Sie die entsprechenden Zertifikate bereits erzeugt und die Dateien an den vorgeschriebenen Stellen hinterlegt haben. Aus Platzgründen soll diese Wiederholung hier unterbleiben. Bitte lesen Sie die entsprechenden Abschnitte ansonsten nach und versuchen Sie zunächst eine normale VPN Verbindung mit x509 Zertifikaten aufzubauen.

VPN Gateway

Die Konfiguration des VPN Gateways ähnelt sehr stark der in Listing 5.48 abgebildeten Konfiguration. Angepasst an den Roadwarrior ist die Konfiguration in Listing 5.64 dargestellt.

```
config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search

conn %default
        authby=rsasig
        leftrsasigkey=%cert
        rightrsasigkey=%cert
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftcert=certs/newyork_cert.pem
        leftid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
```

```
OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net"
        right=%any
        keyingtries=1

conn x509-newyork-roadwarrior
        auto=add

conn x509-newyorknet-roadwarrior
        leftsubnet=10.0.1.0/24
        auto=add
```

Listing 5.64 Roadwarrior mit x509 Zertifikaten (VPN Gateway)

Hier wird, wie in den letzten Kapiteln zur Roadwarriorkonfiguration mit PSKs und RSA Schlüsseln, in der Konfigurationsdatei darauf geachtet, dass die Tunnel nicht automatisch gestartet (`auto=add`) und vom VPN Gateway bei einem Fehler nicht wieder aufgebaut werden (`keyingtries=1`). Die Tunnel müssen jedoch geladen werden (`auto=add`), da ansonsten der Tunnelaufbau des Roadwarriors abgelehnt wird.

Außerdem wird die Identität des Roadwarriors nicht mehr in dieser Datei erwähnt. Der Roadwarrior wird diese Identität bei der Anmeldung am VPN Gateway selbst übermitteln.

Die Datei `/etc/ipsec.secrets` kann ebenfalls unverändert aus dem letzten Abschnitt übernommen werden. Diese Datei enthält den Hinweis, in welcher Datei sich der private Schlüssel befindet, und mit welcher Passphrase er geschützt ist (Listing 5.65).

```
: RSA newyork_req.pem "certkenn"
```

Listing 5.65 Die Datei `ipsec.secrets` auf dem x509 VPN Gateway

Mit der Erzeugung beziehungsweise Anpassung dieser Dateien ist die Konfiguration des x509 VPN Gateways abgeschlossen. Zur Wiederholung und zur Kontrolle hier noch einmal die Liste der erforderlichen Dateien:

- **`/etc/ipsec.secrets`** siehe oben
- **`/etc/ipsec.conf`** siehe oben
- **`/etc/ipsec.d/cacerts/cacert.pem`** selbstsigniertes Zertifikat der CA
- **`/etc/ipsec.d/crls/crl.pem`** signierte Rückrufliste der CA
- **`/etc/ipsec.d/certs/newyork_cert.pem`** durch die CA ausgestelltes Zertifikat des VPN Gateways
- **`/etc/ipsec.d/private/newyork_req.pem`** privater Schlüssel des VPN Gateways

Wenn Sie diesen Dateien teilweise andere Namen gegeben haben, so achten Sie darauf, dass Sie die entsprechenden Konfigurationsdateien anpassen. Im Falle des CA Zertifikates und der Rückrufliste ist dies jedoch unkritisch. Sämtliche in diesem Verzeichnis sich befindenden Dateien werden von FreeS/WAN gelesen.

Ist soweit alles vorbereitet, kann FreeS/WAN gestartet werden und der erfolgreiche Start in der Protokolldatei `/var/log/secure` nachvollzogen werden.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
```

Roadwarrior

Die Konfiguration des Roadwarriors ähnelt sehr stark der in Listing 5.64 abgebildeten Konfiguration des VPN Gateways. Angepasst an den Roadwarrior ist die Konfiguration in Listing 5.66 dargestellt.

```
# In Version 2 ist die folgende Zeile erforderlich
# version 2

config setup
        interfaces=%defaultroute
        klipsdebug=none
        plutodebug=none
        # Die folgenden beiden Zeilen werden von 2.x nicht unterstützt.
        plutoload=%search
        plutostart=%search

conn %default
        authby=rsasig
        leftrsasigkey=%cert
        rightrsasigkey=%cert
        left=3.0.0.1
        leftnexthop=3.255.255.254
        leftid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=NewYork/Email=ralf@spenneberg.net"
        right=%defaultroute
   rightid="/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
OU=Wireless-VPN/CN=Berlin/Email=ralf@spenneberg.net"
        rightcert=certs/berlin_cert.pem
        keyingtries=0

conn x509-newyork-rwberlin
        auto=start
```

```
conn x509-newyorknet-rwberlin
        leftsubnet=10.0.1.0/24
        auto=start
```

Listing 5.66 Roadwarrior Berlin mit x509 Zertifikaten (Roadwarrior)

Hier werden, wie in den letzten Kapiteln zur Roadwarriorkonfiguration mit PSKs und RSA Schlüsseln, in der Konfigurationsdatei die Tunnel automatisch gestartet (`auto=start`) und vom Roadwarrior wieder aufgebaut, wenn diese durch einen Fehler (`keyingtries=0`) zusammenbrechen. Da der Roadwarrior seine eigene IP Adresse zum Zeitpunkt der Konfiguration nicht kennt wird hier `%defaultroute` verwendet.

Die Datei `/etc/ipsec.secrets` kann ebenfalls unverändert aus dem letzten Kapitel übernommen werden. Diese Datei enthält den Hinweis, in welcher Datei sich der private Schlüssel befindet, und mit welcher Passphrase er geschützt ist (Listing 5.67).

```
: RSA berlin_req.pem "certkenn"
```

Listing 5.67 Die Datei `ipsec.secrets` auf dem Roadwarrior

Mit der Erzeugung beziehungsweise Anpassung dieser Dateien ist die Konfiguration des X.509 Roadwarriors abgeschlossen. Dennoch sollte auch hier noch einmal überprüft werden, ob sämtliche Dateien sich an den vorgesehenen Orten befinden (siehe letzter Abschnitt).

Wenn Sie diesen Dateien teilweise andere Namen gegeben haben, so achten Sie darauf, dass Sie die entsprechenden Konfigurationsdateien anpassen. Im Falle des CA Zertifikates und der Rückrufliste ist dies jedoch unkritisch. Sämtliche in diesem Verzeichnis sich befindenden Dateien werden von FreeS/WAN gelesen.

Ist soweit alles vorbereitet, kann FreeS/WAN gestartet und der Tunnelaufbau beobachtet werden.

```
# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...

Feb 17 14:08:05 newyork pluto[1369]:   loaded host cert file
'/etc/ipsec.d/certs/newyork_cert.pem' (3805 bytes)
Feb 17 14:08:05 newyork pluto[1369]: added connection description
"x509-newyorknet-roadwarrior"
Feb 17 14:08:08 newyork pluto[1369]:   loaded host cert file
'/etc/ipsec.d/certs/newyork_cert.pem' (3805 bytes)
Feb 17 14:08:08 newyork pluto[1369]: added connection description
"x509-newyork-roadwarrior"
Feb 17 14:08:10 newyork pluto[1369]: listening for IKE messages
```

```
Feb 17 14:08:10 newyork pluto[1369]: adding interface ipsec0/eth0 3.0.0.1
Feb 17 14:08:11 newyork pluto[1369]: loading secrets from
"/etc/ipsec.secrets"
Feb 17 14:08:11 newyork pluto[1369]: loaded private key file
'/etc/ipsec.d/private/newyork_req.pem' (1675 bytes)
Feb 17 14:25:29 newyork pluto[1369]: "x509-newyorknet-roadwarrior"[1]
5.0.0.1 #1: responding to Main Mode from unknown peer 5.0.0.1
Feb 17 14:25:29 newyork pluto[1369]: "x509-newyorknet-roadwarrior"[1]
5.0.0.1 #1: Peer ID is ID_DER_ASN1_DN: 'C=DE, ST=NRW, L=Steinfurt,
O=Spenneberg.com, OU=Wireless-VPN, CN=Berlin, E=ralf@spenneberg.net'
Feb 17 14:25:29 newyork pluto[1369]: "x509-newyork-roadwarrior"[1] 5.0.0.1
#1: deleting connection "x509-newyorknet-roadwarrior" instance with peer
5.0.0.1
Feb 17 14:25:29 newyork pluto[1369]: "x509-newyork-roadwarrior"[1] 5.0.0.1
#1: sent MR3, ISAKMP SA established
Feb 17 14:25:29 newyork pluto[1369]: "x509-newyorknet-roadwarrior"[2]
5.0.0.1 #2: responding to Quick Mode
Feb 17 14:25:31 newyork pluto[1369]: "x509-newyorknet-roadwarrior"[2]
5.0.0.1 #2: IPsec SA established
Feb 17 14:25:31 newyork pluto[1369]: "x509-newyork-roadwarrior"[1] 5.0.0.1
#3: responding to Quick Mode
Feb 17 14:25:32 newyork pluto[1369]: "x509-newyork-roadwarrior"[1] 5.0.0.1
#3: IPsec SA established
```

Listing 5.68 Protokoll des VPN Gateways während des Tunnelaufbaus

Mit Hilfe der Protokolldatei kann nachvollzogen werden, dass der Aufbau des Tunnels erfolgreich und unproblematisch erfolgt. Die Anzahl der Roadwarriors ist hier kein Problem.

Fazit

Die Verwaltung der Roadwarrior mit X.509 Zertifikaten ist die einfachste und mächtigste Variante. Hierbei ist es nicht erforderlich, dass die öffentlichen Schlüssel oder PSKs zwischen den Roadwarriors und dem VPN Gateway ausgetauscht werden. Der einzige öffentliche Schlüssel, der auf allen Stationen bekannt sein muss, ist der Schlüssel der CA. Anschließend vertrauen die Kommunikationspartner im VPN automatisch allen weiteren Rechnern, die über ein signiertes Zertifikat dieser CA verfügen. Zusätzliche Sicherheit bietet eine Zertifikatsrückrufliste (CRL), die angelegt werden kann und Zertifikate enthält, die bereits vor Ablauf ihrer Gültigkeit nicht mehr verwendet werden dürfen. Hierzu ist es sinnvoll, diese Liste auf einem Webserver oder einem LDAP Server zur Verfügung zu stellen. Die Kommunikationspartner können dann diese Liste in regelmäßigen Abständen (zum Beispiel täglich) herunterladen. Ein Aktualisierung von Pluto erfolgt mit dem Befehl `ipsec auto --reread-crls`.

5.6 FreeS/WAN 2.x

Die aktuelle Version von FreeS/WAN ist die Version 2. Viele Systeme setzen jedoch noch die Version 1.9x ein. Daher wurde bisher nur diese Version besprochen. Die Konfigurationsdateien können auch fast unverändert von Version 1.9x auf Version 2 übernommen werden. Dieser Abschnitt bespricht die durchzuführenden Änderungen, damit die besprochenen Konfigurationsdateien auch auf Version 2 anzuwenden sind. Die besondere Neuerung der Version 2, die automatische Unterstützung der opportunistischen Verschlüsselung, wird im Kapitel *Fortgeschrittene Konfiguration* besprochen.

Um sofort nach der Installation die opportunistische Verschlüsselung zu unterstützen, führt FreeS/WAN 2 die `Policy Groups` ein. Sie ermöglichen die Konfiguration der Tunnel in Gruppen. Für den Einsatz von FreeS/WAN 2 als Ersatz für FreeS/WAN 1.9x bei Beibehaltung der Konfiguration müssen diese Gruppen deaktiviert werden.

Im Folgenden wird sowohl die Deaktivierung als auch der Einsatz der Policy Groups besprochen.

5.6.1 Upgrade von FreeS/WAN 1.9x

Wichtig bei einem Upgrade ist die Anpassung der Konfigurationsdatei. Hierfür muss zu Beginn der Parameter `version` stehen.

```
version 2
```

Außerdem haben sich einige Default-Werte geändert. Wenn die Konfigurationsdatei sie nicht sowieso schon geändert hat, sind folgende Zeilen hinzuzufügen:

```
config setup
        interfaces=%none     # new default is %defaultroute
        plutoload=%none      # new default is %search
        plutostart=%none     # new default is %search

conn %default
        uniqueids=no                # new default is yes
        keyingtries=3               # new default is %forever
        disablearrivalcheck=yes     # new default is no
        authby=secret               # new default is rsasig
        leftrsasigkey=%none         # new default %dnsondemand
        rightrsasigkey=%none        # new default %dnsondemand
```

Zusätzlich muss die eingebaute opportunistische Verschlüsselung, die für ihre Konfiguration die Policy Groups verwendet, deaktiviert werden. Hierfür ist es erforderlich, dass die folgenden Zeilen zur FreeS/WAN Konfigurationsdatei hinzugefügt werden:

```
conn block
  auto=ignore

conn private
  auto=ignore

conn private-or-clear
  auto=ignore

conn clear-or-private
  auto=ignore

conn clear
  auto=ignore

conn packetdefault
  auto=ignore
```

Diese Zeilen führen dazu, dass die eingebauten Verbindungsdefinitionen (`block`, `private`, `private-or-clear`, `clear-or-private`, `clear` und `packetdefault`) ignoriert werden, und nur die vom Administrator definierten Verbindungen über das Verhalten von FreeS/WAN entscheiden.

5.6.2 Anwendung der Policy Groups

Die Policy Groups bieten eine neue elegante Variante zur Konfiguration von FreeS/WAN ab Version 2.0. Hiermit ist es möglich verschiedenste Verbindungsdefinitionen, die bei den Vorgängerversionen einzeln angegeben werden mussten, in Gruppen zusammenzufassen. Dies erhöht die Lesbarkeit und Übersichtlichkeit.

Hierzu definiert der Administrator eine Policy Group, die dann für alle Rechner und Netzwerke angewendet wird, die sich in einer entsprechenden Liste befinden. FreeS/WAN erzeugt dann intern die einzelnen benötigten Verbindungen. Der Administrator muss sie nicht mehr einzeln in der Konfigurationsdatei `/etc/ipsec.conf` erzeugen.

Die Anwendung der Policy Groups ist bisher nur sinnvoll, wenn opportunistische Verschlüsselung eingesetzt werden soll. Hierbei handelt es sich um eine Funktion, die bisher unter allen dem Autor bekannten IPsec Implementierungen nur von FreeS/WAN unterstützt wird. Die opportunistische Ver-

schlüsselung (Opportunistic Encryption, OE) versucht, automatisch eine verschlüsselte Verbindung mit jedem beliebigen Rechner aufzubauen. Hierzu benötigt die OE die öffentlichen Schlüssel der Kommunikationspartner oder eines VPN-Gateways, welches den Kommunikationspartner schützen kann. Die OE ermittelt diese Information mit Hilfe des DNS Systems. Hierzu sind besondere Einträge (KEY und TXT Resource Records) im DNS System erforderlich.

Die genaue Funktion und die damit verbundene Anwendung der Policy Groups wird daher in einem späteren Kapitel besprochen. Hier soll nur kurz die Funktion der Gruppen erläutert werden. FreeS/WAN bringt bereits fünf vordefinierte Policy Groups mit: `private`, `private-or-clear`, `clear-or-private`, `clear` und `block`.

1. **`private`** FreeS/WAN versucht eine verschlüsselte Verbindung mit den aufgeführten Rechnern und Netzwerken aufzubauen. Wenn dies fehlschlägt, werden ausgehende Pakete verworfen.
2. **`private-or-clear`** FreeS/WAN versucht eine verschlüsselte Verbindung mit den aufgeführten Rechnern und Netzwerken aufzubauen. Wenn dies fehlschlägt, werden die Daten ohne Verschlüsselung übertragen.
3. **`clear-or-private`** FreeS/WAN übermittelt alle Daten ohne Verschlüsselung. Wird jedoch von der Gegenseite eine verschlüsselte Verbindung aufgebaut, so nutzt auch dieser Rechner die Verbindung und überträgt die Daten verschlüsselt.
4. **`clear`** FreeS/WAN erlaubt nur die Übertragung der Daten in Klartext. Eine Verschlüsselung wird abgelehnt.
5. **`block`** FreeS/WAN blockiert jede ausgehende Verbindung zu den angegebenen Rechnern und Netzwerken.

Der Administrator kann sich aber auch selbst weitere Gruppen definieren. Um nun einen bestimmten Rechner oder bestimmte Netzwerke in eine dieser Gruppen aufzunehmen legt der Administrator lediglich eine entsprechende Datei im Verzeichnis `/etc/ipsec.d/policies` an. Die Datei `private` oder `private-or-clear` führt zum Bespiel alle Rechner und Netzwerke auf, die zu diesen Gruppe gehören.

```
# cd /etc/ipsec.d/policies
# cat private
   3.0.0.0/24
   3.0.7.0/27
   www.spenneberg.com
# cat private-or-clear
   0.0.0.0/0      # Alle, versuche grundsätzlich zunächst eine
                  # verschlüsselte Verbindung aufzubauen!
```

5.7 Konfiguration der Firewall

Befindet sich auf oder vor dem VPN Gateway eine Firewall so erfordert der erfolgreiche Aufbau eines VPN Tunnels eine Anpassung der Konfiguration. Sind die Firewall und FreeS/WAN auf demselben Gerät, so bietet FreeS/WAN die Möglichkeit beim Aufbau eines Tunnels hierfür die Firewallregeln anzupassen.

Zunächst sollen die grundsätzlichen Dinge besprochen werden. Im Anschluss wird dann erklärt, wie FreeS/WAN die Regeln bei dem Aufbau eines Tunnels anpassen kann.

Für den Aufbau eines VPN Tunnels ist es erforderlich, dass die IPsec Protokolle ESP und bei Bedarf auch AH die Firewall passieren können. Zusätzlich ist es bei automatisch verschlüsselten Verbindungen (PSK, RSA, X.509 Authentifizierung) erforderlich, das IKE Protokoll durch die Firewall zu lassen.

Handelt es sich bei der Firewall um einen Linux Rechner mit einen Linux Kernel 2.4.x so können die Regeln mit dem `iptables` Kommando erzeugt werden. Hier kann leider nicht die Syntax und der Hintergrund des Kommandos erläutert werden. Dies füllt ein eigenes Buch. Es soll daher auf das Buch von Robert Ziegler »Linux Firewalls« in der zweiten Auflage verwiesen werden, das ebenfalls im Markt+Technik Verlag erschienen ist (ISBN 3-8272-6257-7).

Im Folgenden werden die Regeln vorgestellt, für den Fall, dass das VPN Gateway sich auf der Firewall befindet oder dahinter befindet.

Ist der `iptables` Paketfilter auf der Firewall, so ist es erforderlich, dass der Paketfilter den Versand und Empfang von IKE, ESP und AH Paketen zulässt. Das Protokoll AH wird jedoch nur benötigt, wenn es auch von FreeS/WAN genutzt wird (`auth=ah`). Zusätzlich muss der Paketfilter Netzwerkpakete, die den Tunnel nutzen möchten, passieren lassen. Listing 5.69 demonstriert die benötigten Regeln.

```
# Auszug eines Firewallskriptes
#
# Externe Netzwerkkarte
EXTCARD=eth0
# IPsec Netzwerkkarte
IPSEC=ipsec0
#
# Eigenes Netzwerk
LOCALNET=10.0.1.0/24
# Partner Netzwerk
REMOTENET=10.0.2.0/24
```

```
#
# Erlaube IKE (500/udp)
iptables -A INPUT  -p udp --dport 500 -i $EXTCARD -j ACCEPT
iptables -A OUTPUT -p udp --dport 500 -o $EXTCARD -j ACCEPT
#
# Erlaube ESP und AH
for protocol in "50,51"
do
  iptables -A INPUT  -p $protocol -i $EXTCARD -j ACCEPT
  iptables -A OUTPUT -p $protocol -o $EXTCARD -j ACCEPT
done
#
# Erlaube die Tunnelverwendung
iptables -A FORWARD -s $LOCALNET -d $REMOTENET -o $IPSEC -j ACCEPT
iptables -A FORWARD -d $LOCALNET -s $REMOTENET -i $IPSEC -j ACCEPT
```

Listing 5.69 Paketfilter Regeln bei Einsatz von FreeS/WAN auf der Firewall

Diese Regeln lassen ungehindert sämtlichen Verkehr im VPN zu. Dies ist nicht die ideale Lösung, aber häufig so gewünscht. Es sollte dennoch versucht werden diesen Verkehr auf ein absolutes Mindestmaß zu reduzieren und entsprechende Regeln zu erzeugen. Es entsteht ansonsten die Möglichkeit an der Firewall vorbei über das VPN auf das geschützte Netz zuzugreifen. Dies ist sicherlich nicht immer erwünscht. Auch wenn der Zugriff aus einem anderen geschützten Netzwerk erfolgt, birgt dieser Zugriff immer noch Gefahren. Die Sicherheit des gesamten Netzwerkes (bestehend aus den beiden über das VPN verbundenen Netzen) ist nur so hoch wie ihr schwächstes Glied.

Befindet sich die Firewall zwischen dem VPN Gateway und dem Internet, so ist folgender Regelsatz (5.70) erforderlich. Er erlaubt die Passage von IKE, ESP und AH Paketen durch die Firewall zu dem VPN Gateway.

```
# Auszug eines Firewallskriptes
#
# Externe Netzwerkkarte
EXTCARD=eth0
# Interne Netzwerkkarte
INTCARD=eth1
#
# lokales VPN Gateway
LOCALVPN=3.0.0.1
# Entferntes VPN Gateway
REMOTEVPN=7.0.0.1
#
# Erlaube IKE (500/udp)
iptables -A FORWARD  -p udp --dport 500 -i $EXTCARD -o $INTCARD \
   -s $REMOTEVPN -d $LOCALVPN -j ACCEPT
```

```
iptables -A FORWARD  -p udp --dport 500 -o $EXTCARD -i $INTCARD \
   -d $REMOTEVPN -s $LOCALVPN -j ACCEPT
#
# Erlaube ESP und AH
for protocol in "50,51"
do
  iptables -A FORWARD  -p $protocol -i $EXTCARD -o $INTCARD \
     -s $REMOTEVPN -d $LOCALVPN -j ACCEPT
  iptables -A FORWARD  -p $protocol -o $EXTCARD -i $INTCARD \
     -d $REMOTEVPN -s $LOCALVPN -j ACCEPT
done
```

Listing 5.70 Paketfilter Regeln bei dem Einsatz von FreeSIWAN hinter der Firewall

5.7.1 Anpassung des _updown Skriptes

Mit den bisher gezeigten Paketfilter-Regeln ist es möglich, ein VPN aufzubauen und zu nutzen. Jedoch ist es häufig sinnvoll, zusätzliche Regeln beim Aufbau eines Tunnels zu erzeugen, um so gezielt den Durchgang durch eine Firewall zu erlauben.

Hierfür kann das `_updown` Skript genutzt werden. Es wird von Pluto für fünf verschiedene Vorgänge aufgerufen.

- **`prepare-host|prepare-client`** Diese Routine kann für vorbereitende Maßnahmen vor dem Aufbau des Tunnels genutzt werden. Hier werden zum Beispiel alte Routen gelöscht.
- **`route-host|route-client`** Diese Routine wird genutzt, um eine Route einzutragen (`ipsec auto –route`).
- **`unroute-host|unroute-client`** Hiermit wird eine Route wieder entfernt, wenn ein Tunnel gelöscht wird (`ipsec auto –delete`)
- **`up-host|up-client`** Diese Aktion wird ausgeführt, wenn der Tunnel aufgebaut wird. Hierzu ist mindestens die Kommunikation mit IKE Paketen erforderlich. Alle weiteren nötigen Regeln können hier aktiviert werden.
- **`down-host|down-client`** Diese Aktion wird bei der Beendigung des Tunnels ausgeführt. Hier können die Regeln wieder gelöscht werden.

Hierbei wird bei einem `*-host` Aufruf davon ausgegangen, dass der Tunnel nur von dem Gateway selbst genutzt wird. Der `*-client` Aufruf geht davon aus, dass der Tunnel von einem Clientrechner oder -netzwerk hinter dem Gateway genutzt wird. Diesem Skript werden zusätzlich die folgenden Variablen übergeben, die in Firewallregeln genutzt werden können:

- **`PLUTO_VERSON`** Momentan Version 1.1 oder 1.0

- **PLUTO_VERB** Einer der oben aufgeführten Vorgänge
- **PLUTO_CONNECTION** Der Name der zu behandelnden Verbindung
- **PLUTO_NEXT_HOP** Erster Router, zu dem die Tunnelpakete gesendet werden.
- **PLUTO_INTERFACE** Der Name der virtuellen `ipsecX` Netzwerkkarte
- **PLUTO_ME** Die eigene IP Adresse
- **PLUTO_MY_CLIENT** Die IP Adresse des Clients hinter diesem VPN Gateway
- **PLUTO_MY_PORT** Der verwendete IKE Port (meist 500, als Source- und Destination-Port)
- **PLUTO_MY_PROTOCOL** Das eingesetzte Protokoll: 50 (ESP), 51 (AH), 6 (UDP) oder 17 (TCP). Die IKE Kommunikation kann auch über TCP erfolgen, daher wird hier auch die Protokollnummer 17 berücksichtigt.
- **PLUTO_MY_CLIENT_NET** Das Netzwerk hinter diesem VPN Gateway
- **PLUTO_MY_CLIENT_MASK** Die Netzmaske des Netzwerkes `PLUTO_MY_CLIENT_NET`. Wenn es sich nur um einen Rechner handelt 255.255.255.255.
- **PLUTO_PEER, PLUTO_PEER_CLIENT, PLUTO_PEER_CLIENT_NET, PLUTO_PEER_CLIENT_MASK, PLUTO_PEER_PORT, PLUTO_PEER_PROTOCOL** Analog zu `PLUTO_MY_*`.
- **PLUTO_PEER_ID** Die Identifikation des Peers. Sie kann sehr gut für Protokollzwecke genutzt werden.

Das `_updown` Skript von FreeS/WAN 1.99 unterstützt bisher nur die Modifikation von Firewallregeln mit dem Kommando `ipfwadm`. Dieses Kommando wurde beim Kernel 2.0 für die Administration der Firewall verwendet. Der Linux Kernel 2.4 verfügt über eine gewisse Kompatibilität bei Verwendung des `ipfwadm.o` Moduls. Der wesentlich mächtigere Befehl `iptables` wird erst ab FreeS/WAN 2.00 oder nach dem X.509 Patch verwendet.

ACHTUNG

Bei Verwendung des X.509 Patches wird jedoch nicht automatisch die Unterstützung des `iptables` Kommandos garantiert. Dieser Patch liefert lediglich ein eigenes `_updown.x509` Skript mit, das aber nicht genutzt wird. Damit dieses Skript nun verwendet wird, ist es erforderlich, den folgenden Parameter bei der Definition der Verbindung zu verwenden:

```
leftupdown=/usr/lib/ipsec/_updown.x509
```

Listing 5.71 Wahl des `_updown` Skriptes

Um eine Vorstellung von der Erzeugung der Regeln zu erhalten, ist in Listing 5.72 ein Auszug aus der Datei `_updown_x509` enthalten.

```
up-client:)
        # connection to my client subnet coming up
        # If you are doing a custom version, firewall commands
          go here.
        if [ "$PLUTO_MY_PROTOCOL" == "6" ] || [ "$PLUTO_MY_PROTOCOL" ==
         "17" ]
        then
          iptables -I FORWARD 1 -o $PLUTO_INTERFACE -p
           $PLUTO_PEER_PROTOCOL \
            -s $PLUTO_MY_CLIENT_NET/$PLUTO_MY_CLIENT_MASK --sport
               $PLUTO_MY_PORT \
            -d $PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK --dport
               $PLUTO_PEER_PORT -j ACCEPT
          iptables -I FORWARD 1 -i $PLUTO_INTERFACE
            -p $PLUTO_MY_PROTOCOL \
            -s $PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK --sport
               $PLUTO_PEER_PORT \
            -d $PLUTO_MY_CLIENT_NET/$PLUTO_MY_CLIENT_MASK --dport
               $PLUTO_MY_PORT -j ACCEPT
        else
          iptables -I FORWARD 1 -o $PLUTO_INTERFACE -p
               $PLUTO_PEER_PROTOCOL \
            -s $PLUTO_MY_CLIENT_NET/$PLUTO_MY_CLIENT_MASK \
            -d $PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK -j ACCEPT
          iptables -I FORWARD 1 -i $PLUTO_INTERFACE
            -p $PLUTO_MY_PROTOCOL \
            -s $PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK \
            -d $PLUTO_MY_CLIENT_NET/$PLUTO_MY_CLIENT_MASK -j ACCEPT
        fi
        if [ "$PLUTO_PEER_CLIENT" == "$PLUTO_PEER/32" ]
        then
          logger -t $TAG -p $FAC_PRIO \
            "+ $PLUTO_PEER_ID  $PLUTO_PEER -- $PLUTO_ME ==
               $PLUTO_MY_CLIENT"
        else
          logger -t $TAG -p $FAC_PRIO \
            "+ $PLUTO_PEER_ID  $PLUTO_PEER_CLIENT == $PLUTO_PEER --
               $PLUTO_ME == $PLUTO_MY_CLIENT"
        fi
        ;;
```

Listing 5.72 Beispiel updown Firewallregeln (aus `_updown.x509`)

Die Befehle aus Listing 5.72 erzeugen PaketfilterRegeln, die lediglich den Austausch von Paketen zwischen den Kommunikationspartnern des Tunnels erlauben. Zusätzlich wird mit dem Befehl `logger` die Anpassung der Fire-

wallregeln protokolliert. Die Variablen `$TAG` und `$FAC_PRIO` können am Anfang dieses Skriptes angepasst werden. Mit ihrer Standardeinstellung führen sie zur Protokollierung in der Datei `/var/log/messages`:

```
Feb 18 10:32:24 newyork vpn: + C=DE, ST=NRW, L=Steinfurt,
O=Spenneberg.com, OU=Wireless-VPN, CN=Berlin, E=ralf@spenneberg.net
5.0.0.1 -- 3.0.0.1 == 10.0.1.0/24
Feb 18 10:32:26 newyork vpn: + C=DE, ST=NRW, L=Steinfurt,
O=Spenneberg.com, OU=Wireless-VPN, CN=Berlin, E=ralf@spenneberg.net
5.0.0.1 -- 3.0.0.1
```

Listing 5.73 Protokollierung der Firewallmodifikation

ACHTUNG

Die Parameter `leftfirewall|rightfirewall` führen lediglich zur Aktivierung von `ipfwadm` Regeln. Dann wird das `_updown` Skript mit dem Argument `*:ipfwadm` aufgerufen. Sicherlich ist es möglich an der entsprechenden Stelle `iptables` Befehle aufzurufen. Dies kann jedoch später zu Verwirrung führen.

Bei der Anpassung dieser Datei sollte auf jeden Fall darauf geachtet werden, dass nicht die Original-Datei modifiziert wird. Sie wird möglicherweise bei einem Update des FreeS/WAN Paketes überschrieben und die Modifikation geht verloren. Daher sollte immer das Skript umbenannt werden und mit der Direktive `leftupdown` und `rightupdown` geladen werden.

5.7.2 Fazit

Es besteht die Möglichkeit manuell die Firewallregeln festzulegen. Dazu ist es sinnvoll grundsätzlich den Empfang und Versand von IKE, ESP und AH Paketen zu erlauben. Die Kommunikation über das Interface `ipsecX` kann auch grundsätzlich erlaubt werden.

In einigen Fällen ist es aber wichtig, diese Regeln erst bei Bedarf zu erzeugen. FreeS/WAN bietet hier mächtige Funktionen, die jedoch ein Skripting erfordern. Möglicherweise sollen in Abhängigkeit der Identität des Partners unterschiedliche Regeln erzeugt werden. Bei einem Roadwarrior ist die hierfür erforderliche IP Adresse vor Aufbau des Tunnels nicht bekannt. In solchen Fällen kann ein angepasstes `_updown` Skript die Lösung bieten. Die Unterscheidung des Roadwarriors erfolgt über die Variable `$PLUTO_PEER_ID`. Die entsprechende IP Adresse befindet sich dann in der Variablen `$PLUTO_PEER_IP`. Damit lassen sich entsprechende Regeln im Shellskript erzeugen.

6 IPsec mit Linux 2.6

Die native IPsec Implementierung im Linux Kernel ab Version 2.5.47 basiert auf dem USAGI Projekt. Hierbei handelt es sich um eine alternative Implementierung des IPsec Stacks zum FreeS/WAN Projekt. Daraus resultieren zunächst unterschiedliche Fähigkeiten und Anwendungsmöglichkeiten. Die Werkzeuge zur Administration unterscheiden sich von FreeS/WAN. Jedoch glaubt der Autor, dass diese Implementierung sehr schnell akzeptiert und angenommen wird, da sie, trotz einiger momentaner Unzulänglichkeiten, fest im Kernel integriert ist und einige Lizenzbeschränkungen wegfallen.

6.1 Einleitung

Mit dem FreeS/WAN Projekt existiert bereits seit 1996 ein Projekt, das die Implementierung eines IPsec Netzwerkstacks vorantreibt. Dennoch wurde von Dave Miller und Alexey Kuznetsov im Herbst 2002 eine native IPsec Implementierung basierend auf dem USAGI Projekt (*http://www.linux-ipv6.org/*) begonnen. Das USAGI Projekt versucht einen kompletten IPv6 Netzwerk Stack für Linux zu erzeugen. Die momentan im Kernel existierende IPv6 Implementierung ist unvollständig, sehr alt und fehlerhaft. Hierbei arbeitet das USAGI Projekt eng mit dem WIDE< Projekt (*http://www.wide.ad.jp/*), dem *TAHI* Projekt (*http://www.tahi.org/*) und dem KAME Projekt (*http://www.kame.net/*) zusammen. Alle drei Projekte beschäftigen sich mit der Implementierung und dem Test von IPv6.

Das KAME Projekt entwickelt einen IPv6 und IPsec (auch für IPv4) Stack für die *BSD Betriebssysteme. Diese Entwicklung ist sehr weit fortgeschritten. Das USAGI Projekt hat große Teile der Architektur und der Userspace Befehle übernommen. Daher ist die Konfiguration von IPsec mit dem Linux Kernel 2.6 sehr ähnlich der IPsec Konfiguration unter NetBSD und FreeBSD.[1]

Ausschlaggebend für die Entwicklung des nativen IPsec Stacks waren mehrere Gründe. Zum einen wurde es aus politischen Gründen US-Amerikanern nicht ermöglicht an dem FreeS/WAN Code mitzuwirken, aus Angst, dadurch könnten Exportprobleme für den gesamten Code entstehen. Zusätzlich gab es viele Bedenken in Bezug auf die Qualität des Kernelcodes KLIPS

1. OpenBSD verfügt seit Jahren über eine eigene IPsec-Implementierung und mit dem `isakmpd` über einen eigenen IKE Daemon. Es wird hier nicht `racoon` verwendet! Der `isakmpd` IKE Daemon kann auch in Kombination mit Linux verwendet werden. Diese Anwendung wird in einem eigenen Kapitel besprochen.

(*http://www.edlug.ed.ac.uk/archive/Sep2002/msg00244.html*). Selbst Henry Spencer, ein ehemaliger führender Entwickler des FreeS/WAN Projektes schrieb am 8. November 2002: »And, in fairness, KLIPS is the ugliest and least maintainable part of FreeS/WAN, and deserves to be supplanted.« (*http://lists.freeswan.org/pipermail/design/2002-November/003901.html*).

Führende Entwickler des FreeS/WAN Projektes haben bereits angekündigt, dass sie bei Erscheinen des Linux Kernels 2.6 auch dessen IPsec Stack unterstützen wollen. Hierzu werden dann die Userspace Werkzeuge `ipsec` und `pluto` angepasst, dass sie die entsprechenden Schnittstellen in dem Linux Kernel 2.6 nutzen können.

6.2 Lizenz

Der Kernel IPsec Stack unterliegt wie auch der Rest des Linux Kernels der GNU General Public License (GPL, siehe Anhang). Die Userspace Werkzeuge `setkey` und `racoon` unterliegen ebenfalls Open Source Lizenzen. Hierbei handelt es sich um die Lizenz des WIDE Projektes und des OpenSSL Projektes.

6.3 Installation

Die Installation ist recht einfach. Hierzu werden ein Linux Kernel mit einer Version > 2.5.47 oder 2.6.x und ein aktuelles `iputils` Paket mit `racoon` benötigt. Der Autor hofft, dass der Linux Kernel 2.6 bei Erscheinen dieses Buches bereits das Licht der Linuxwelt erblickt. Dann werden wahrscheinlich auch einige Distributionen sehr schnell diesen Kernel einbauen. Bis dahin ist es jedoch erforderlich, den Kernel selbst zu übersetzen. Für Test- und Übungszwecke ist es möglicherweise ausreichend, ihn als User Mode Linux Kernel zu übersetzen. Das Kapitel 11, »Testumgebungen«, hat weitere Informationen.

6.3.1 Kernel

Die Übersetzung des Kernels ist relativ unproblematisch. Zunächst ist es erforderlich, über einen aktuellen Kernel Sourcecode zu verfügen. Er kann von *http://www.kernel.org* geladen werden. Ist der Linux Kernel 2.6 noch nicht verfügbar, so ist ein aktueller Kernel der Version 2.5.x zu wählen. Hierbei soll aber noch einmal auf den experimentellen Status des Linux Kernels 2.5.x hingewiesen werden. Ungewöhnliche Abstürze und Datenverluste können auf-

treten. Der Autor verwendete für die Beispiele in diesem Buch verschiedene Versionen des Linux Kernels 2.5 (2.5.53, 2.5.58 und so weiter).

Wurde der Kernel Sourcecode geladen, kann er zunächst ausgepackt werden. Anschließend muss er konfiguriert und übersetzt werden. Bei der Konfiguration ist es erforderlich, dass für die spätere Verwendung von IPsec mindestens die in Listing 6.1 genannten Optionen aktiviert wurden. Außerdem sollte der Kernel an die persönliche Konfiguration angepasst werden. Hier sind inzwischen eine ganze Reihe zusätzliche Optionen hinzugefügt worden.

```
Networking support (NET) [Y/n/?] y
  *
  * Networking options
  *
  PF_KEY sockets (NET_KEY) [Y/n/m/?] y
  IP: AH transformation (INET_AH) [Y/n/m/?] y
  IP: ESP transformation (INET_ESP) [Y/n/m/?] y
  IP: IPsec user configuration interface (XFRM_USER) [Y/n/m/?] y

Cryptographic API (CRYPTO) [Y/n/?] y
  HMAC support (CRYPTO_HMAC) [Y/n/?] y
  Null algorithms (CRYPTO_NULL) [Y/n/m/?] y
  MD5 digest algorithm (CRYPTO_MD5) [Y/n/m/?] y
  SHA1 digest algorithm (CRYPTO_SHA1) [Y/n/m/?] y
  DES and Triple DES EDE cipher algorithms (CRYPTO_DES) [Y/n/m/?] y
  AES cipher algorithms (CRYPTO_AES) [Y/n/m/?] y
```

Listing 6.1 Wichtige IPsec Optionen des Kernel 2.6

Für die Übersetzung und Konfiguration des Kernels sind in dessen Wurzelverzeichnis die in Listing 6.2 genannten Befehle auszuführen. Der beim Linux Kernel 2.4 erforderliche Befehl `make dep` fällt hierbei weg.

```
# make clean
# make menuconfig   # oder make xconfig, make oldconfig oder make config
# make bzImage
# make modules
```

Listing 6.2 Konfiguration und Übersetzung des Linux Kernels

Nun ist für eine erfolgreiche Installation des Kernels und seiner Module zu sorgen. Dazu sind die Befehle `make install` und `make modules_install` auszuführen. Nun ist es noch erforderlich, den Bootmanager anzupassen, so dass der neue Kernel beim nächsten Reboot im Menü angeboten wird. Hier soll zu diesem Zweck auf das Kapitel 5.3.2, »Kompilierung und Installation des Sourcecodes« verwiesen werden.

Die meisten Distributionen verfügen wahrscheinlich noch nicht über ein `modutils`-Paket, welches in der Lage ist die Module des neuen Kernels zu laden. Wenn die Distribution kein derartiges Paket zur Verfügung stellt, besteht die Möglichkeit, das neue Paket `module-init-tools` von *http://www.kernel.org/pub/linux/kernel/people/rusty/modules/* zu laden. Dort befindet sich auch ein `modutils-<version>.src.rpm` Paket, das auf RPM basierten Rechnern für ein Upgrade des vorhandenen Paketes genutzt werden kann. Dazu ist dieses Paket zunächst mit `rpmbuild –rebuild` als Binärpaket zu erzeugen. Dieses Paket enthält sowohl die Modulverwaltungsbefehle für den Linux Kernel 2.4 als auch für den Linux Kernel 2.5/2.6.

6.3.2 Userspace Befehle

Für die Verwaltung von IPsec können drei verschiedene Systeme eingesetzt werden. Entweder kann der Benutzer FreeS/WAN benutzen. Herbert Xu stellt auf seiner Homepage (*http://gondor.apana.org.au/~herbert/*) einen Patch hierfür zur Verfügung. Leider kollidiert dieser Patch im Moment noch mit allen weiteren FreeS/WAN Patches. Thomas Walpuski hat den OpenBSD `isakmpd` auf den Linux Kernel 2.6 portiert (*http://bender.thinknerd.de/~thomas/IPsec/isakmpd-linux.html*). Er wird in einem eigenen Kapitel besprochen. Die in diesem Kapitel besprochenen KAME Tools verwenden zwei Befehle: `setkey` und `racoon`. Sie sind von Alexey Kuznetsov in das *iputils* Paket übertragen worden. Die aktuellste Version wird im Moment unter *http://ipsec-tools.sf.net* veröffentlicht. Der Autor hat ein RPM Paket dieses Paketes auf seiner Homepage veröffentlicht (*https://www.spenneberg.org/VPN/Kernel-2_6-IPsec*). Wenn die Übersetzung selbst durchgeführt werden soll, so ist im Moment darauf zu achten, dass hierzu der Sourcecode eines Linux Kernels 2.5 oder 2.6 verfügbar sein muss. Der Ort muss im Makefile der Bibliothek `libipsec` und der Befehle `setkey` und `racoon` angegeben werden.

6.4 Konfiguration mit setkey und racoon

Die Konfiguration unterscheidet sich sehr stark von der Konfiguration mit FreeS/WAN.

Die IPsec Konfiguration erfolgt mit dem Befehl `setkey`. Dieser Befehl liest dazu die Angaben aus einer Datei oder von der Standardeingabe. Da die Verwendung von Dateien sinnvoller und weniger fehlerträchtig ist, wird im weiteren diese Methode genutzt.

Zunächst soll jedoch der Befehl `setkey` vorgestellt werden.

6.4.1 Das Kommando setkey

Das Kommando `setkey` ist das zentrale Kommando für die Konfiguration von IPsec im Linux Kernel 2.6. Es entspricht dem Kommando `ipsec` bei FreeS/WAN. Jedoch verwendet es eine andere Syntax und Konfigurationsdatei. Mit dem Kommando kann die Security Association Database (SAD) und die Security Policy Database (SPD) direkt editiert werden. Dies ist bei FreeS/WAN mit keinem Werkzeug so möglich.

Das Kommando akzeptiert die folgenden Optionen:

- **`-D`** Diese Option gibt sämtliche Einträge der SAD aus (Dump). Wenn zusätzlich die Option `-P` angegeben wird, so werden die Einträge der SPD ausgegeben.
- **`-F`** Diese Option löscht sämtliche Einträge der SAD (Flush). Wenn zusätzlich die Option `-P` angegeben wird, so werden die Einträge der SPD gelöscht.
- **`-P`** Bei Angabe dieser Option beziehen sich die Befehle auf die SPD anstelle der SAD.
- **`-a`** Üblicherweise werden »tote« Einträge der SAD nicht angezeigt. »Tote« Einträge sind in ihrer Gültigkeit abgelaufen, werden aber noch von SPD Einträgen referenziert. Die Option zeigt auch diese Einträge an.
- **`-d`** Debugging
- **`-x`** Gibt sämtliche Nachrichten des PF_KEY Kommunikationskanals aus.
- **`-h`** Die Ausgabe der PF_KEY Nachrichten (`-x`) erfolgt hexadezimal.
- **`-l`** Diese Option kann im Zusammenhang mit der Option `-D` verwendet werden, um die Ausgabe in einer Endlosschleife zu ermöglichen.
- **`-v`** Verbose
- **`-f datei`** Liest die durchzuführenden Operationen aus der angegebenen Datei
- **`-c`** Liest die durchzuführenden Operationen von der Standardeingabe

Von diesen Optionen ist die Option `-f` die wichtigste. Mit ihr ist es möglich, die durchzuführenden Operationen aus einer Datei einzulesen. Eine typische Datei ist in Listing 6.3 dargestellt.

```
#!/usr/sbin/setkey -f

# Lösche die SAD und SPD
flush;
spdflush;
```

```
# manuelle Parameter für AH SAs
add 3.0.0.1 5.0.0.1 ah 0x200 -A hmac-md5
0xbf9a081e7ebdd4fa824c822ed94f5226;
add 5.0.0.1 3.0.0.1 ah 0x300 -A hmac-md5
0xbf9a081e7ebdd4fa824c822ed94f5226;

# manuelle Parameter für ESP SAs
add 3.0.0.1 5.0.0.1 esp 0x201 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;
add 5.0.0.1 3.0.0.1 esp 0x301 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;

# Richtlinien zur Verwendung der SAs
spdadd 3.0.0.1 5.0.0.1 any -P out ipsec
           esp/transport//require
           ah/transport//require;

spdadd 5.0.0.1 3.0.0.1 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

Listing 6.3 Aufbau einer manuell verschlüsselten Verbindung mit `setkey`

Hierbei unterstützt der Befehl `setkey` die folgenden Operationen, die hier in alphabetischer Reihenfolge genannt werden:

- **`add`** *`src dst protocol spi [extensions] algorithm;`*

  Hiermit wird ein Eintrag zur SA Datenbank hinzugefügt. Dabei ist es erforderlich, die Source- und Destination IP Adresse als IPv4 Adresse anzugeben. Es ist nicht möglich hier Rechnernamen zu verwenden. `setkey` führt keine Namensauflösung durch. Das anzugebende Protokoll kann entweder `esp` oder `ah` sein.[2] An das Protokoll schließt sich der Security Parameter Index an. Dies ist eine dezimale oder hexadezimale Zahl von 255 bis -65535, beziehungsweise von 0x100 bis 0xffff). *`[extensions]`* sind optionale Erweiterungen. Linux unterstützt

  – **`-m mode`** Angabe des Modus: `tunnel` oder `transport`. Transport Modus ist der Default Modus, das bedeutet, dass für eine Tunnel SA der Tunnel Modus speziell angegeben werden muss![3]

2. Auf BSD Betriebsystemen wird hier zusätzlich `esp-old` und `ah-old` und `ipcomp` unterstützt. Die Option `ipcomp` sollte bei Erscheinen des Buches auch unter Linux unterstützt werden.
3. Achtung! Dies unterscheidet die Linux Implementierung von der KAME Implementierung. Bei Linux kann eine SA entweder nur für den Tunnel- oder den Transport Modus genutzt werden. KAME unterstützt jedoch zusätzlich den Mode `any`, der beide Modi ermöglicht.

- **`-r size`** Größe des Schiebefenster für den Schutz vor Replay Angriffen (Anti Replay Service). Hier kann eine dezimale Zahl von 0 bis 32 angegeben werden. Wird keine Zahl angegeben, so ist der Default Wert 32 für AH und ESP mit Authentifizierung. ESP ohne Authentifizierung verwendet einen Default Wert von 0.[4]
- **`-lh time`**, **`-ls time`** Diese Angaben erlauben die Definition der Lebensdauer der SA (hard/soft).

Nun muss für die Definition der SA noch der Algorithmus und der zu verwendende Schlüssel angegeben werden. Hierbei wird der Verschlüsselungsalgorithmus und Schlüssel mit `-E algo key` angegeben. Dabei sind die folgenden Algorithmen möglich: `des-cbc` und `3des-cbc`. Der Authentifizierungsalgorithmus wird mit `-A algo key` angegeben. Hierbei sind `hmac-md5` und `hmac-sha1` möglich. Diese Liste wird in nächster Zukunft erweitert werden um `simple` (keine Verschlüsselung), `blowfish-cbc`, `aes-cbc`, `hmac-sha2-256` und `hmac-sha2-512`. Auch eine Kompression der Daten vor der Verschlüsselung ist bereits möglich (`-C algo`). Hierbei wird mit Stand des Linux Kernel 2.5.69 jedoch nur der Algorithmus `deflate` unterstützt. Der Schlüssel kann in allen Fällen als hexadezimale Zahl (mit 0x...) oder als Zeichenkette in doppelten Anführungszeichen angegeben werden. Hierbei ist darauf zu achten, dass der Schlüssel die richtige Länge aufweist. Das Protokoll AH erlaubt lediglich eine Authentifizierung (`-A algo key`). Das Protokoll ESP erlaubt sowohl eine Authentifizierung als auch eine Verschlüsselung. Die Angabe der Verschlüsselung ist immer zwingend erforderlich. Zusätzlich kann dann auch die Authentifizierung spezifiziert werden. Dabei ist die Reihenfolge `-E algo key -A algo key` zwingend vorgeschrieben (siehe Listing 6.7).

- **`delete`** *`src dst protocol spi;`*

  Hiermit kann ein vorhandener Eintrag der SAD gelöscht werden. Dabei ist es erforderlich, die Source und Destination IP Adresse, das Protokoll und die SPI zur genauen Identifizierung des Eintrages anzugeben.

- **`deleteall`** *`src dst protocol;`*

  Diese Operation erlaubt es alle zutreffenden Einträge in der SAD zu löschen.

- **`dump`** *`[protocol];`*

  Dieser Befehl gibt alle auf das angegebene Protokoll zutreffenden Einträge der SAD aus.

4. Die BSD Implementierung KAME verwendet hier immer einen Defaultwert von 0!

- **flush** *[protocol];*

  Dieser Befehl löscht alle zutreffenden Einträge der SAD.

- **get** *src dst protocol spi;*

  Dieser Befehl zeigt den angegebenen Eintrag aus der SAD an. Die hierbei anzugebenden Werte wurden bereits bei der `add` Operation erklärt.

- **spdadd** *src_range dst_range upperspec policy;*

  Mit diesem Befehl kann ein Eintrag zur Security Policy Database hinzugefügt werden. Diese Datenbank definiert, wann welche SA eingesetzt werden soll. Um diese Zuordnung zu erzeugen ist es erforderlich zunächst die Rechner anzugeben, deren Kommunikation durch eine SA geschützt werden soll. Hierbei kann immer nur eine Richtung angegeben werden, und zwar in nummerischer Form. Vier verschiedene Formen sind möglich:

  – `IP Adresse`
  – `IP Adresse/Netzmaske` Hierbei muss die Netzmaske in CIDR Notation angegeben werden, zum Beispiel `/24`.
  – `IP Adresse[Port]` Die Angabe der eckigen Klammern ist erforderlich. Hier kann auch das Schlüsselwort `[any]` verwendet werden.
  – `IP Adresse/Netzmaske[Port]`

  Die Angabe *upperspec* definiert das Protokoll, das geschützt werden soll. Hier kann entweder der Name des Protokolls (`/etc/protcols`) oder seine Nummer verwendet werden. Als Wildcard kann das Schlüsselwort `any` genutzt werden.

  Schließlich muss noch die *policy* mit der Option `-P` definiert werden. Es existieren drei verschiedene Policies:

  – ***direction* discard**

    Der IPsec Code verwirft alle zutreffenden Pakete.

  – ***direction* none**

    Der IPsec Code führt keinerlei Veränderungen an den Paketen durch, er ignoriert sie.

  – ***direction* ipsec *protocol/mode/src-dst/level***

    Der IPsec Code erkennt und bearbeitet das Paket.

  Der Platzhalter *direction* erlaubt die Angabe der Richtung mit drei verschiedenen Werten: `in`, `out` oder `fwd`. Um die Pakete mit IPsec zu verschlüsseln und/oder zu authentifizieren, wird die Policy `ipsec` benötigt.

Die Angabe *protocol* definiert das Protokoll `ah` oder `esp`.[5] Mit der Angabe *mode* kann entweder der `transport` oder der `tunnel` Mode aktiviert werden. Beim Tunnel Mode ist es zwingend erforderlich, die IP Adressen der SA als *src* und *dst* getrennt durch ein Minus-Zeichen anzugeben. Beim Transport Mode kann diese Angabe wegfallen.

Die Angabe *level* erlaubt die Angabe `use` und `require`. Der Kernel versucht eine SA zu verwenden, wenn `use` angegeben wurde. Existiert keine SA, so werden die Pakete ohne IPsec Behandlung versandt und empfangen. Bei der Angabe `require` verlangt der Kernel eine SA. Existiert keine SA, so fordert der Kernel sie vom Key Exchange Daemon (`racoon`, siehe 6.4.3, »Automatische Verbindung mit `racoon`«) an, wenn er läuft.

- **spddelete** *src_range dst_range upperspec* **-P** *direction;*

  Hiermit kann ein spezieller Eintrag aus der SPD gelöscht werden. Die Parameter wurden bereits bei `spdadd` erläutert.

- **spddump**;

  Dieses Kommando zeigt alle Einträge der SPD an.

- **spdflush**;

  Dieses Kommando löscht alle Einträge der SPD.

ACHTUNG

Der Befehl `setkey` führt exakt die vom Benutzer angeforderten Operationen aus. Hierbei überprüft er die Angaben nicht auf ihre RFC Konformität. Dies ist Aufgabe des Benutzers!

Mit diesen Operationen ist es nun möglich, sämtliche Arten von IPsec Verbindungen aufzubauen. Im Folgenden wird der Aufbau von manuell verschlüsselten Verbindungen mit `setkey` und automatisch verschlüsselten Verbindungen mit `setkey` und `racoon` beschrieben. Die Vorstellung von `racoon` erfolgt dann im entsprechenden Kapitel (6.4.3, »Automatische Verbindung mit `racoon`«).

6.4.2 Manuelle Verbindung

Dieses Kapitel beschreibt den Aufbau einer manuellen Verbindung mit dem Kommando `setkey`. Dies ist das zentrale Kommando für die Verwaltung der Security Associations Database (SAD) und der Security Policy Database (SPD) beim IPsec Stack des Linux Kernels 2.6. Dieses Werkzeug erlaubt dabei

5. Zukünftige Versionen werden hier auch `ipcomp` unterstützen.

einen direkten Zugriff auf diese Informationen. Ein derartiger Zugriff ist bei FreeS/WAN mit dem Werkzeug `ipsec` nicht möglich. Diese Datenbank wird bei FreeS/WAN komplett vor dem Benutzer verborgen.

Bei einer manuell verschlüsselten Verbindung definiert der Administrator, die für die Verbindung erforderlichen Schlüssel manuell. Hierbei handelt es sich um die Schlüssel für die Authentifizierung und Verschlüsselung der Pakete und die anzuwendenden Protokolle AH oder ESP. Bei einer automatisch verschlüsselten Verbindung ist es Aufgabe des IKE Protokolls, diese Schlüssel zwischen den beiden VPN Partnern auszutauschen. Dabei kann das IKE Protokoll einen regelmäßigen Wechsel der Schlüssel garantieren. Bei einer manuell verschlüsselten Verbindung ist es Aufgabe des Administrators diese Aufgabe wahrzunehmen und für den Schlüsselaustausch zu sorgen. Ist eine dritte Partei in der Lage die verwendeten manuellen Schlüssel zu ermitteln, so kann sie alle versendeten und empfangenen Nachrichten entschlüsseln!

Aus diesem Grund soll hier nochmals darauf hingewiesen werden, dass eine automatisch verschlüsselte Verbindung mit dem IKE Protokoll eine wesentlich höhere Sicherheit bei gleichem oder sogar geringerem Administrationsaufwand bedeutet. Der Aufbau einer manuell verschlüsselten Verbindung sollte daher nur zu Testzwecken und bei einer Inkompatibilität des IKE Protokolls genutzt werden.

Der hier verwendete Testaufbau ist im Kapitel »Testumgebungen« ausführlich beschrieben. Dort finden sich auch Hinweise, wie sie mit VMware oder User Mode Linux aufgebaut werden können.

Manuelle Verbindung im Transport Modus

Das Kommando `setkey` verfügt bisher nicht über eine feste Konfigurationsdatei. Der Name und der Ort der Konfigurationsdatei sind frei wählbar. Die Linux Distributoren werden hier sicherlich Abhilfe schaffen. Um jedoch eine Verwechslung mit der Konfigurationsdatei `/etc/ipsec.conf`[6] von FreeS/WAN zu vermeiden, soll im weiteren der Name `/etc/setkey.conf` genutzt werden.

Diese Datei soll nun erzeugt werden, um eine manuell verschlüsselte Verbindung im Transport Modus aufzubauen. Diese Verbindung soll den Rechnern 3.0.0.1 und 5.0.0.1 die verschlüsselte und authentifizierte Kommunikation erlauben. Dabei erzeugen die Rechner im Transport Modus direkt die entsprechenden IPsec Pakete. Es erfolgt keine Kapselung von IP Paketen in IPsec Paketen wie beim Tunnel Modus. Das Endergebnis der Konfigurationsdatei für den Rechner 3.0.0.1 ist in Listing 6.4 bereits dargestellt.

6. Unter BSD wird meist dieser Name genutzt.

```
# Loesche die SAD und SPD
flush;
spdflush;

# manuelle Parameter fuer AH SAs
# add src dst proto spi -A authalgo key;
add 3.0.0.1 5.0.0.1 ah 700 -A hmac-md5 0xbf9a081e7ebdd4fa824c822ed94f5226;
add 5.0.0.1 3.0.0.1 ah 800 -A hmac-md5 0xbf9a081e7ebdd4fa824c822ed94f5226;

# manuelle Parameter fuer ESP SAs
# add src dst proto spi -E encalgo key;
add 3.0.0.1 5.0.0.1 esp 701 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;
add 5.0.0.1 3.0.0.1 esp 801 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;

# Richtlinien zur Verwendung der SAs
# spdadd src-range dst-range upperspec policy;
spdadd 3.0.0.1 5.0.0.1 any -P out ipsec
           esp/transport//require
           ah/transport//require;

spdadd 5.0.0.1 3.0.0.1 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

Listing 6.4 Manuelle Verbindung im Transport Mode (3.0.0.1)

In jedem `setkey` Skript sollten zu Beginn die Kommandos `flush` und `spdflush` aufgerufen werden. Hiermit werden die Security Associations Database und die Security Policy Database komplett gelöscht. Ansonsten besteht die Gefahr, dass alte Einträge mit der neuen Konfiguration kollidieren. Jede Zeile in dieser Konfigurationsdatei muss mit einem Semikolon abgeschlossen werden!

Anschließend werden die manuellen SAs erzeugt. Eine Security Association ist immer unidirektional und nur gültig für ein IPsec Protokoll. Wenn das AH Protokoll in beiden Richtungen genutzt werden soll, so werden zwei Security Associations benötigt. Die beiden nächsten Zeilen erzeugen die entsprechenden SAs für das AH Protokoll und die anschließenden beiden Zeilen für das ESP Protokoll (`add`). Jede SA benötigt einen eigenen SPI, mit dem sie eindeutig identifiziert werden kann. Zusätzlich ist die Angabe des Protokolls (`esp|ah`) erforderlich. ESP unterstützt sowohl Authentifizierung als auch eine Verschlüsselung der Pakete, AH lediglich die Authentifizierung. Hierfür ist die Angabe des Protokolls und des manuell erzeugten Schlüssels wichtig. Dieser Schlüssel kann als hexadezimale Zahl mit dem Prefix `0x` oder als Zeichenkette in Anführungszeichen angegeben werden.

Wurden die SAs erzeugt, müssen nun die Security Policies erzeugt werden, die definieren, wann die SAs eingesetzt werden müssen. Die Security Policies werden mit dem Befehl `spdadd` erzeugt, der von `setkey` gelesen wird. Dieser Befehl benötigt die Angabe der IP Adressen, deren Pakete mit IPsec behandelt werden sollen. In diesem Fall handelt es sich um den Transport Mode. Daher kommunizieren die Rechner 3.0.0.1 und 5.0.0.1 direkt miteinander. Diese IP Adressen müssen daher als `src-range` und `dst-range` angegeben werden. An die IP Adressen schließt sich das zu schützende Protokoll an. In den meisten Fällen wird hier die Angabe `any` richtig sein. Jedoch besteht die Möglichkeit lediglich `tcp` oder `icmp` Pakete mit IPsec zu schützen.

Der wichtigste Teil der Definition der Security Policy ist die Policy selbst. Hier wird definiert, aus welcher Richtung die Policy die Pakete betrachten soll und ob eine Behandlung der Pakete erfolgen soll. Die Richtung kann mit `in|out` und die Behandlung mit `discard|none|ipsec` angegeben werden. Um eine Verschlüsselung und Authentifizierung zu erreichen, ist die Angabe von `ipsec` erforderlich. Auch die Security Policy ist lediglich unidirektional. Daher sind für eine richtige Verbindung immer zwei Security Policies erforderlich: Eine für die Richtung `in` und eine für die Richtung `out`. Nun können die SAs angegeben werden. Diese SAs werden über ihr Protokoll und die IP Adressen (*`protocol/mode/src-dst/level`*) referenziert. Im Transport Modus handelt es sich um dieselben IP Adressen, die bereits als `src-range dst-range` angegeben wurden. Daher kann die Angabe hier entfallen. Vergleichen Sie dies mit der Angabe im nächsten Abschnitt! Die Angabe des `level` definiert, ob eine Verschlüsselung lediglich erwünscht (`use`) oder zwingend erforderlich (`require`) ist.

In einer Security Policy kann bei der Verwendung des Transport Mode sowohl eine ESP als auch eine AH SA angegeben werden. Wenn dies gewünscht wird, ist aber die Reihenfolge wichtig. RFC 2401 verlangt, dass zuerst das ESP und anschließend das AH Protokoll auf das Paket angewendet wird. Daher ist diese Reihenfolge auch bei der Definition der Policy zu wählen.

ACHTUNG

Der Linux Kernel erzwingt diese Reihenfolge nicht und erlaubt so die Erzeugung nicht RFC-konformer IPsec Verbindungen!

Nach der Datei für den Rechner 3.0.0.1 muss eine analoge Datei für den Rechner 5.0.0.1 erzeugt werden (Listing 6.5). Hierbei ist die unterschiedliche Sichtweise des Rechners wichtig. Es ist nicht möglich die Datei 1:1, wie bei FreeS/WAN, zu übernehmen.

```
# Loesche die SAD und SPD
flush;
spdflush;

# manuelle Parameter fuer AH SAs
# add src dst proto spi -A authalgo key;
add 3.0.0.1 5.0.0.1 ah 700 -A hmac-md5 0xbf9a081e7ebdd4fa824c822ed94f5226;
add 5.0.0.1 3.0.0.1 ah 800 -A hmac-md5 0xbf9a081e7ebdd4fa824c822ed94f5226;

# manuelle Parameter fuer ESP SAs
# add src dst proto spi -E encalgo key;
add 3.0.0.1 5.0.0.1 esp 701 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;
add 5.0.0.1 3.0.0.1 esp 801 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;

# Richtlinien zur Verwendung der SAs
# spdadd src-range dst-range upperspec policy;
spdadd 3.0.0.1 5.0.0.1 any -P in ipsec
           esp/transport//require
           ah/transport//require;

spdadd 5.0.0.1 3.0.0.1 any -P out ipsec
           esp/transport//require
           ah/transport//require;
```

Listing 6.5 Manuelle Verbindung im Transport Mode (5.0.0.1)

Die notwendigen Änderungen der Datei für den Rechner 5.0.0.1 sind im Listing 6.5 fett markiert. Die Richtungen der Security Policies müssen ausgetauscht werden.

Wurde die Konfiguration auf beiden Systemen erzeugt, so sollte zunächst überprüft werden, ob die Netzwerkkonnektivität zwischen beiden Systemen gewährleistet ist. Hierzu genügt ein simples Ping Kommando. Besteht keine Netzwerkkonnektivität, so kann sie durch IPsec sicherlich nicht erzeugt werden!

Anschließend werden auf beiden Rechnern die Security Association Database und Security Policy Database mit dem Kommando `setkey -f /etc/setkey.conf` gefüllt. Wurde der Befehl auf beiden Rechnern ausgeführt, so sollte ein `ping` möglich sein. Ein Mitschnitt mit tcpdump zeigt aber nur noch verschlüsselte Pakete an:

```
12:45:39.373005 5.0.0.1 > 3.0.0.1: AH(spi=0x00000320,seq=0x1):
ESP(spi=0x00000321,seq=0x1) (DF)
12:45:39.448636 3.0.0.1 > 5.0.0.1: AH(spi=0x000002bc,seq=0x1):
ESP(spi=0x000002bd,seq=0x1)
```

```
12:45:40.542430 5.0.0.1 > 3.0.0.1: AH(spi=0x00000320,seq=0x2):
ESP(spi=0x00000321,seq=0x2) (DF)
12:45:40.569414 3.0.0.1 > 5.0.0.1: AH(spi=0x000002bc,seq=0x2):
ESP(spi=0x000002bd,seq=0x2)
```

Listing 6.6 Mitschnitt des verschlüsselten Pings

Der Aufbau der manuellen Verbindung im Transport Mode ist recht einfach. Im Vergleich zu FreeS/WAN fällt auf, dass die Konfigurationsdatei jeweils aus der Sicht des VPN Partners erstellt werden muss. Nach einer Kopie der Datei ist eine Anpassung der Richtungen der Security Policies erforderlich.

Manuelle Verbindung im Tunnel Modus

Im IPsec Tunnel Modus werden die IP-Pakete komplett in ein IPsec-Paket gepackt. Dies ist der Standardmodus von vielen IPsec Implementierungen , so auch von FreeS/WAN. Dieser Modus kann auch genutzt werden, um zwei Netzwerken die Kommunikation über ein VPN zu erlauben. Dabei werden die kompletten IP-Pakete vom ersten VPN Gateway in ein IPsec Paket verpackt, das zum Partner Gateway transportiert wird. Dort wird das IPsec Paket wieder ausgepackt und das Original Paket weitergeleitet. Es ensteht ein echter Tunnel, daher auch der Name.

Die fertige Konfigurationsdatei für einen einfachen Tunnel ist in Listing 6.7 dargestellt:

```
#!/usr/sbin/setkey -f

# Loesche die SAD und SPD
flush;
spdflush;

# manuelle Parameter fuer ESP Authentifizierung und Verschluesselung
add 5.0.0.1 3.0.0.1  esp 0x200 -m tunnel -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada -A hmac-md5
0xbf9a081e7ebdd4fa824c822ed94f5226;
add 3.0.0.1 5.0.0.1  esp 0x200 -m tunnel -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada -A hmac-md5
0xbf9a081e7ebdd4fa824c822ed94f5226;

# Richtlinien zur Verwendung der SAs (Tunnel Newyork-Berlin)
spdadd 3.0.0.1 5.0.0.1 any -P out ipsec
           esp/tunnel/3.0.0.1-5.0.0.1/require;

spdadd 5.0.0.1 3.0.0.1 any -P in ipsec
           esp/tunnel/5.0.0.1-3.0.0.1/require;
```

Listing 6.7 Manuelle Verbindung im Tunnel Mode (3.0.0.1)

Bei dieser manuellen Verbindung im Tunnel Modus wurde ein anderer Ansatz gewählt, als bei der manuellen Verbindung im Transport Modus (letzter Abschnitt). Hier erfolgt die Authentifizierung der Pakete mit dem ESP Protokoll und nicht mit dem AH Protokoll. Häufig wird in einem VPN das AH-Protokoll gar nicht eingesetzt, sondern lediglich das ESP-Protokoll. Aber der Reihe nach.

Zunächst werden wieder die SAD und die SPD mit den Befehlen `flush` und `spdflush` gelöscht. Anschließend werden die SAs erzeugt. Hierbei werden zwei SAs für das Protokoll ESP mit identischem SPI erzeugt. Der SPI wird hexadezimal mit 0x200 angegeben. Da es sich um eine SA mit Tunnel Modus handeln soll, ist es erforderlich, sie mit der Option `-m tunnel` zu spezifizieren. Wird dies nicht angegeben, so handelt es sich immer um eine SA für den Transport Modus. Anschließend werden der Algorithmus und der Schlüssel für die Verschlüsselung und die Authentifizierung angegeben. Diese Reihenfolge ist zwingend vorgeschrieben. Es ist nicht möglich zunächst die Authentifizierung und dann die Verschlüsselung zu definieren.

Bei der Wahl der eingetragenen Werte wurde im Beispiel darauf geachtet, dass sie den Werten der manuellen FreeS/WAN Verbindung aus Abschnitt 5.5.3, »Manuelle Verbindung« entsprechen.

Nun müssen noch die Einträge in die SPD vorgenommen werden. Sie unterscheiden sich von der Definition im Transport Modus nur durch die zusätzliche Angabe der Source und Destination IP Adresse der zu verwendenden SA. Warum diese Angabe erforderlich ist, wird im weiteren Verlauf deutlich werden.

Wenn so die Verbindung auf Rechner 3.0.0.1 erzeugt wurde, ist es nun erforderlich, die entsprechende Datei für den Rechner 5.0.0.1 zu erzeugen. Um die gesamte Konfiguration hier nicht noch einmal anzuzeigen, soll nur darauf hingewiesen werden, dass die Richtungen der Security Policies (`in,out`) ausgetauscht werden müssen.

Wird nun die Konfiguration mit `setkey -f /etc/setkey.conf` aktiviert, so wird der nachfolgende Ping verschlüsselt übertragen:

```
13:55:04.084453 3.0.0.1 > 5.0.0.1: ESP(spi=0x00000200,seq=0x1) (DF)
13:55:04.116751 5.0.0.1 > 3.0.0.1: ESP(spi=0x00000200,seq=0x1)
13:55:05.316298 3.0.0.1 > 5.0.0.1: ESP(spi=0x00000200,seq=0x2) (DF)
13:55:05.348958 5.0.0.1 > 3.0.0.1: ESP(spi=0x00000200,seq=0x2)
```

Da bei der Wahl der Konfigurationsparameter darauf geachtet wurde, dass sie mit den entsprechenden Parametern in Listing 5.23 übereinstimmen, ist die Interoperabilität gewährleistet und kann getestet werden. Der Rechner 3.0.0.1 wurde bei FreeS/WAN als NewYork und der Rechner 5.0.0.1 als Berlin bezeichnet.

Erweiterungen und Anmerkungen

Diese manuelle Verbindung sichert nun den Verkehr zwischen 3.0.0.1 (NewYork) und 5.0.0.1 (Berlin). Wenn sich hinter diesen Gateways weitere Netze befinden, die sich ebenfalls verschlüsselt unterhalten sollen, so werden hierfür weitere eigene Tunnel benötigt. Die Anzahl der aufgesetzten Tunnel ist relativ unkritisch. Es ist möglich mehrere Hundert Tunnel zu definieren. Diese Tunnel müssen als Security Policies in die Datenbank mit dem `setkey` Kommando eingetragen werden.

Hier soll daher nur ein weiterer Tunnel beschrieben werden, der zwei Subnetze hinter diesen Gateways verbindet. Die verwendeten IP Adressen und den Aufbau der Testumgebung können Sie wieder dem entsprechenden Kapitel 11, »Testumgebungen« entnehmen.

Der Tunnel `NewYorkNet-BerlinNet` wird im Listing 6.8 beschrieben.

```
#!/usr/sbin/setkey -f

# Loesche die SAD und SPD
flush;
spdflush;

# manuelle Parameter fuer ESP Authentifizierung und Verschluesselung
add 5.0.0.1 3.0.0.1  esp 0x200 -m tunnel -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada -A hmac-md5
0xbf9a081e7ebdd4fa824c822ed94f5226;
add 3.0.0.1 5.0.0.1  esp 0x200 -m tunnel -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada -A hmac-md5
0xbf9a081e7ebdd4fa824c822ed94f5226;



# Richtlinien zur Verwendung der SAs (Tunnel Newyork-Berlin)
spdadd 3.0.0.1 5.0.0.1 any -P out ipsec
           esp/tunnel/3.0.0.1-5.0.0.1/require;

spdadd 5.0.0.1 3.0.0.1 any -P in ipsec
           esp/tunnel/5.0.0.1-3.0.0.1/require;
```

```
# Richtlinien zur Verwendung der SAs (Tunnel NewYorkNet-BerlinNet)
# spdadd src-range dst-range upperspec policy;
spdadd 10.0.1.0/24 10.0.2.0/24 any -P out ipsec
          esp/tunnel/3.0.0.1-5.0.0.1/require;

spdadd 10.0.2.0/24 10.0.1.0/24 any -P in ipsec
          esp/tunnel/5.0.0.1-3.0.0.1/require;
```

Listing 6.8 Manuell verschlüsselte Verbindung mit `setkey`: Subnet-Subnet

Die Datei unterscheidet sich von Listing 6.7 nur durch zwei zusätzliche Einträge in der Security Policy Database. Sie definieren, dass bei Paketen von 10.0.1.0/24 nach 10.0.2.0/24 die angegebene SA zu verwenden ist. Nun wird deutlich, warum das Kommando `spdadd` zweimal die Angabe einer Source und Destination Adresse bei einem Tunnel verlangt. Die erste Angabe definiert die zu transportierenden Pakete. Die zweite Angabe spezifiziert die Tunnelendpunkte und darüber die zu verwendende SA. Bei der Definition der SA werden lediglich die Tunnelendpunkte definiert. Über diese SA können die unterschiedlichen Pakete transportiert werden!

Wurden die Rechner 3.0.0.1 (NewYork) und 5.0.0.1 (Berlin) dementsprechend konfiguriert, also auf Berlin wieder die Richtung der SPD Einträge angepasst, so kann der Inhalt der SAD (Listing 6.9) und der SPD (Listing 6.10) zunächst angezeigt werden. Anschließend kann dann ein `ping` aus dem NewYorkNet in das BerlinNet durchgeführt werden (Abbildung 6.1).

```
# setkey -D
3.0.0.1 5.0.0.1
        esp mode=tunnel spi=512(0x00000200) reqid=0(0x00000000)
        E: 3des-cbc  3f0b868a d03e68ac c6e4e464 4ac8bb80 ecea3426 d3d30ada
        A: hmac-md5  bf9a081e 7ebdd4fa 824c822e d94f5226
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 20 18:34:06 2003   current: Feb 20 18:36:30 2003
        diff: 144(s)    hard: 0(s)      soft: 0(s)
        last: Feb 20 18:34:16 2003      hard: 0(s)      soft: 0(s)
        current: 1088(bytes)    hard: 0(bytes)  soft: 0(bytes)
        allocated: 8    hard: 0 soft: 0
        sadb_seq=1 pid=573 refcnt=0
5.0.0.1 3.0.0.1
        esp mode=tunnel spi=512(0x00000200) reqid=0(0x00000000)
        E: 3des-cbc  3f0b868a d03e68ac c6e4e464 4ac8bb80 ecea3426 d3d30ada
        A: hmac-md5  bf9a081e 7ebdd4fa 824c822e d94f5226
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 20 18:34:06 2003   current: Feb 20 18:36:30 2003
        diff: 144(s)    hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
```

```
        current: 672(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 8    hard: 0 soft: 0
        sadb_seq=0 pid=573 refcnt=0
```

Listing 6.9 Inhalt der SAD auf 3.0.0.1

```
# setkey -DP
5.0.0.1[any] 3.0.0.1[any] any
        in ipsec
        esp/tunnel/5.0.0.1-3.0.0.1/require
        created:Feb 20 18:34:06 2003 lastused:Feb 20 18:34:18 2003
        lifetime:0(s) validtime:0(s)
        spid=8 seq=3 pid=574
        refcnt=5
10.0.2.0/24[any] 10.0.1.0/24[any] any
        in ipsec
        esp/tunnel/5.0.0.1-3.0.0.1/require
        created:Feb 20 18:34:06 2003 lastused:
        lifetime:0(s) validtime:0(s)
        spid=24 seq=2 pid=574
        refcnt=1
3.0.0.1[any] 5.0.0.1[any] any
        out ipsec
        esp/tunnel/3.0.0.1-5.0.0.1/require
        created:Feb 20 18:34:06 2003 lastused:Feb 20 18:34:18 2003
        lifetime:0(s) validtime:0(s)
        spid=1 seq=1 pid=574

        refcnt=3
10.0.1.0/24[any] 10.0.2.0/24[any] any
        out ipsec
        esp/tunnel/3.0.0.1-5.0.0.1/require
        created:Feb 20 18:34:06 2003 lastused:Feb 20 18:34:26 2003
        lifetime:0(s) validtime:0(s)
        spid=17 seq=0 pid=574
        refcnt=3
```

Listing 6.10 Inhalt der SPD auf 3.0.0.1

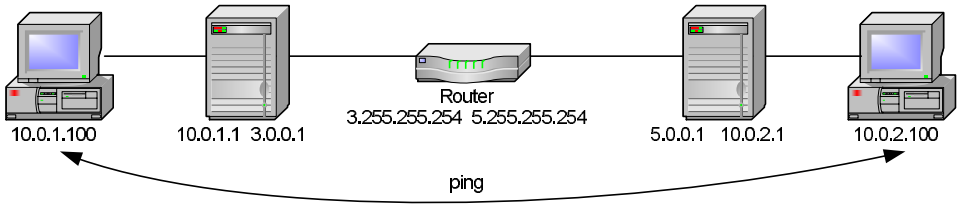


Abbildung 6.1 Ping aus dem NewYorkNet ins BerlinNet

Fazit

Der Aufbau einer manuell verschlüsselten Verbindung mit dem Kommando `setkey` ist sehr einfach durchzuführen. Hierbei können auch heterogene Lösungen beispielsweise mit FreeS/WAN aufgebaut werden. Wichtig ist jedoch bei einer manuell verschlüsselten Lösung, dass die Schlüssel besonders sicher aufbewahrt und zwischen den Partnern ausgetauscht werden. Erhält eine dritte Partei Zugang zu den manuellen Schlüsseln, so ist sie in der Lage sämtliche Pakete mitzulesen und zu entschlüsseln. Sie kann außerdem zusätzliche Pakete einschleusen oder vorhandene modifizieren.

Wenn immer möglich sollten daher automatisch verschlüsselte Verbindungen vorgezogen werden. Sie weisen eine wesentlich höhere Sicherheit auf, da die Sitzungsschlüssel für die Verschlüsselung und Authentifizierung automatisch vom IKE Protokoll ausgetauscht werden. Hierfür ist beim Linux Kernel 2.6 der IKE Daemon `racoon` des KAME Projektes verantwortlich.

6.4.3 Automatische Verbindung mit racoon

`racoon` ist der IKE Daemon des *KAME* Projektes, der von Dave Miller und Alexey Kuznetsov auf Linux portiert wurde. Er nimmt hier dieselbe Aufgabe wahr, die auch Pluto bei FreeS/WAN hat. Hierzu verwendet er das IKE Protokoll um eine ISAKMP SA auszuhandeln und anschließend die IPsec SAs zu erzeugen. Dabei wird `racoon` normalerweise nicht von selbst tätig, sondern bei Bedarf durch den Kernel angestoßen. Der Kernel kommuniziert hierzu mit `racoon` über den PF_KEY Socket. Jedes Mal, wenn der Kernel ein Paket entsprechend der Security Policy Database mit IPsec verschlüsseln oder authentifizieren muss, aber nicht über die entsprechende SA verfügt, fordert er `racoon` auf, diese SAs auszuhandeln und bereitzustellen.

Erforderlich für einen erfolgreichen Einsatz von `racoon` ist daher dessen Konfiguration in der Datei `/etc/racoon.conf` und zusätzlich die Konfiguration der SPD mit dem Befehl `setkey`.

Dieses Kapitel betrachtet zunächst `racoon` und dessen Konfiguration ganz allgemein. Anschließend wird die Konfiguration von `racoon` bei einer Authentifizierung mit einem Preshared Key (PSK) und X.509 Zertifikaten besprochen und durchgespielt.

Konfiguration von racoon

Der Kernel greift über den PF_KEY Socket auf `racoon` zu und fordert ihn auf mit dem IKE Protokoll die ISAKMP SA und die IPsec SAs auszuhandeln, wenn die Security Policies, die mit dem `setkey` Kommando erzeugt wurden,

es verlangen. Hierzu muss `racoon` als Dienst zur Verfügung stehen. Beim Start von `racoon` stehen eine ganze Reihe von Optionen zur Verfügung, mit denen dieser Start modifiziert werden kann. Sobald `racoon` gestartet wurde, kann er mit dem Werkzeug `racoonctl` administriert werden.

Für eine Einbindung in den Linux Startprozess bietet es sich an, für `racoon` ein SysV Startskript zu erzeugen und in die SysV Initscripts einzubinden. Eine Variante eines derartigen Skriptes ist in Listing 6.11 abbgebildet.

```
#! /bin/bash
#
# racoon          Start/Stop the racoon IKE daemon.
#
# chkconfig: 2345 90 60
# description: racoon is the IKE daemon of the KAME tools. Use it with \
#              the native Linux 2.6 IPsec stack

# processname: racoon
# config: /etc/racoon.conf
# pidfile: /var/run/racoon.pid

# Source function library.
. /etc/init.d/functions

OPTS=""

. /etc/sysconfig/racoon

RETVAL=0


prog="racoon"

start() {
        echo -n $"Starting $prog: "
        daemon racoon $OPTS
        RETVAL=$?
        echo
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/racoon
        return $RETVAL
}

stop() {
        echo -n $"Stopping $prog: "
        killproc racoon
        RETVAL=$?
        echo
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/racoon
```

```
        return $RETVAL
}

rhstatus() {
        status racoon
}

restart() {
        stop
        start
}

reload() {
        echo -n $"Reloading racoon daemon configuration: "
        killproc racoon -HUP
        retval=$?
        echo
        return $RETVAL

}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        restart
        ;;
  reload)
        reload
        ;;
  status)
        rhstatus
        ;;
  condrestart)
        [ -f /var/lock/subsys/crond ] && restart || :
        ;;
  *)
        echo $"Usage: $0 {start|stop|status|reload|restart|condrestart}"
        exit 1
esac

exit $?
```

Listing 6.11 SysV Startskript für `racoon`

Dieses Skript liest die `racoon` Startoptionen aus der Datei `/etc/sysconfig/racoon`, die zu diesem Zweck erzeugt werden sollte.

`racoon` unterstützt die folgenden Optionen beim Start:

- **`-B`** Hiermit wird `racoon` aufgefordert die SAs aus einer in der Konfigurationsdatei `racoon.conf` definierten Datei einzulesen und statisch in der SAD einzutragen (für Backup Zwecke).
- **`-d`** Erhöht den Detailgrad der Debugmeldungen. Je mehr `-d` angegeben werden, um so detailliert werden die Meldungen.
- **`-F`** `racoon` läuft im Vordergrund und gibt sämtliche Meldungen auf der Konsole aus.
- **`-f`** *`configfile`* Anstelle der Default Datei (`/etc/racoon.conf`) liest `racoon` die angegebene Konfigurationsdatei.
- **`-l`** *`logfile`* `racoon` protokolliert üblicherweise an den Syslog mit der Facility LOG_DAEMON und der Priority LOG_INFO. Bei Angabe dieser Option protokolliert `racoon` in der Datei.
- **`-p`** *`IKE port`* Der Default IKE Port ist 500/udp. Hiermit kann ein anderer UDP Port angegeben werden.
- **`-v`** Dies aktiviert eine ausführlichere Ausgabe (verbose).
- **`-4`** Hiermit wird ein IPv4 Socket erzeugt.
- **`-6`** Hiermit wird ein IPv6 Socket erzeugt.

In den meisten Fällen kann `racoon` ohne weitere Startoptionen gestartet werden. Lediglich zu Debugging Zwecken bietet es sich an den Detailgrad der Meldungen zu erhöhen und `racoon` im Vordergrund laufen zu lassen.

Die eigentliche Konfiguration von `racoon` erfolgt in der Konfigurationsdatei `/etc/racoon.conf`. Eine alternative Konfigurationsdatei kann beim Start mit der Option `-f` *`configfile`* angegeben werden.

Diese Konfigurationsdatei soll im Folgenden komplett vorgestellt werden, um einen Eindruck von der Mächtigkeit von `racoon` zu erhalten. Im Anschluss werden dann die Konfiguration von `racoon` mit PSK und X.509 Zertifikaten vorgestellt.

Wenn Ihnen diese Aufstellung zu langatmig und trocken erscheint, so können Sie ohne weiteres zunächst weiterblättern zum Abschnitt »`racoon` und Preshared Keys« und später diese Darstellung in Kombination mit dem Stichwortverzeichnis als Konfigurationsreferenz nutzen. Die unterstützten Zeiteinheiten sind: `sec, secs, second, seconds, min, mins, minute, minutes, hour, hours`.

Die Konfigurationsdatei unterstützt die folgenden Einträge:

- **path include** *pfad*; Hiermit kann ein Verzeichnis angegeben werden, in dem sich weitere einzulesende Dateien befinden (siehe `include` *datei*).
- **path pre_shared_key** *datei* In dieser Datei können PSKs gespeichert werden. Die Syntax dieser Datei wird weiter unten erläutert.
- **path certificate** *pfad*; Dieses Verzeichnis enthält sämtliche Zertifikate.
- **patch backupsa** *datei*; `racoon` wird jede erzeugte SA an diese Datei anhängen. So kann bei einem Neustart von `racoon` diese Datei mit der Option `-B` wieder eingelesen werden.
- **include** *datei*; Diese Datei wird zusätzlich aus dem mit `path include` angegebenen Verzeichnis als Konfigurationsdatei eingelesen.
- **timer { ... }** Mit dieser Direktive können mehrere Zeitgeber in geschweiften Klammern gesetzt werden. Es existieren die folgenden Zeitgeber mit ihren Standardwerten (in Klammern):
  - **counter** *zahl*; Maximale Anzahl von Retries (5)
  - **interval** *zahl zeiteinheit*; Das Zeitintervall, nachdem ein Retry durchgeführt werden soll (10 sec)
  - **persend** *zahl* Wie viele Pakete gesendet werden sollen (1).
  - **phase1** *zahl zeiteinheit* Maximale Zeit für den Aufbau der Phase 1 (15 sec)
  - **phase2** *zahl zeiteinheit* Maximale Dauer für den Aufbau der Phase 2 (10 sec)
- **listen { ... }** Mit dieser Direktive kann die IP Adresse und der Port für `racoon` definiert werden. Wird sie nicht verwendet, so bindet sich `racoon` an alle verfügbaren Netzwerkschnittstellen. Hierzu können in den geschweiften Klammern die folgenden Optionen angegeben werden:
  - **isakmp** *adresse[port]*; Nun horcht `racoon` nur noch auf die angegebene Adresse und den optional anzugebenden Port.
  - **strict_address**; Verlangt, dass sämtliche angegebenen Adressen gebunden sein müssen.
- **remote** (*address*|**anonymous**) [port] **{ ... }** Hiermit wird nun die wesentliche Konfiguration von `racoon` vorgenommen. Diese Direktive definiert den Kommunikationspartner und wie `racoon` mit ihm in Phase 1 kommunizieren soll. Hierbei ist die Angabe der IP Adresse des Partners erforderlich. Wird hier `anonymous` gewählt, so trifft diese Definition auf alle Partner zu, für deren IP Adresse keine andere `remote` Direktive existiert. Der Port kann optional angegeben werden, wenn es sich nicht um

den IKE Port 500/udp handelt. In geschweiften Klammern können nun genauere Angaben zu Phase 1 spezifiziert werden:

- **exchange_mode** (**main**|**aggressive**|**base**); racoon unterstützt in Phase 1 sowohl den Main, Aggressive als auch den Base Mode.[7] Diese Angabe definiert, welchen Mode racoon als Initiator benutzen soll und auf welchen Mode er antworten darf. Mehrere oder alle Modes können mit Komma getrennt angegeben werden.

- **doi ipsec_doi**; Hiermit wird entsprechend dem RFC 2407 die IPSEC Domain of Interpretation (DOI) genutzt. Dabei können Gruppierungen von verwandten Protokollen für die Verhandlung von ISAKMP SAs erzeugt werden. FreeS/WAN nutzt ebenfalls ipsec_doi. Die Angabe ist optional.
- **situation identity_only**; racoon verwendet SIT_IDENTITY_ONLY entsprechend RFC 2407. Diese Angabe ist optional.
- **my_identifier** *idtype*; Dieser Parameter gibt die Identität, die racoon dem Partner übermittelt. Dies entspricht bei FreeS/WAN dem Parameter (left|right)id. Beim *idtype* kann es sich handeln um:
  - **address** [*addresse*] Dies ist die Defaulteinstellung, wenn kein Identifier angegeben wurde.
  - **user_fqdn** *user@domain* Dies erlaubt die Angabe einer E-Mail Adresse.
  - **fqdn** *domain* Hier kann eine vollqualifizierte Domäne (FQDN) angegeben werden. Sie wird nicht über den DNS aufgelöst.
  - **keyid** *datei* Die Identität wird aus der angegebenen Schlüsseldatei gelesen.
  - **asn1dn** [*string*] Es wird der angegebene ASN.1 DN verwendet. Wenn kein *string* angegeben wird, so liest racoon die Information aus dem Subject Feld des Zertifikates.
- **peer_identifier ;** Hier kann die Identität des Partners definiert werden, wenn racoon sie prüfen soll. Ob die Identität geprüft wird, hängt jedoch zusätzlich von verify_identifier ab.
- **verify_identifier (on|off);** Nur wenn hier der Wert on gesetzt wurde, wird tatsächlich der Partner verifiziert. Verwendet er eine andere Identität, so schlägt der Verbindungsaufbau fehl. Default ist off.

7. Der Base Mode wird von FreeS/WAN nicht unterstützt und wurde im Draft *draft-ietf-ipsec-ike-base-mode* definiert. Dieser Draft wurde jedoch nie zum Standard erhoben.

- **certificate_type x509** *zertifikat privschlüssel*; Dieser Parameter spezifiziert das zu verwendende eigene Zertifikat. Die Angabe *zertifikat* enthält den Dateinamen des Zertifikates und *privschlüssel* den Dateinamen des dazu passenden privaten Schlüssels.
- **peers_certfile** (**dnssec**|datei); Normalerweise überträgt der Partner sein Zertifikat in Phase 1. Bei Angabe dieses Parameters ignoriert racoon das übertragene Zertifikat und ermittelt es entweder mit Hilfe von DNS oder liest es aus der angegebenen Datei.
- **send_cert (on|off)**; racoon sendet sein Zertifikat ebenfalls automatisch an den Partner. Dieser Parameter erlaubt es, das abzustellen.
- **send_cr (on|off)**; Damit racoon das Zertifikat des Partners erhält, fordert er es zunächst an. Hiermit kann diese Anforderung deaktiviert werden.
- **verify_cert (on|off)**; racoon überprüft das vom Partner erhaltene Zertifikat automatisch. Diese Verifizierung kann abgeschaltet werden. Achtung: Das erzeugt eine Sicherheitslücke!
- **lifetime time** *zahl einheit*; Hiermit kann die zu verwendende Lebensdauer für die ISAKMP SA der Phase 1 definiert werden.
- **initial_contact (on|off)**; Wenn der Linux Rechner mit racoon neu gestartet wird, und versucht eine SA neu aufzubauen, so wird der Partner möglicherweise mit einer alten SA antworten, die der Linux Rechner nicht mehr kennt. Ist initial_contact gesetzt, so sendet racoon eine Meldung, die die Gegenseite auffordert, die neue SA zu verwenden (Default: on).
- **passive (on|off)**; racoon versucht jede Verbindung selbst aktiv aufzubauen. Hiermit lädt racoon lediglich die Verbindung, so dass ein Client sie aufbauen kann. Sinnvoll für einen VPN Server mit Roadwarriors (vergleichbar mit auto=add bei FreeS/WAN).
- **proposal_check** *level*; Es gibt vier verschiedene Varianten, wie racoon auf die Vorschläge des Partners in Phase zwei reagieren kann: obey, strict, claim und exact. Default ist strict.
  - **obey** racoon nimmt immer den Vorschlag des Partners an.
  - **strict** racoon nimmt nur den Vorschlag des Partners an, wenn die vorgeschlagene Lebensdauer kürzer ist. Wenn von beiden Seiten PFS gefordert wird, so müssen die verwendeten Diffie Hellmann Gruppen übereinstimmen.
  - **claim** wie strict außer, dass racoon eine RESPONDER-LIFETIME Meldung sendet, wenn die eigene Lebensdauer kürzer ist als die des Partners.

  - **strict** Die Lebensdauer und die Parameter der PFS müssen exakt übereinstimmen.

- **generate_policy (on|off);** Wenn `racoon` passiv den Aufbau einer Verbindung erwartet, so kann `racoon` beim Aufbau die Einträge in der SPD entsprechend dem Wunsch des Clients vornehmen. Default ist `off`.
- **support_mip6 (on|off);**
- **nonce_size** *Zahl;* Größe des Nonces im Diffie Hellmann Schlüsselaustausch (8-256, default: 16)
- **proposal { ... }** Hier wird der Proposal für Phase 1 definiert. In geschweiften Klammern können die folgenden Angaben gemacht werden:
  - **encryption_algorithm** *algo;* Mögliche Werte sind `des`, `3des`, `blowfish`, `rijndal` und `cast128`. Die Angabe ist zwingend erforderlich.
  - **hash_algorithm** *algo;* Mögliche Werte sind `md5`, `sha1`, `sha2_256`, `sha2_385` und `sha2-512`. Die Angabe ist zwingend erforderlich.
  - **authentication_method** *methode;* Als Methode muss entweder `pre_shared_key`, `rsasig` oder `gssapi_krb` gewählt werden. Die letzte Methode verwendet für die Authentifizierung Kerberos.
  - **dh_group** *gruppe;* Hier kann die Diffie Hellmann Gruppe (MODP Gruppe) angegeben werden. Gültig sind `modp768`, `modp1024`, `ec2n155`, `ec2n185`, `modp1536`, `modp2048`, `modp3072`, `modp4096`, `modp6144` und `modp8192`.
  - **lifetime time** *zahl einheit;*
  - **gssapi_id** *identität;* Hier kann der zu verwendende GSS-API Name angegeben werden. Wird diese Angabe unterlassen, so wird der FQDN Rechnername verwendet.
- **sainfo** (*source destination*|**anonymous**) { ... } Hiermit können die Parameter der Phase 2 konfiguriert werden. *source* und *destination* können sowohl Adressen als Identitäten sein. Adressen werden angegeben als **address** *address[/prefix][port] proto*. Das Feld *proto* enthält hierbei das Protokoll, welches mit IPsec verschlüsselt/authentifiziert werden soll. Mit `any` wird jedes Protokoll verschlüsselt. Anstelle der Adresse kann jedoch auch eine Identität angegeben werden. Diese Identitäten folgen der selben Syntax wie bei der Direktive `my_identifier`.

  Im Detail kann die SA in den geschweiften Klammern konfiguriert werden. Hier werden folgende Parameter unterstützt:

  - **pfs_group** *group;* Hier muss die Diffie Hellmann Gruppe für PFS angegeben werden (mögliche Werte siehe oben).

  - **lifetime time** *zahl einheit*; Lebensdauer der IPsec SA
  - **encryption_algorithm** *algo*; Hier kann eine Komma-separierte Liste der folgenden Algorithmen angegeben werden: `des, 3des, des_iv64, des_iv32, rc5, rc4, idea, 3idea, cast128, blowfish, null_enc, towfish` und `rijndael`. Bei Algorithmen mit variabler Schlüssellänge kann zusätzlich angegeben werden: `blowfish 448`. Achtung: Möglicherweise unterstützt der Kernel nicht alle Algorithmen!
  - **authentication_algorithm** *algo*; Hier können die Authentifizierungsalgorithmen gewählt werden: `des, 3des, des_iv64, des_iv32, hmac_md5, hmac_sha1, sha2_256, sha2_385, sha2-512` und `non-auth`.
  - **compression_algorithm** *algo*; Hier steht momentan nur `deflate, oui` und `lzs` zur Verfügung. Ab Version 2.5.69 unterstützt der Linux Kernel die Option `deflate`.
- **log** *level*; Es kann zwischen drei verschiedenen Stufen gewählt werden: `notify, debug` (default) und `debug2`.
- **padding { ... }**; Eine Blockcipher kann Daten nur in ganzen Blöcken verschlüsseln. Daher müssen diese Blöcke häufig aufgefüllt werden. Dieser Parameter spezifiziert, wie das Auffüllen erfolgen soll:
  - **randomize (on|off)** Es wird ein zufälliger Wert für das Padding verwendet (Default: `on`).
  - **randomize_length (on|off)** Die Länge des Pads wird zufällig ermittelt (Default: `off`).
  - **maximum_length** *zahl* Maximale Länge des Pads (Default: `20`).
  - **exclusive_tail (on|off)** Fügt die Länge des Pads am Ende des Pads ein (Default: `on`).

Nach seinem Start kann `racoon` mit dem Werkzeug `racoonctl` administriert werden. Dies ist jedoch nur möglich, wenn `racoon` mit dem Parameter `ENABLE_ADMINPORT=1` übersetzt wurde. So ist es nicht notwendig nach Änderungen der Konfiguration `racoon` neu zu starten. Das Werkzeug erlaubt auch die Anzeige, das Löschen und das Aufbauen von SAs. Da hier aber keine Authentifzierung erforderlich ist, ist dieser Zugriff meist deaktiviert. Der Befehl `racoonctl` bietet folgende Optionen bei seinem Aufruf:

- **reload-config** `racoon` liest seine Konfiguration neu ein.
- **[-l [-l]] show-sa [esp|ah|isakmp]** Dies zeigt alle oder nur die SAs des angegebenen Protokolls an.

- **flush-sa [esp|ah|isakmp]** Dies löscht alle oder nur die SAs des angebenen Protokolls.
- **delete-sa** <sa-opts> Löscht eine vorhandene SA.
- **establish-sa** <sa-opts> Baut eine SA auf.

Die Angabe <sa-opts> ist für ISAKMP SAs zu ersetzen durch `isakmp (inet|inet6)` *src dst*. Bei IPsec SAs ist hier anzugeben `(esp|ah) (inet|inet6)` *src/netmask/port dst/netmask/port* `(icmp|tcp|udp|any)`.

Dieses Kapitel hat sämtliche Optionen, die beim Start von `racoon` verwendet und die in der Konfigurationsdatei `/etc/racoon.conf` genutzt werden können vorgestellt und erläutert. Die folgenden beiden Kapitel stellen nun den Einsatz von `racoon` vor und erläutern die Konfiguration am Beispiel. Viele der Optionen müssen nicht oder nur selten angegeben werden, wenn IKE Interoperabilitätsprobleme auftreten.

racoon und Preshared Keys

In diesem Kapitel wird die Konfiguration von `racoon` besprochen. Ziel der vorgestellten Konfiguration ist der Aufbau eines verschlüsselten Tunnels, bei der die Authentifizierung über ein Preshared Key (PSK) erfolgt.

Hierbei soll wieder, wie in allen bisherigen Beispielen, ein Tunnel vom Netzwerk *NewYorkNet* (10.0.1.0/24) über den Rechner *NewYork* (3.0.0.1) und den Rechner *Berlin* (5.0.0.1) zum Netzwerk *BerlinNet* (10.0.2.0/24) aufgebaut werden. Diese Testumgebung ist im Kapitel 11, »Testumgebungen« genau erläutert.

Die Konfiguration soll mit der Erzeugung der Datei `/etc/racoon.conf` begonnen werden. Die fertige Datei ist in Listing 6.12 dargestellt. Die Verwendung und die Bedeutung der gesetzten Parameter werden noch erklärt.

```
path pre_shared_key "/etc/psk.txt";

remote 5.0.0.1 {
        exchange_mode main;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}
```

```
sainfo address 10.0.1.0/24 any address 10.0.2.0/24 any {
        pfs_group 768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Listing 6.12 Konfiguration von `racoon` für PSKs (NewYork)

Zunächst wird mit dem Parameter `path` die Datei angegeben, in der sich der gemeinsame Schlüssel für die Authentifizierung der Gegenstelle befindet. In dieser Datei kann pro Zeile ein PSK abgespeichert werden. Hierbei muss auf einer Zeile die Identität der Gegenseite und der Schlüssel abgespeichert werden. Beginnt der Schlüssel mit `0x` so wird der Schlüssel als hexadezimale Zahl angesehen. Ansonsten wird die Zeichenkette als PSK eingelesen. Der Schlüssel darf Leerzeichen enthalten. Alle Zeichen ab dem zweiten Wort jeder Zeile werden als Schlüssel angesehen. Listing 6.13 zeigt eine Beispieldatei. Die Rechte dieser Datei sind auf 600 gesetzt und der Eigentümer der Datei ist `root`.

```
# IPv4 Adressen
10.0.5.1                schlechte psk
5.0.0.1                 0xe10bd52b0529b54aac97db63462850f3
# USER_FQDN
ralf@spenneberg.net     kfaizeafasdf
# FQDN
www.spenneberg.net      Dies ist ein PSK
```

Listing 6.13 Format der Datei psk.txt

Nach der Definition der PSKs erfolgt die Spezifikation der Parameter für den Aufbau der Phase 1 mit dem Rechner 5.0.0.1. Hier wird der IKE Mode mit dem Parameter `exchange_mode` definiert. Anschließend wird das Proposal für Phase 1 definiert. `racoon` wird diese Angabe zusammen mit den Einträgen der SPD (siehe unten) verwenden um die Proposals zu erzeugen, die an den Partner gesendet werden. Hier wird der Verschlüsselungsalgorithmus `3des` und der Hash-Algorithmus `md5` angegeben. Die Authentifizierung erfolgt mit einem PSK und als Diffie Hellmann Gruppe soll die Gruppe 2 mit 1024 Bit verwendet werden. Diese Angaben sind erforderlich und definieren Phase 1 (siehe Kapitel 10, »Fehlersuche«).

Nun müssen noch einige Parameter für die IPsec SAs der zweiten Phase definiert werden. Dies erfolgt mit der Direktive `sainfo`. Die Angabe kann allgemein mit `anonymous` oder spezifisch wie oben zu sehen erfolgen. Anschließend ist die Diffie Hellmann Gruppe für die Perfect Forward Secrecy (PFS),

der Verschlüsselungsalgorithmus, und jeweils der Algorithmus für die Authentifizierung und Kompression der Pakete zu spezifizieren. Alle Parameter sind verpflichtend.

Die entsprechende Datei für Berlin ist spiegelverkehrt zu konfigurieren. Um Missverständnissen vorzubeugen, soll sie hier aufgeführt werden.

```
path pre_shared_key "/etc/psk.txt";

remote 3.0.0.1 {
        exchange_mode main;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo address 10.0.2.0/24 any address 10.0.1.0/24 any {
        pfs_group 768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Listing 6.14 Konfiguration von `racoon` für PSKs (Berlin)

Bei der Erzeugung der Dateien ist es wichtig, dass die verwendeten Algorithmen und DH Gruppen übereinstimmen.

Sind soweit die Konfigurationsdateien erzeugt worden, so ist `racoon` einsatzbereit. Jedoch ist `racoon` nicht selbstständig in der Lage die IKE Verhandlungen durchzuführen und die IPsec Verbindungen aufzubauen. Hierzu muss er vom Kernel aufgefordert werden. Damit der Kernel `racoon` auffordern kann, benötigt dieser entsprechende Informationen, wann dies geschehen muss. Dies wird durch Security Policies in der SPD gesteuert. Sie müssen mit dem Kommando `setkey` erzeugt werden. Sobald der Kernel feststellt, dass ein Paket entsprechend einer Security Policy mit dem IPsec Protokoll behandelt werden muss und er nicht über eine entsprechende SA in der IPsec Datenbank verfügt, wird er `racoon` auffordern diese zu erzeugen. `racoon` wird mit der Gegenseite in IKE Verhandlungen treten, und nach Abschluss die SAs in der SAD eintragen, so dass die Pakete vom Kernel verschlüsselt und authentifiziert werden können.

In diesem Fall sind die folgenden Security Policies erforderlich:

```
#!/usr/sbin/setkey -f
# Dies ist die Datei /etc/setkey.conf
#
# Loesche die SAD und SPD
flush;
spdflush;

# Richtlinien zur Verwendung der SAs (Tunnel NewYorkNet-BerlinNet)
spdadd 10.0.1.0/24 10.0.2.0/24 any -P out ipsec
           esp/tunnel/3.0.0.1-5.0.0.1/require;

spdadd 10.0.2.0/24 10.0.1.0/24 any -P in ipsec
           esp/tunnel/5.0.0.1-3.0.0.1/require;
```

Listing 6.15 Security Policies für `racoon` (NewYork)

Wenn Ihnen die Syntax dieser Richtlinien nicht geläufig ist, so lesen Sie bitte das letzte Kapitel zum Aufbau manuell verschlüsselter Verbindungen mit `setkey`. Hier wird jeweils ein Tunnel von 10.0.1.0/24 nach 10.0.2.0/24 über 3.0.0.1 und 5.0.0.1 und umgekehrt definiert und die Verschlüsselung mit ESP verlangt.

Wurde auch diese Datei spiegelverkehrt auf dem Rechner Berlin erzeugt – hierzu ist der Austausch der Richtungen `in` und `out` erforderlich – so kann der Tunnel getestet werden. Zu Testzwecken und weil wahrscheinlich in der einen oder anderen Datei sich noch ein Tippfehler versteckt hat, sollte zunächst auf beiden Systemen der Befehl `setkey -f /etc/setkey.conf` ausgeführt werden. Wurde dieser Befehl fehlerfrei ausgeführt, sollte auf beiden Systemen der Befehl `racoon -F` aufgerufen werden. Findet `racoon` seine Konfigurationsdatei `/etc/racoon.conf` nicht, kann sie mit der Option `-f /etc/racoon.conf` angegeben werden. Listing 6.16 zeigt den Start von `racoon` und den fehlerfreien Aufbau des Tunnels.

```
# racoon   -F
Foreground mode.
2003-02-21 18:11:17: INFO: main.c:170:main(): @(#)racoon 20001216 20001216
sakane@kame.net
2003-02-21 18:11:17: INFO: main.c:171:main(): @(#)This product linked
OpenSSL 0.9.6b [engine] 9 Jul 2001 (http://www.openssl.org/)
2003-02-21 18:11:17: INFO: isakmp.c:1365:isakmp_open(): 127.0.0.1[500]
used as isakmp port (fd=7)
2003-02-21 18:11:17: INFO: isakmp.c:1365:isakmp_open(): 10.0.1.1[500] used
as isakmp port (fd=8)
2003-02-21 18:11:17: INFO: isakmp.c:1365:isakmp_open(): 3.0.0.1[500] used
as isakmp port (fd=9)
```

```
2003-02-21 18:11:37: INFO: isakmp.c:1689:isakmp_post_acquire(): IPsec-SA
request for 5.0.0.1 queued due to no phase1 found.
2003-02-21 18:11:37: INFO: isakmp.c:794:isakmp_ph1begin_i(): initiate new
phase 1 negotiation: 3.0.0.1[500]<=>5.0.0.1[500]
2003-02-21 18:11:37: INFO: isakmp.c:799:isakmp_ph1begin_i(): begin
Identity Protection mode.
2003-02-21 18:11:37: INFO: vendorid.c:128:check_vendorid(): received
Vendor ID: KAME/racoon
2003-02-21 18:11:37: INFO: vendorid.c:128:check_vendorid(): received
Vendor ID: KAME/racoon
2003-02-21 18:11:38: INFO: isakmp.c:2417:log_ph1established(): ISAKMP-SA
established 3.0.0.1[500]-5.0.0.1[500]
spi:6a01ea039be7bac2:bd288ff60eed54d0
2003-02-21 18:11:39: INFO: isakmp.c:938:isakmp_ph2begin_i(): initiate new
phase 2 negotiation: 3.0.0.1[0]<=>5.0.0.1[0]
2003-02-21 18:11:39: INFO: pfkey.c:1106:pk_recvupdate(): IPsec-SA
established: ESP/Tunnel 5.0.0.1->3.0.0.1 spi=68291959(0x4120d77)
2003-02-21 18:11:39: INFO: pfkey.c:1318:pk_recvadd(): IPsec-SA
established: ESP/Tunnel 3.0.0.1->5.0.0.1 spi=223693870(0xd554c2e)
```

Listing 6.16 Start von `racoon` und Aufbau des Tunnels

Der Tunnelaufbau wurde von einem Rechner im NewYorkNet gestartet durch eine SSH Verbindung zu einem Rechner im BerlinNet.

Fazit

Der Aufbau einer Verbindung mit PSKs ist beim Einsatz von `racoon` nicht weniger aufwändig als bei FreeS/WAN. `racoon` unterstützt jedoch den Aggressive Modus und kann daher insbesondere beim Einsatz von PSKs eine sinnvolle Alternative zu FreeS/WAN sein.

racoon und x509 Zertifikate

Der IKE Daemon `racoon` ist ohne weiteren Patch in der Lage mit X.509 Zertifikaten eine Authentifizierung durchzuführen. Zusätzlich bietet er auch die Möglichkeit die Authentifizierung mit GSS-API und Kerberos zu bewältigen. Hier soll jedoch nur die Möglichkeit der X.509 Zertifikate erläutert werden. Die Vorteile bei der Verwendung von X.509 Zertifikaten wurden bereits mehrfach im Zusammenhang mit FreeS/WAN ausgeführt. Deshalb sollen hier nur die wesentlichen Aspekte wiederholt werden.

Bei jeder normalen Authentifizierung ist es erforderlich, dass die Informationen zwischen allen Kommunikationspartnern ausgetauscht werden müssen. So müssen die PSKs oder öffentliche Schlüssel jedem Kommunikationspartner bekannt sein, so dass er jeden anderen authentifizieren kann. Hierdurch entsteht ein hoher Administrationsaufwand und bei vielen Kommunikationspartnern besteht schnell die Gefahr, dass der Überblick verloren geht. Hier bieten

X.509 Zertifikate Abhilfe, da alle Kommunikationspartner lediglich ihren eigenen privaten Schlüssel, ihr eigenes Zertifikat und das Zertifikat einer zentralen Zertifikatsautorität (CA) besitzen müssen. Sie vertrauen dann automatisch sämtlichen Kommunikationspartnern, die ebenfalls ein von dieser CA unterzeichnetes Zertifikat und den entsprechenden privaten Schlüssel besitzen.

Hier soll nun der im letzten Kapitel aufgebaute Tunnel so konfiguriert werden, dass die Authentifizierung mit einem X.509 Zertifikat erfolgen kann. Die Erzeugung der Zertifikate wurde im entsprechenden FreeS/WAN Kapitel beschrieben (siehe Exkurs »Exkurs: Erzeugung von X.509 Zertifikaten mit OpenSSL«). Es werden hier dieselben Zertifikate eingesetzt. Natürlich müssen diese Zertifikate an einer anderen Stelle abgespeichert werden.

Die Konfiguration der SPD mit dem Befehl `setkey` unterscheidet sich nicht vom letzten Kapitel. Sie ist als Referenz im Listing 6.17 erneut für den Rechner NewYork abgedruckt.

```
#!/usr/sbin/setkey -f
# Dies ist die Datei /etc/setkey.conf
#
# Loesche die SAD und SPD
flush;
spdflush;

# Richtlinien zur Verwendung der SAs (Tunnel NewYorkNet-BerlinNet)
spdadd 10.0.1.0/24 10.0.2.0/24 any -P out ipsec
           esp/tunnel/3.0.0.1-5.0.0.1/require;

spdadd 10.0.2.0/24 10.0.1.0/24 any -P in ipsec
           esp/tunnel/5.0.0.1-3.0.0.1/require;
```

Listing 6.17 Die Befehle zur Erzeugung der SPD Einträge

Die Konfigurationsdatei `/etc/racoon.conf` unterscheidet sich stärker von der Version des letzten Kapitels. Die Datei für die Verwendung von x509 Zertifikaten ist in Listing 6.18 abgedruckt.

```
path certificate "/etc/certs";

remote 5.0.0.1 {
        exchange_mode main;
        certificate_type x509 "newyork_cert.pem" "newyork_req.pem";
        my_identifier asn1dn;
```

```
   peers_identifier asn1dn;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method rsasig;
                dh_group modp1024;
        }
}


sainfo address 10.0.1.0/24 any address 10.0.2.0/24 any {
        pfs_group 768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Listing 6.18 Konfiguration von `racoon` für x509 Zertifikate (NewYork)

Der Einfachheit halber sollen die identischen Zertifikate verwendet werden, die im Exkurs OpenSSL (»Exkurs: Erzeugung von X.509 Zertifikaten mit OpenSSL«) erzeugt wurden.

Hierzu ist zunächst das Verzeichnis `/etc/certs` zu erstellen. Anschließend müssen die Dateien `newyork_cert.pem`, `newyork_req.pem`, `crl.pem` und `cacerts.pem` hier hinein kopiert werden. Die Überprüfung der über das Netz übermittelten Zertifikate erfolgt mit OpenSSL. Damit OpenSSL die CRL und das Zertifikat der CA findet, müssen sie umbenannt oder symbolische Links erzeugt werden:

```
# cd /etc/certs
# ln -s cacert.pem `openssl x509 -noout -hash -in cacert.pem `.0
# ln -s crl.pem  `openssl x509 -noout -hash -in cacert.pem `.r0
# ls -F
0c63af7c.0@  0c63af7c.r0@ cacert.pem newyork_req.pem newyork_cert.pem
```

Listing 6.19 OpenSSL konforme Benennung des CA Certs

Da `racoon` nicht in der Lage ist einen verschlüsselten privaten Schlüssel zu lesen, müssen diese noch entschlüsselt werden:

```
# openssl rsa -in newyork_req.pem -out newyork_req.pem
read RSA key
Enter PEM pass phrase: certkenn
writing RSA key
```

Listing 6.20 Entfernung der Verschlüsselung des privaten Schlüssels

Wurden die Konfigurationsdateien entsprechend für NewYork und Berlin erstellt und die Schlüssel und Zertifikate an die entsprechenden Stellen kopiert, so können die Einträge in der SPD erzeugt und `racoon` gestartet werden. Nach dem Start kann dann von NewYorkNet oder BerlinNet der Tunnel gestartet werden.

Mit dieser Konfiguration sollte auch die Möglichkeit bestehen ein heterogenes VPN mit FreeS/WAN aufzubauen. Einer der beiden VPN Gateways Berlin oder NewYork kann durch die entsprechende FreeS/WAN Konfiguration ersetzt werden. Der Aufbau des Tunnels schlägt jedoch fehl (siehe Listing 6.21). FreeS/WAN unterstützt nur die Gruppen 2 (modp1024) und 5 (modp1536) für die Perfect Forward Secrecy.

```
Feb 21 20:45:26 berlin pluto[625]: "x509-newyorknet-berlinnet" #1:
responding to Main Mode
Feb 21 20:45:26 berlin pluto[625]: "x509-newyorknet-berlinnet" #1:
ignoring Vendor ID payload
Feb 21 20:45:26 berlin pluto[625]: "x509-newyorknet-berlinnet" #1:
Peer ID is ID_DER_ASN1_DN: 'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
OU=Wireless-VPN, CN=NewYork, E=ralf@spenneberg.net'
Feb 21 20:45:26 berlin pluto[625]: "x509-newyorknet-berlinnet" #1:
sent MR3, ISAKMP SA established
Feb 21 20:45:26 berlin pluto[625]: "x509-newyorknet-berlinnet" #1:
ignoring informational payload, type IPSEC_INITIAL_CONTACT
Feb 21 20:45:26 berlin pluto[625]: "x509-newyorknet-berlinnet" #1:
received and ignored informational message
Feb 21 20:45:27 berlin pluto[625]: "x509-newyorknet-berlinnet" #2:
only OAKLEY_GROUP_MODP1024 and OAKLEY_GROUP_MODP1536 supported for PFS
```

Listing 6.21 Fehlerhafter Aufbau des Tunnels (FreeSWAN Log)

Wird die Konfiguration auf der Seite von `racoon` entsprechend angepasst (`pfs_group modp1024;`), so kann der Tunnel erfolgreich aufgebaut werden (Listing 6.22).

```
Feb 21 21:16:51 berlin pluto[658]: "x509-newyorknet-berlinnet" #1:
responding to Main Mode
Feb 21 21:16:51 berlin pluto[658]: "x509-newyorknet-berlinnet" #1:
ignoring Vendor ID payload
Feb 21 21:16:52 berlin pluto[658]: "x509-newyorknet-berlinnet" #1:
Peer ID is ID_DER_ASN1_DN: 'C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
OU=Wireless-VPN, CN=NewYork, E=ralf@spenneberg.net'
Feb 21 21:16:52 berlin pluto[658]: "x509-newyorknet-berlinnet" #1:
sent MR3, ISAKMP SA established
Feb 21 21:16:52 berlin pluto[658]: "x509-newyorknet-berlinnet" #1:
ignoring informational payload, type IPSEC_INITIAL_CONTACT
```

```
Feb 21 21:16:52 berlin pluto[658]: "x509-newyorknet-berlinnet" #1:
received and ignored informational message
Feb 21 21:16:53 berlin pluto[658]: "x509-newyorknet-berlinnet" #2:
responding to Quick Mode
Feb 21 21:16:54 berlin pluto[658]: "x509-newyorknet-berlinnet" #2:
IPsec SA established
```

Listing 6.22 Erfolgreicher Aufbau des Tunnels (FreeS/WAN Log)

racoon und Roadwarrior

In diesem Kapitel soll nun die Konfiguration von `racoon` in einem Roadwarrior Szenario vorgestellt werden. Als Roadwarrior bezeichnet man Rechner, die nicht über eine feste IP Adresse und eine dauerhafte Internetverbindung verfügen. Das bedeutet, dass die Verbindungen immer nur vom Roadwarrior aufgebaut werden können und das VPN Gateway Verbindungen von jeder IP Adresse erlauben muss.

Im Falle von `racoon` auf dem VPN Server bedeutet das, dass der Kernel nicht mehr von `racoon` die Erzeugung der IPsec SAs verlangen kann. Der Aufbau der ISAKMP SA mit dem IKE Protokoll wird direkt von dem Roadwarrior gestartet. Hierzu verbindet sich der Roadwarrior mit `racoon`. Wenn `racoon` die entsprechenden IPsec SAs erzeugt hat, ist der Kernel jedoch immer noch nicht in der Lage sie auch zu nutzen. Der Kernel benötigt zusätzlich noch Security Policies, die definieren, wann die IPsec SAs zu verwenden sind. Sie müssen auch von `racoon` erzeugt werden. Die entsprechende `racoon` Konfigurationsdatei für das VPN Gateway ist in Listing 6.23 dargestellt.

```
path certificate "/etc/certs";

remote anonymous {
        exchange_mode main;
        generate_policy on;
        passive on;
        certificate_type x509 "newyork_cert.pem" "newyork_req.pem";
        my_identifier asn1dn;
        peers_identifier asn1dn;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method rsasig;
                dh_group modp1024;
        }
}
```

```
sainfo anonymous {
        pfs_group modp1024;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Listing 6.23 `racoon` als VPN Gateway (NewYork)

Diese Datei wurde an vier Stellen modifiziert. Zunächst wurde der Kommunikationspartner als anonym spezifiziert (`remote anonymous`) und nicht mit einer IP Adresse identifiziert. Die hier angegebenen Parameter für Phase 1 gelten nun für alle Kommunikationspartner. Ebenso wurde für die Parameter der Phase 2 die genaue Spezifikation der IP Adressen durch die Angabe `sainfo anonymous` ersetzt.

Um zu verhindern, dass `racoon` selbst versucht Verbindungen aufzubauen, wurde der Parameter `passive` gesetzt. Nun muss `racoon` noch die entsprechenden Security Policies nach Aufbau der Verbindung und Aushandlung der ISAKMP SA und der IPsec SAs in der SPD eintragen. Hierzu dient der Parameter `generate_policy`.

Damit nicht vorhandene Security Policies Probleme bereiten, sollten diese mit einem Aufruf von `setkey -f /etc/setkey.conf` entfernt werden bevor, `racoon` gestartet wird. Die hierfür erforderlich Datei `/etc/setkey.conf` ist in Listing 6.24 abgebildet.

```
#!/usr/sbin/setkey -f

# Loesche die SAD und SPD
flush;
spdflush;
```

Listing 6.24 Löschen der SAD und SPD mit `setkey`

Nun kann von jedem externen Rechner aus eine beliebige Verbindung zu diesem Gateway aufgebaut werden. Hierbei definiert der Initiator die zu verwendenden IP Adressen, Algorithmen und DH Gruppen.

6.4.4 Fazit

Die Verwendung des Linux Kernels 2.6 zum Aufbau von VPN Lösungen ist relativ unproblematisch. Kinderkrankheiten, die der Autor noch beim Test der Werkzeuge auf der Basis des Linux Kernels 2.5.47-62 hatte, werden bei

der Freigabe des Linux Kernel 2.6 wahrscheinlich behoben sein. Der IKE Daemon `racoon` ist in der Lage, mit den verschiedenen anderen Implementierungen zu interoperieren und bietet umfassende Optionen für die Anpassung an andere VPN Systeme.

6.5 Verwendung von isakmpd

`isakmpd` ist seit einigen Jahren der IKE-Daemon des OpenBSD-Projektes und ersetzt damit in vielen Fällen den `photurisd`, welcher noch ein Photuris-Key-Management-Protokoll (RFC 2522, 2523) verwendet. Er wird von vielen Personen wegen seines sauberen Codes und der übersichtlichen Konfiguration geschätzt. Der Isakmpd unterstützt hierbei die folgenden Funktionen:

- Unterstützung von IPv4 und IPv6.
- `isakmpd` kann die Pakete bei Bedarf mit DES, 3DES, CAST, Blowfish oder AES (AES in Phase 1 erst in Versionen ab dem 28.08.2003) verschlüsseln.
- Für die Authentifizierung der Pakete bietet `isakmpd` MD5, SHA, RIPEMD, SHA2-256, SHA2-384 und SHA2-512 (in Phase 1 nur MD5 und SHA).
- Diffie Hellmann Gruppen 1, 2 oder 5 und die ECC-Gruppen 3 und 4.
- Authentifizierung des Peers mit Preshared Keys (PSK), DSS oder RSA-Signaturen (X.509 Zertifikaten).
- Unterstützung von X.509 Zertifikaten
- Unterstützung des Main Modus und des Aggressive Modus.
- Unterstützung des IKE Mode Config. Dies ist eine ältere Alternative zum DHCP-over-IPsec Modus und erlaubt unter anderem die Verteilung von IP Adressen, Netzmasken, DNS und WINS Server an einen Client. Hierbei kann jedoch leider nicht auf einen Adressen Pool zugegriffen werden, wie bei DHCP.

`isakmpd` wurde von Thomas Walpuski auf den Linux Kernel 2.5/2.6 portiert und kann jetzt hier eingesetzt werden.

6.5.1 Installation

Thomas Walpuski beschreibt auf *http://bender.thinknerd.de/~thomas/IPsec/isakmpd-linux.html* die Installation unter Linux aus dem Quelltext. Anwender einer Debian- oder RPM-basierten Distribution können jedoch auch ein Paket installieren. Das RPM-Paket ist unter *http://www.spenneberg.org/VPN/Kernel-2_6_IPsec* zu finden. Bei der Installation eines Paketes ist darauf zu achten,

dass die Linux Kernel 2.5 und 2.6-testX noch mehrfache Änderungen durchlaufen haben. Dies führt dazu, dass ein Paket, das für einen 2.6.0-test1 Kernel kompiliert wurde, auf einem 2.6.0-test4 Kernel nicht mehr funktioniert, sondern erneut übersetzt werden muss.

Für die Übersetzung des `isakmpd` ist eine vorherige Installtion der KeyNote-Bibliothek (*http://www1.cs.columbia.edu/~angelos/keynote.html*) erforderlich. Die Version 2.3 kann mit den folgenden Befehlen übersetzt werden:

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

Unter *http://www.spenneberg.org/VPN/Kernel-2_6_IPsec* ist ein RPM-Paket verfügbar.

Um `isakmpd` aus den Quellen zu installieren, muss zunächst das Quelltext-Archiv geladen oder mit CVS ausgecheckt werden. Thomas Walpuski hält auf der Seite auch Snapshots des CVS bereit. Eine Release-Version mit Versionsnummer existiert nicht. Um `isakmpd` aus dem CVS auszuchecken, sind die folgenden Befehle nötig:

```
# export CVS_RSH=/opt/ssh/bin/ssh
# export CVSROOT=:pserver:anoncvs@anoncvs.de.openbsd.org:/cvs
# cvs login
Logging in to :pserver:anoncvs@anoncvs.de.openbsd.org:2401/cvs
CVS password: anoncvs
# cvs co -d isakmpd src/sbin/isakmpd
```

Anschließend kann der Administrator den Quelltext konfigurieren. Dieser Vorgang erfolgt manuell. Hierzu muss die Datei `GNUmakefile` editiert werden. Dort ist die Zeile

```
OS=linux
```

als einzige OS-Zeile zu aktivieren. Anschließend sollte noch eine Datei kopiert werden und der `isakmpd` kann übersetzt werden. Teilweise wird auch die Datei `bitstring.h` aus den BSD-Quellen benötigt. Sie sollte dann auch in das entsprechende Verzeichnis `sysdep/linux/sys` kopiert werden. Die CVS-Version ab September 2003 enthält bereits die Datei `queue.h` und `bitstring.h` an den richtigen Stellen.

```
# cp sysdep/freeswan/sys/queue.h sysdep/linux/sys
# CFLAGS="-I/usr/kerberos/include -I/usr/src/linux-2.6.0" make
# make install
```

Die zusätzlichen Compiler-Flags sind erforderlich, um die Position des Linux-Kernel-Quelltextbaumes und der Kerberos-Include-Dateien anzuzeigen.

Ein abschließendes `make install` installiert `isakmpd`.

6.5.2 Anwendung mit PSKs

Um den Rahmen des Buches nicht zu sprengen, soll hier keine genaue Betrachtung aller möglichen Optionen von `isakmpd` erfolgen. Vielmehr soll an Hand von zwei Anwendungsfällen die Konfiguration und die Funktionsweise von `isakmpd` beschrieben werden.

Im ersten Fall soll die Konfiguration von `isakmpd` bei einer Authentifizierung mit Preshared Keys beschrieben und erläutert werden. Dabei soll das in Abbildung 6.2 beschriebene Szenario zunächst nachgebildet werden. Die hier verwendeten IP Adressen und Namen sind bereits mehrfach, auch in den anderen Beispielen, verwendet worden.

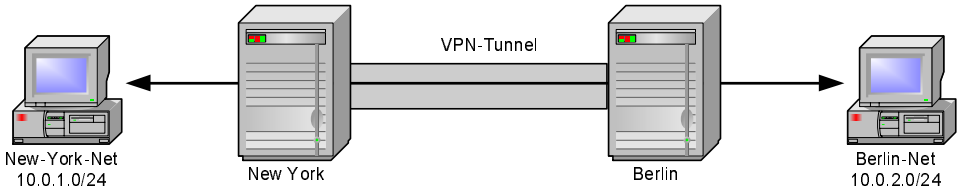


Abbildung 6.2 Ein VPN-Tunnel mit `isakmpd` verbindet die Netzwerke NewYork-Net und BerlinNet

Die Konfigurationsdatei isakmpd.conf

`isakmpd` verwendet eine Konfigurationsdatei und eine Richtliniendatei. Zuerst soll die Konfigurationsdatei `/etc/isakmpd/isakmpd.conf` vorgestellt werden.

```
[General]
Listen-on=              3.0.0.1

[Phase 1]
5.0.0.1=                ISAKMP-peer-east

[Phase 2]
Connections=            IPsec-west-east
```

```
[ISAKMP-peer-east]
Phase=                  1
Transport=              udp
Address=                5.0.0.1
Local-address=          3.0.0.1
Configuration=          Default-main-mode
Authentication=         soM59Mer

[IPsec-west-east]
Phase=                  2
ISAKMP-peer=            ISAKMP-peer-east
Configuration=          Default-quick-mode
Local-ID=               Net-west
Remote-ID=              Net-east

[Net-west]
ID-type=                IPV4_ADDR_SUBNET
Network=                10.0.1.0
Netmask=                255.255.255.0

[Net-east]

ID-type=                IPV4_ADDR_SUBNET
Network=                10.0.2.0
Netmask=                255.255.255.0

[Default-main-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=             3DES-SHA,BLF-SHA,3DES-MD5,BLF-MD5

[Default-quick-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          QUICK_MODE
Suites=                 QM-ESP-AES-SHA-PFS-SUITE
```

Listing 6.25 Die Datei `/etc/isakmpd/isakmpd.conf` für die Authentifizierung mit PSKs

Diese Datei ist nach dem Format einer Windows-ini-Datei aufgebaut. Sie besteht aus einzelnen Sektionen, deren Namen in eckigen Klammern eingetragen werden. In diesen Abschnitten befinden sich dann Tags, denen Werte zugewiesen werden.

Den Anfang in dieser Datei macht die Sektion `[General]`. Hier können in erster Linie Default Werte definiert werden. Im einzelnen sind folgende Werte modifizierbar:

- **Default-phase-1-ID** Dies ist die Default-Identifikation für Phase 1 des IKE-Protokolls
- **Default-phase-1-lifetime** Hiermit kann die Lebensdauer der ISAKMP SA definiert werden. Der Default Wert ist `3600,60:86400`. Das bedeutet, dass der Default Wert 3600 Sekunden, also eine Stunde beträgt. Die minimale akzeptierte Lebensdauer beträgt 60 Sekunden und die maximale Lebensdauer 86.400 Sekunden, also einen Tag.
- **Default-phase-2-lifetime** Hier wird die Lebensdauer der IPsec SA definiert. Der Default Wert beträgt `1200,60:86400`. Das bedeutet, dass die IPsec SA per Default eine Lebensdauer von 1200 Sekunden, 20 Minuten, erhält. Maximal beträgt die Lebensdauer jedoch einen Tag und minimal eine Minute.
- **Default-phase-2-suites** Wenn `isakmpd` dynamisch die IPsec SAs erzeugt, kann hier die Default-Suite angegeben werden. Wird diese Option nicht gesetzt, so ist QM-ESP-3DES-SHA-PFS-SUITE der Default Wert. Dies bedeutet, Phase 2 verwendet den Quickmode mit dem ESP-Protokoll um die Integrität, Authentizität und Vertraulichkeit zu garantieren. Dazu werden die 3DES-Verschlüsselung und ein SHA-HMAC eingesetzt. Die Perfect-Forward-Secrecy ist aktiviert.
- **Check-interval** Dieser Wert definiert, wie häufig `isakmpd` Verbindungen auf ihren Zustand überprüft und sie bei Fehlern neustartet.
- **Exchange-max-time** Maximale Dauer des Austausches, bevor `isakmpd` aufgibt (120 Sekunden default).
- **Listen-on** `isakmpd` bindet sich auf angegebene Adressen (default: alle).
- **Policy-file** Name der Richtliniendatei (default: `/etc/isakmpd/isakmpd.policy`).
- **Pubkey-directory** Die Zertifikate in diesem Verzeichnis stuft `isakmpd` explizit als vertrauenswürdig ein. Dabei müssen die Zertifikate einen subjectAltName enthalten und diesen auch als Dateinamen verwenden. Zertifikate mit einem IPv4-subjectAltName werden anschließend in `/etc/isakmpd/pubkeys/ipv4/A.B.C.D` gespeichert. IPv6-subjectAltName-Zertifikate werden in `/etc/isakmpd/pubkeys/ipv4/abcd:abcd:abcd::ab:cd` gespeichert, FQDN und UFQDN entsprechend in den Unterverzeichnissen `fqdn/` bzw. `ufqdn/`.
- **Renegotiate-on-HUP** Wenn diese Option gesetzt wird, handelt `isakmpd` bei jedem HUP-Signal die IPsec SAs neu aus.
- **Retransmits** Diese Einstellung definiert, wie häufig `isakmpd` eine Nachricht wiederholt, bevor er aufgibt.

- **Shared-SADB** Diese Einstellung erlaubt es, dass mehrere `isakmpd` auf eine SADB zugreifen und diese administrieren dürfen.

Üblicherweise ist eine Modifikation der Parameter im Abschnitt `[General]` nicht erforderlich. Die Default Werte sind in den meisten Umgebungen sinnvoll. Lediglich eine Einschränkung der IP Adressen mit dem Parameter `Listen-on` ist beim Einsatz des `isakmpd` auf einem VPN-Gateway empfohlen.

Anschließend wird die Sektion `[Phase 1]` definiert. Hier werden die ISAKMP-Peers definiert und einzelnen IP Adressen der Name eines ISAKMP-Peers zugeordnet. Ist die IP Adresse des Peers nicht bekannt, so kann auch ein Default-Eintrag definiert werden:

```
Default=ISAKMP-peer-unkown
```

Anschließend an die Definition der Phase 1 werden die Tunnel der `[Phase 2]` spezifiziert. Hier können Verbindungen (Connections) definiert werden, die `isakmpd` automatisch und aktiv versucht zu starten. Wenn ein VPN-Gateway lediglich Anforderungen für den Aufbau eines Tunnels von außen annehmen soll, aber nicht selbst aktiv den Tunnel aufbauen darf, so sieht die Konfigurationsdatei hierfür den Parameter `Passive-connections` vor. Diese Funktion wird heute kaum noch verwendet. Weiter unten wird in einem eigenen Abschnitt ein Beispiel für ein Roadwarrior-Szenario gegeben.

Nun werden die in Phase 1 und 2 referenzierten symbolischen Namen genauer definiert. Zunächst die Peers der Phase 1. Dabei können die folgenden Optionen angegeben werden:

- **Phase** Dies ist immer der Wert 1. Er definiert, dass diese Beschreibung einen ISAKMP Peer enthält.
- **Transport** Der Name des Transportprotokolls: UDP.
- **Port** Der IKE-Port (Default: 500).
- **Local-address** Die eigene IP Adresse, die in der IKE-Verhandlung genutzt werden soll. Diese Angabe ist wichtig bei Systemen, die über mehrere IP Adressen verfügen.
- **Address** Die IP Adresse des Peers, wenn sie bekannt ist.
- **Configuration** Angabe der zu verwendenden ISAKMP-Konfiguration. Sie definiert den ISAKMP Modus und die Verfahren zur Verschlüsselung (Default: `Default-phase-1-configuration`). Diese Default-Einstellung wird weiter unten erklärt.
- **Authentication** Hier wird bei der Verwendung von PSKs, der PSK für die Authentifizierung beim Peer gespeichert.

- **ID** Hier wird ein Verweis auf einen weiteren Abschnitt eingetragen, der die eigene Phase-1-Identifikation (ID) beschreibt. Ist dieser Wert nicht gesetzt, so wird als ID die eigene IP Adresse verwendet. Wurde im `[General]`-Abschnitt eine `Default-phase-1-ID` gesetzt, so wird diese genutzt. Da in der angegebenen Beispieldatei keine `Phase-1-ID` definiert wird, wird die Erläuterung dieser auf das nächste Kapitel verschoben.
- **Remote-ID** Dieser Eintrag beschreibt die ID, die wir vom Peer erwarten. Wenn nichts angegeben wird, verwendet `isakmpd` die IP Adresse des Peers. Da in der angegebenen Beispieldatei keine `Phase-1-ID` definiert wird, wird ihre Erläuterung auf das nächste Kapitel verschoben.
- **Flags** Eine Komma-separierte Liste von weiteren `isakmpd`-Optionen. Bisher existiert nur das Flag `ikecfg`, das durch die Konfigurationsdatei gesetzt werden kann. Dieses Flag aktiviert den IKECFG Modus (siehe unten). `isakmpd` verwendet intern weitere Flags, wie `ready`, `stayalive`, `on-demand`, `replaced` und `fading`. Sie können jedoch nicht von dem Benutzer beeinflusst werden!

Nun folgt üblicherweise eine Auflösung der `Phase-1-ID`-Referenzen. Da in der angegebenen Beispieldatei keine `Phase-1-ID` definiert wird, wird ihre Erläuterung auf das nächste Kapitel verschoben.

An die Beschreibung des Phase-1-Peers schließt sich die Beschreibung der Phase-2-Verbindungen an. Sie wurden im Abschnitt `[Phase-2]` durch symbolische Namen referenziert und müssen nun mit Leben gefüllt werden.

Die Beschreibung von Phase-2-Verbindungen erfolgt mit den folgenden Parametern:

- **Phase** Dieser Parameter unterscheidet die Beschreibungen der Phase 2 von denen der Phase 1. Der Wert ist daher hier konstant 2.
- **ISAKMP-peer** Der Name des ISAKMP-Peers, mit dem diese Verbindung ausgehandelt wird. Dies ist eine Referenz auf den vorhergehenden Abschnitt.
- **Configuration** Die Konfiguration von Phase 2. Wird hier keine explizite Konfiguration angegeben, so sind die weiter unten definierten Default Werte aktiv.
- **Local-ID** Dieser Wert spezifiziert optional eine lokale Client ID, die den Tunnel benutzen soll. Hierbei handelt es sich wieder um eine Referenz auf einen eigenen Abschnitt.
- **Remote-ID** Dieser Wert spezifiziert optional eine remote Client ID, die den Tunnel benutzen soll. Beide Werte, dieser und der vorige, werden

von `isakmpd` auch verwendet, wenn er einen Tunnel einer neuen Anfrage zuordnen muss.

- **`Flags`** Hier können in einer Komma-separierten Liste weitere Flags angegeben werden, die das Verhalten der Verbindung beeinflussen. Bisher existiert hier nur das Flag: `active-only`. Dieses Flag beschreibt Verbindungen, die `isakmpd` automatisch aufbaut, auf denen er aber keine Verbindungsaufbauten von außen entgegen nimmt.

Wenn in der Beschreibung der IPsec-Verbindung eine `Local-ID` oder `Remote-ID` angegeben wurden, müssen diese IPsec IDs beschrieben werden. Hierzu stehen die folgenden Parameter zur Verfügung.

- **`ID-type`** Dies definiert die Art des Clients, der den Tunnel nutzen darf. Gültige Werte sind: `IPV4_ADDR`, `IPV6_ADDR`, `IPV4_ADDR_SUBNET` und `IPV6_ADDR_SUBNET`. Wenn `isakmpd` den Tunnel für die Kommunikation zweier IPv4 Netzwerke aufbauen soll, so muss dieser Parameter entsprechend auf `IPV4_ADDR_SUBNET` gesetzt werden.
- **`Address`** Wenn der ID Type `*_ADDR` ist, wird mit diesem Parameter die IP Adresse angegeben.
- **`Network`** Wenn der ID Type den Wert `*_SUBNET` hat, so definiert dieser Parameter die Netzwerkadresse.
- **`Netmask`** Hiermit kann die Netzmaske des `Network` angegeben werden.
- **`Protocol`** Dieser Protokollselektor erlaubt eine Einschränkung der im Tunnel erlaubten Transport-Protokolle, zum Beispiel `udp`. Wird die Angabe weggelassen, so dürfen alle Protokolle im Tunnel verwendet werden.
- **`Port`** Hiermit kann der erlaubte Port für diese ID bei zuzätzlich angegebenen Protokollen definiert werden, zum Beispiel: `bootp`. So ist eine Einschränkung des im Tunnel transportieren IP-Verkehrs möglich. Diese Angaben entsprechen dem Protokoll- und Portselektor des FreeS/WAN-X.509-Patches.

In der Beschreibung des ISAKMP-Peers wurde auf eine `Configuration=Default-main-mode` verwiesen. Sie muss nun definiert werden. Hierfür stehen drei Parameter zur Verfügung:

- **`DOI`** Dieser Parameter definiert die *Domain of Interpretation*. Hier ist im Moment nur der Default Wert möglich: `IPSEC`.
- **`EXCHANGE_TYPE`** Dieser Parameter bestimmt den Modus in Phase 1. Möglich sind die Werte `ID_PROT` (Main Modus, Identity Protection) und `AGGRESSIVE`.
- **`Transforms`** Hier kann eine Liste von Transforms definiert werden, die `isakmpd` in Phase 1 vorschlagen soll. Hierbei handelt es sich wieder nur

um symbolische Namen, die anschließend definiert werden müssen. Jedoch erzeugt `isakmpd` automatisch jede mögliche Permutation von `{DES,BLF,3DES,CAST}-{MD5,SHA}[-GRP{1,2,5}][-{DSS,RSA_SIG}]`. Auf diese Transforms kann also ohne eine zusätzliche Definition zugegriffen werden.

ACHTUNG

Sie sollten sich hier eine Reihe von sinnvollen Transforms definieren. Die Default Werte sind in vielen Umgebungen nicht ausreichend oder nicht optimal, da `isakmpd` per Default nur die 3DES-Verschlüsselung verwendet. `isakmpd` unterstützt in Phase 1 AES in Versionen ab dem 28.08.2003. Dann werden auch die AES-Transforms automatisch generiert.

Wenn bei der Beschreibung des ISAKMP-Peers keine `Configuration` definiert wurde, so verwendet der `isakmpd`, wie beschrieben, die `[Default-phase-1-configuration]`. Diese Konfiguration hat folgenden Inhalt:

```
[Default-phase-1-configuration]
DOI=            IPSEC
EXCHANGE_TYPE=  ID_PROT
Transforms=     3DES-SHA-RSA_SIG
```

Listing 6.26 Die Default-Phase-1-Konfiguration

Die Angabe der Transform `3DES-SHA-RSA_SIG` bedeutet, da eine Verbindung mit einem Preshared Key hier nicht erfolgreich ist, da `isakmpd` mit RSA-Signaturen authentifizieren möchte. Es ist daher bei der Verwendung von PSKs immer erforderlich, eine eigene ISAKMP-Konfiguration zu definieren!

Wenn der Administrator seine eigene ISAKMP-Transform definieren möchte, so kann er das mit folgenden Parametern tun: `ENCRYPTION_ALGORITHM`, `KEY_LENGTH`, `HASH_ALGORITHM`, `AUTHENTICATION_METHOD`, `GROUP_DESCRIPTION`, `PRF` (Pseudo Random Function) und `Life`. Diese Parameter werden in der Manpage `isakmpd.conf(5)` genauer erläutert.

Außerdem wurde in der Beschreibung der IPsec-Verbindung eine Konfiguration `Default-quick-mode` referenziert. Sie muss auch definiert werden. Hierzu stehen die folgenden drei Parameter zur Verfügung:

- **`DOI`** Dieser Wert ist wie bei der Definition der ISAKMP-Konfiguration immer `IPSEC`.
- **`EXCHANGE_TYPE`** Dieser Wert ist für den Quick-Mode der Phase 2 immer `QUICK_MODE`.

- **Suites** Hier stehen eine Vielzahl von vordefinierten Suites zur Verfügung. isakmpd erzeugt nach seinem Start automatisch alle möglichen Permutationen von QM-{ESP,AH}[-TRP]-{DES,3DES,CAST,BLF,AES}[-{MD5,SHA,RIPEMD,SHA2-{256,384,512}}][-PFS[-GRP{1,2,5}]]-SUITE. Die Option -TRP aktiviert den Transport Modus während die Option -PFS die Perfect-Forward-Secrecy aktiviert. isakmpd generiert keine Suites, die gleichzeitig ESP und AH verwenden. Sie müssen manuell erzeugt werden.

Wenn der Benutzer eigene IPsec-Suites erzeugen möchte, so ist dies möglich mit den Parametern Protocols, PROTOCOL_ID, Transforms, ReplayWindow, TRANSFORM_ID, ENCAPSULATION_MODE, AUTHENTICATION_ALGORITHM, GROUP DESCRIPTION und Life. Die Manpage isakmpd.conf(5) erläutert die Verwendung dieser Parameter.

Die Richtliniendatei isakmpd.policy

Zusätzlich zur Konfigurationsdatei /etc/isakmpd/isakmpd.conf benötigt isakmpd für seinen Betrieb auch noch eine Richtliniendatei /etc/isakmpd/isakmpd.policy, in der keynote(5)-Richtlinien gespeichert werden.

Eine typische Richtliniendatei für die Verwendung mit PSKs ist in Listing 6.27 dargestellt.

```
KeyNote-Version: 2
Comment: Diese Richtlinie akzeptiert jeden Peer mit richtigen Kennwort
Authorizer: "POLICY"
Licensees: "passphrase:soM59Mer"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg == "aes" &&
            esp_auth_alg == "hmac-sha" -> "true";
```

Listing 6.27 Bei der Verwendung von PSKs wird als Licensee der PSK angegeben

isakmpd greift für die Authentifzierung auf das KeyNote-System zurück. Das ist ein mächtiges und sehr flexibles System zur Verwaltung von Vertrauensstellungen. Eine genaue Betrachtung des KeyNote-Systems kann hier aus Platzgründen nicht erfolgen. Interessierte Leser werden auf das RFC 2704 und die Manpages keynote(4) und keynote(5) verwiesen. Hier soll lediglich die Konfiguration in dieser Datei erläutert werden.

Das Format der Richtliniendatei besteht aus einzelnen Feldern. Jedes Feld besteht aus einem Namen, gefolgt von einem Doppelpunkt und dem Inhalt des Feldes. Felder können über mehrere Zeilen aufgeteilt werden, wenn die Folgezeilen mit einem Leerzeichen oder Tabulator beginnen. Insgesamt sind

sieben verschiedene Felder möglich: `KeyNote-Version`, `Comment`, `Local-Constants`, `Authorizer`, `Licensees`, `Conditions` und `Signature` (hier nicht erforderlich). Bis auf die Angabe des `Authorizer` sind alle weiteren Felder optional. Die Groß- und Kleinschreibung der Feldnamen ist einzuhalten.

Jede Richtliniendatei sollte zunächst in der ersten Zeile die verwendete KeyNote-Version definieren. Die aktuell eingesetzte Version ist Version 2(.3). Anschließend kann ein Kommentar definiert werden, der die folgende KeyNote-Assertion beschreibt.

Nun folgt das Feld `Authorizer`. Dieses Feld hat üblicherweise den Wert `POLICY`. Jedoch ist es möglich Subpolicies zu definieren. Die Manpage `isakmpd.policy(5)` hat einige Beispiele.

Auf den `Authorizer` folgt das Feld `Licensees`. Dieses Feld definiert die Authentifizierungsinformationen. Dabei kann es sich um eine oder mehrere Passphrasen, öffentliche Schlüssel, oder X.509 Zertifikatsnamen handeln. Hier wird eine Passphrase eingesetzt. Es besteht auch die Möglichkeit die Passphrase mit MD5 oder SHA-1 zu verschlüsseln und folgendermaßen anzugeben:

```
Licensees: "passphrase-md5-hex:3858f62230ac3fc915f300c664312c663f"
```

Abschließend werden die Bedingungen definiert, die erfüllt werden müssen, damit die Anmeldung erfolgreich ist. Dazu wird das Feld `Conditions` verwendet. Hier können nun eine Vielzahl von Attributen abgefragt werden. Unter anderem sind dies die Attribute `app_domain`, `initiator`, `pfs`, `esp_present`, `esp_enc_alg`, `comp_alg`, `esp_auth_alg` und so weiter Eine vollständige Liste enthält die Manpage `isakmpd.policy(5)`.

Die in Listing 6.27 dargestellte Datei prüft, ob das ESP-Protokoll verwendet wird und zur Authentifizierung der SHA-1-HMAC und zur Verschlüsselung das AES-Verfahren genutzt wird. Wichtig ist, dass bei Erfüllung der Bedingungen der Wert `true` zurückgeliefert wird. Die Syntax hierfür ist dem Listing zu entnehmen.

Die beschriebene Richtlinie schränkt den ESP-Tunnel auf `aes-hmac-sha1` ein. Wenn dies nicht erfolgen soll, kann die Richtlinie weiter gefasst werden:

```
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" &&
            esp_auth_alg != "null" -> "true";
```

Dies erlaubt aber auch eine Verwendung von DES mit nur 56 Bit! Um dies zu verhindern, kann folgende Modifikation durchgeführt werden:

```
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" &&
            esp_enc_alg != "des" &&
            esp_auth_alg != "null" -> "true";
```

Start und Test der Verbindung

Um `isakmpd` zu starten, genügt im Normalfall das folgende Kommnado

```
# isakmpd
```

Dennoch verfügt `isakmpd` um eine Vielzahl von Optionen, die beim Start angegeben werden können. Die wichtigsten Optionen sollen kurz vorgestellt werden.

- **`-4 | -6`** Diese Optionen definieren die Adressfamilie, die `isakmpd` unterstützt. Standard sind sowohl IPv4 und IPv6.
- **`-c datei`** Hiermit kann eine alternative Konfigurationsdatei angegeben werden. Normalerweise verwendet er die Datei `/etc/isakmpd/isakmpd.conf`.
- **`-d`** Diese Option veranlasst `isakmpd` im Vordergrund zu laufen und alle Meldungen auf `stderr` auszugeben.
- **`-D class=level`** Die Option `-D` kann einen Debuglevel aktivieren. Hierbei wird eine der Klassen Misc(0), Transport(1), Message(2), Crypto(3), Timer(4), Sysdep(5), SA(6), Exchange(7), Negotiation(8), Policy(9) oder All(A) und ein Level von 0-99 angegeben, zum Beispiel: `-D A=90`.
- **`-f fifo`** Diese Option spezifiziert einen alternativen Fifo, die `isakmpd` öffnet. Über diese Named Pipe kann `isakmpd` gesteuert werden (siehe fifo weiter unten). Die Angabe `-` bindet den Fifo an `stdin`. (Default: `/var/run/isakmpd.fifo`)
- **`-L`** Wird diese Option gesetzt, so speichert `isakmpd` alle IKE-Pakete zusätzlich unverschlüsselt in einer lokalen Datei ab. Diese Datei kann mit der Option `-l` spezifiziert werden und später mit `tcpdump` oder `ethereal` betrachtet werden (Default: `/var/run/isakmpd.pcap`).

Für einen Test der Verbindungen empfiehlt es sich nicht, das oben erwähnte Kommando zu verwenden, sondern zum Beispiel: `isakmpd -4 -d -DA=50`

ACHTUNG

Wenn `isakmpd` im Vordergrund mit der Option `-d` gestartet wurde und anschließend mit Strg-C abgebrochen wird, räumt er nicht auf. Das bedeutet, dass die Security Policies und Security Associations im Speicher erhalten bleiben. Auch bei einem Neustart löscht er sie nicht. Hierfür muss manuell der Befehl `setkey` aus den KAME-Tools verwendet werden: `setkey -F; setkey -FP`. Wird `isakmpd` mit dem Signal `SIGTERM` beendet, so räumt er auf und löscht alle SAD und SPD Einträge.

Wurde `isakmpd` gestartet, so gibt es eine Reihe von Möglichkeiten ihn von außen zu steuern. Zunächst unterstützt `isakmpd` die Signale SIGHUP und SIGUSR1. Bei Empfang des Signals SIGHUP lädt `isakmpd` seine Konfigurationsdatei neu ein. Wurde in der Konfigurationsdatei `Renegotiate-on-HUP` definiert, so verhandelt er auch alle IPsec-SAs neu. Bei Empfang des SIGUSR1-Signals speichert `isakmpd` einen Bericht seines Zustandes in der Datei `/var/run/isakmpd.report`. Der Name dieser Berichtsdatei kann mit der Option `-R` bei dem Start verändert werden.

Hat der Benutzer beim Start die Option `-f` verwendet, um die Fifo-Steuerung zu aktivieren, so kann `isakmpd` anschließend mit den folgenden Befehlen gesteuert werden. Diese Befehle bestehen immer nur aus einem Buchstaben:

- **`c name`** Hiermit kann eine Verbindung gestartet werden.
- **`C {set|rm} [section]:tag=value`** Hiermit kann eine einzelne Zeile der Konfiguration geändert werden.
- **`d cookies msgid`** Löscht eine bestimmte SA.
- **`D class level`** Setzt den Debug Level.
- **`p {on|off}`** Aktiviert und deaktiviert die Protokollierung der IKE-Pakete.
- **`Q`** Beendet `isakmpd` (wie SIGTERM).
- **`r`** Erzeugt den Bericht (wie SIGUSR1).
- **`R`** Initialisiert `isakmpd` neu (wie SIGHUP).
- **`S`** Erzeugt einen Bericht mit allen SAs in der Datei `/var/run/iskampd_sa`.
- **`t name`** Beendet die Verbindung.
- **`T`** Beendet alle Verbindungen.

6.5.3 Anwendung mit einem X.509 Zertifikat

`isakmpd` unterstützt auch die Authentifizierung mit RSA-Signaturen in Form von X.509 Zertifikaten. Im Folgenden werden die Konfigurations- und die Richtliniendatei hierfür besprochen. Dabei wird allerdings nur noch auf die Unterschiede zur Anwendung mit Preshared Keys eingegangen. Der entsprechende Abschnitt sollte daher zuvor gelesen worden sein.

Die Konfigurationsdatei isakmpd.conf

Die Konfigurationsdatei `/etc/isakmpd/isakmpd.conf` ist in Listing 6.28 dargestellt.

```
[General]
Listen-on=              3.0.0.1

[Phase 1]
5.0.0.1=                ISAKMP-peer-east

[Phase 2]
Connections=            IPsec-west-east

[ISAKMP-peer-east]
Phase=                  1
Transport=              udp
Address=                5.0.0.1
Local-address=          3.0.0.1
Configuration=          Default-main-mode
ID=                     West

[West]
ID-type=                IPV4_ADDR
Address=                3.0.0.1


[IPsec-west-east]
Phase=                  2
ISAKMP-peer=            ISAKMP-peer-east
Configuration=          Default-quick-mode
Local-ID=               Net-west
Remote-ID=              Net-east

[Net-west]
ID-type=                IPV4_ADDR_SUBNET
Network=                10.0.1.0
Netmask=                255.255.255.0
```

```
[Net-east]
ID-type=                IPV4_ADDR_SUBNET
Network=                10.0.2.0
Netmask=                255.255.255.0

[Default-main-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=
3DES-SHA-RSA_SIG,3DES-MD5-RSA_SIG,BLF-MD5-RSA_SIG,BLF-SHA-
RSA_SIG

[Default-quick-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          QUICK_MODE
Suites=                 QM-ESP-AES-SHA-PFS-SUITE

[X509-certificates]
CA-directory=           /etc/isakmpd/ca/
Cert-directory=         /etc/isakmpd/certs/
CRL-directory=          /etc/isakmpd/crls
Private-key=            /etc/isakmpd/private/local.key
```

Listing 6.28 Die Datei `isakmpd.conf` für die Anwendung mit X.509 Zertifikaten

Zunächst entspricht der Aufbau der Konfigurationsdatei der bereits in Listing 6.25 gezeigten Datei. Lediglich kleinere Änderungen haben stattgefunden. Die Abschnitte `[General]`, `[Phase 1]` und `[Phase 2]` wurden unverändert übernommen. Auch der Abschnitt `[ISAKMP-peer-east]`, der den IKE-Peer beschreibt, ist größtenteils unmodifiziert. Hier wurde lediglich die Zeile `Authentication=soM59Mer` entfernt und die Zeile `ID=West` hinzugefügt. Der Parameter `Authentication` diente zur Angabe des Kennwortes für die Authentifizierung beim IKE-Peer. Da nun die Authentifizierung mit X.509 Zertifikaten erfolgt, ist diese Angabe überflüssig.

Jedoch benötigt `isakmpd` für die Identifizierung des richtigen Zertifikates, welches `isakmpd` im Verlauf der IKE-Verhandlung an den IKE-Peer sendet, eine Referenz seiner eigenen Identifikation (`ID`). Wird sie nicht angegeben, so ist das per Default die eigene IP Adresse. Hier wurde der Parameter `ID` auf den Wert `West` gesetzt. Dieser Wert `West` muss spezifiziert werden. Zunächst wird der Typ der ID mit `ID-type` angegeben. Folgende Typen sind möglich:

- **`IPV4_ADDR`** Eine IPv4-Adresse.
- **`IPV4_ADDR_SUBNET`** Ein IPv4-Netzwerk.
- **`IPV6_ADDR`** Eine IPv6-Adresse.
- **`IPV6_ADDR_SUBNET`** Ein IPv6-Netzwerk.

- **FQDN** Ein vollqualifizierter DNS-Name.
- **USER_FQDN** Eine E-Mail-Adresse.
- **KEY_ID** Eine beliebige Bytefolge die jedoch durch druckbare Zeichen dargestellt werden muss. Diese Funktion hat jedoch keine reale Relevanz.

Wurde der Typ definiert, so existieren in Abhängigkeit des Typen weitere Parameter, mit denen dann der Wert definiert werden kann: `Address`, `Network` und `Netmask` für die Spezifizierung der IDs vom Typ `*_ADDR` und `*_SUBNET`. Für die Angabe der Typen `FQDN`, `USER_FQDN` und `KEY_ID` wird der Parameter `Name` verwendet.

```
[RalfSpenneberg]
ID-type=        USER_FQDN
Name=           ralf@spenneberg.net
```

Die nächste Modifikation in der Datei `isakmpd.conf` für die Verwendung von X.509 Zertifikaten erfolgt im Abschnitt `[Default-main-mode]`. Hier wird der Tatsache Rechnung getragen, dass nun die Authentifizierung mit X.509 Zertifikaten erfolgt. Der RFC-konforme Begriff ist hier die Authentifizierung mit RSA-Signaturen. Daher wurden alle angegebenen `Transforms` in ihre entsprechenden Pendants mit dem Suffix `RSA_SIG` umgewandelt. Die Reihenfolge der angegebenen Transforms definiert die Priorität der Transforms. Das erste besitzt die höchste Priorität.

Schließlich benötigt `isakmpd` noch die Angabe der Zertifikatsverzeichnisse. Hierzu dient der Abschnitt `[X509-Certificates]`. Die angegebenen Verzeichnisse sind die Default-Verzeichnisse und wurden hier nur zur Demonstration und zur Vollständigkeit angegeben.

Das `CA-directory` enthält die Zertifikate der vertrauten Zertifikatsautoritäten. Damit `isakmpd` tatsächlich diesen CAs vertraut und von ihnen unterzeichnete Zertifikate akzeptiert, müssen diese CAs in der Datei `isakmpd.policy` referenziert werden (siehe unten).

Das `Cert-directory` enthält das eigene Zertifikat und kann weitere Peer-Zertifikate enthalten, die Vorrang vor den während der IKE-Verhandlung übertragenen Zertifikaten haben.

Das `CRL-directory` enhält mögliche Rückruflisten der CAs im PEM-Format.

Der letze Eintrag `Private-key` weist schließlich auf den privaten Schlüssel des VPN-Gateways hin. Dieser RSA-Schlüssel wird für die Authentifizierung in Phase 1 verwendet.

Die Richtliniendatei isakmpd.policy

Auch in der Richtliniendatei `/etc/isakmpd/isakmpd.policy` sind geringe Änderungen erforderlich. Die Richtliniendatei für die Verwendung mit X.509 Zertifikaten ist im Listing 6.29 dargestellt.

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensee: "DN:/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.Com/
OU=VPN/CN=RootCA"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg == "aes" &&
            esp_auth_alg == "hmac-sha" -> "true";
```

Listing 6.29 Die Richtliniendatei `isakmpd.policy` für X.509 Zertifikate

Die wesentliche Änderung in dieser Datei ist die Angabe des `Licensees`. Hier wurde bei der Verwendung des Preshared Key die Passphrase angegeben. Bei der Verwendung von RSA-Schlüssel in Form von X.509 Zertifikaten stehen zwei verschiedene Möglichkeiten zur Wahl. Entweder wird der öffentliche Schlüssel direkt angegeben oder über den Distinguished Name (`DN`) das Zertifikat referenziert. Handelt es sich hierbei um ein Zertifikat einer CA, so vertraut `isakmpd` allen gültigen, von der CA unterzeichneten und nicht widerrufenen Zertifikaten. Die Verwendung des `DN` ist die einfachere und empfohlene Variante.

Um den öffentlichen Schlüssel direkt anzugeben, wird die folgenden Syntax verwendet:

```
licensees: "x509-base64:MIICGDCCAYGgAwIBAgIBADANBgkqhkiG9w0BAQQ\
            FADBSMQswCQYDVQQGEwJHQjEOMAwGA1UEChMFQmVuQ28xETAPBg\
            NVBAMTCEJlbkNvIENBMSAwHgYJKoZIhvcNAQkBFhFiZW5AYWxnc\
            m91cC5jby51azAeFw05OTEwMTEyMjQ5MzhaFw05OTExMTAyMjQ5\
            MzhaMFIxCzAJBgNVBAYTAkdCMQ4wDAYDVQQKEwVCZW5DbzERMA8\
            GA1UEAxMIQmVuQ28gQ0ExIDAeBgkqhkiG9w0BCQEWEWJlbkBhbG\
            dyb3VwLmNvLnVrMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBg\
            QCxyAte2HEVouXg1Yu+vDihbnjDRn+6k00Rv6cZqbwA3BQ30mC/\
            3TFJ09VGXCaM0UKfpnxIpkBYLmOA3FWkKI0RvPU7E1AhKkhC1Ds\
            PSBFjYHrB15T5lYzgfwKJCIxTDzZDx2iobUgPa0FRNGVUjpQ4/k\
            MJ2BF4Wh7zY3X08rMzsQIDAQABMA0GCSqGSIb3DQEBBAUAA4GBA\
            DWJ5pbTcE7iKHWLQTMYiz8i9jGi5+Eo1yr1Bab90tgaGQV0zrRH\
            jDHgAAy1h8WSXuyQrXfgbx2rnWFPhx9CfmuAXn7sZmQE3mnUqeP\
            ZL2dW87jdBGqtoUdNcoz5zKBkC943yasNui/O01MiqgadTThTJH\
            d1Pn17LbJC1ZVRNjR5"
```

Listing 6.30 Beispiel eines X.509-Schlüssels aus der Manpage

```
licensees: "DN:/CN=ralfspenneberg/Email=ralf@spenneberg.net"
```

Listing 6.31 Beispiel für die Verwendung des Distinguished Name

Um den *Distinguished Name* des Zertifikates zu erhalten, kann der folgende Befehl verwendet werden:

```
# openssl x509 -in /etc/isakmpd/ca/ca.crt -noout -subject
subject= /C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.Com/OU=VPN/CN=RootCA
```

Wurde die Datei `isakmpd.policy` entsprechend erzeugt, so müssen nun noch die RSA-Schlüssel in Form des privaten Schlüssels, des X.509 Zertifikates, des CA-Zertifikates und der CRL an den entsprechenden Stellen abgespeichert werden.

Die Erzeugung und Speicherung der X.509 Zertifikate

Bei der Erzeugung und der Speicherung der RSA-Schlüssel für `isakmpd` sind einige Besonderheiten zu beachten.

Der private RSA-Schlüssel wird in der Datei `/etc/isakmpd/private/local.key` gespeichert. Dieser Schlüssel darf nicht mit einer Passphrase geschützt sein und muss die Rechte `600` aufweisen.

```
# chmod 600 /etc/isakmpd/private/local.key
```

Um einen geschützten Schlüssel freizuschalten, kann der folgende Befehl verwendet werden.

```
# openssl rsa -in key_passphrase.pem -out key_nopassphrase.pem
```

Das X.509 Zertifikat des Systems wird im Verzeichnis `/etc/isakmpd/certs` gespeichert. Hierbei ist es wichtig, dass das Zertifikat über einen `subjectAltName` verfügt. Dieser `subjectAltName` muss mit der `ID` in der Datei `/etc/isakmpd/isakmpd.conf` übereinstimmen. Wurde dort als `ID` ein `FQDN` verwendet, so muss auch der `subjectAltName` derselbe `FQDN` sein. Das Zertifikat wird dann unter dem Namen `<FQDN>.crt` abgespeichert. Um dem Zertifikat den `subjectAltname` hinzuzufügen, kann entweder bei der Erzeugung bereits diese X.509.v3-Extension angegeben werden (siehe Kapitel »Public Key Infrastructure«) oder nachträglich der Befehl `certpatch` verwendet werden, der im `isakmpd`-Paket enthalten ist. Hierfür ist aber der private Schlüssel der CA erforderlich.

Dies soll hier durch ein Beispiel verdeutlicht werden. Als `ID` wird der `FQDN` `vpn.spenneberg.net` verwendet. In der Konfigurationsdatei `/etc/isakmpd/isakmpd.conf` ist daher folgender Abschnitt erforderlich.

```
[West-ID]
ID-type=      FQDN
Name=         vpn.spenneberg.net
```

Das Zertifikat benötigt den `subjectAltName vpn.spenneberg.net`. Dieser wird mit dem Befehl `certpatch` eingetragen.

```
# certpatch -t fqdn -i vpn.spenneberg.net -k /etc/ssl/ca/ca.key \
> cert.pem /etc/isakmpd/certs/vpn.spenneberg.net.crt
```

Das Zertifikat der CA wird im PEM-Format im Verzeichnis `/etc/isakmpd/ca` abgespeichert. Der gewählte Name ist hierbei unerheblich.

6.5.4 Roadwarrior und isakmpd

Wie bereits weiter oben ausgeführt wurde, wird die Direktive `Passive-connections` kaum verwendet. Wenn `isakmpd` als VPN-Gateway lediglich VPN-Verbindungen entgegen nehmen darf, aber diese nicht selbst aufbauen soll, so wird dort in der Konfigurationsdatei keine Verbindung definiert. Eine mögliche `isakmpd.conf` ist in Listing 6.32 gezeigt.

```
[General]
Listen-on=              3.0.0.1

[Phase 1]
Default=                ISAKMP-VPN

[ISAKMP-VPN]
Phase=                  1
Configuration=          Default-main-mode
ID=                     VPN-GW

[VPN-GW]
ID-type=                IPV4_ADDR
Address=                3.0.0.1

[Default-main-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=             BLF-SHA-RSA_SIG,BLF-MD5-RSA_SIG,3DES-SHA-RSA_SIG

[X509-certificates]
CA-directory=           /etc/isakmpd/ca/
Cert-directory=         /etc/isakmpd/certs/
Private-key=            /etc/isakmpd/private/local.key
```

Listing 6.32 Die Datei `isakmpd.conf` auf einem Roadwarrior-Gateway

Die Authentifizierung wird lediglich über die Richtliniendatei `/etc/isakmpd/isakmpd.policy` geregelt. Hier kann die Datei unverändert übernommen werden. Sie ist zur Referenz hier noch einmal abgedruckt.

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensee: "DN:/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.Com/
OU=VPN/CN=RootCA"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg == "aes" &&
            esp_auth_alg == "hmac-sha" -> "true";
```

Diese Datei stellt nun sicher, dass nur authentifizierte Clients sich verbinden dürfen. Einen Nachteil hat diese Methode jedoch: Der Client spezifiziert die im Tunnel erlaubten IP Adressen. Dies kann, in Abhängigkeit der verwendeten Netzwerkstruktur, eine Sicherheitslücke sein. Bei Verwendung des Parameters `Passive-connections` kann der Server die erlaubten IP Adressbereiche spezifizieren.

6.5.5 Aggressive Modus und PSKs

Bei der Verwendung eines Preshared Key sucht `isakmpd` nach der `Authentication`-Information im Abschnitt, der den ISAKMP-Peer beschreibt. Wenn verschiedene Clients mit unbekannten IP Adressen eine Verbindung auf der Basis von Preshared Keys aufbauen möchten, so müssen alle diese Clients denselben Preshared Key verwenden. Um dies zu vermeiden, kann der Aggressive Modus verwendet werden. Dieser bietet keine verschlüsselte Übertragung der Identität, aber dadurch die Möglichkeit, dass in Abhängigkeit der Identität unterschiedliche Preshared Keys verwendet werden können. Hierzu ist die Konfigurationsdatei folgendermaßen anzupassen.

```
[ISAKMP-peer]
Phase=          1
Configuration=  aggrmode

# Dial-in VPN Accounts

[user@domain]
Phase=          1
Configuration=  aggrmode
Authentication= somepassphrase
```

```
[user2@domain]
Phase=          1
Configuration=  aggrmode
Authentication= some2passphrase

[aggrmode]
DOI=            IPSEC
EXCHANGE_TYPE=  AGGRESSIVE
Transforms=     3DES-MD5,3DES-SHA
```

Damit dies nun funktioniert, muss der Client bei der Einwahl die angegebene ID (entweder `user@domain` oder `user2@domain`) und das entsprechende Kennwort verwenden.

ACHTUNG

Der Aggressive Modus weist einige Sicherheitsprobleme auf. Da die Identifikation nun im Klartext übertragen wird, muss ein Angreifer lediglich das verwendete Kennwort knacken. Hierfür stehen einige Werkzeuge (wie `ikecrack`) zur Verfügung (siehe Kapitel Protokolle). Damit die Authentifizierung sicher erfolgen kann, müssen gute PSKs gewählt werden! Außerdem besteht beim Aggressive Modus die Gefahr des Denial-of-Service-Angriffes.

6.5.6 IKE Config Mode

`isakmpd` kann als Responder in einer VPN-Verbindung dem Initiator mit der IKE-Config-Methode die zu verwendende IP Adresse, DNS-Server und WINS-Server mitteilen. So kann `isakmpd` als VPN-Gateway virtuelle IP Adresse, ähnlich FreeS/WAN mit X.509-Patch und DHCP-Relay verteilen. Ein Client, der diese Funktion nutzen kann, ist zum Beispiel der SSH-Sentinel.

Um diese Funktion zu nutzen, muss `isakmpd` mit der Option `isakmp_cfg` übersetzt werden. Diese Option ist normalerweise im `Makefile` bereits aktiviert.

Anschließend kann in der Konfigurationsdatei `/etc/isakmpd/isakmpd.conf` für jeden Client ein Abschnitt eingetragen werden, der die entsprechenden Informationen trägt. Die Verwendung eines Adressen-Pools, wie bei DHCP, ist hier nicht möglich. Der Name des Abschnittes besteht aus dem `ID-type` und dem Namen, den der Client bei seiner Anmeldung verwendet, zum Beispiel: `[ipv4/5.0.0.1]` oder `[ufqdn/ralf@spenneberg.net]`. Dabei unterstützt `isakmpd` hier noch einen zusätzlichen Typ:`ASN1_DN`. Hier ein Beispiel: `[asn1_dn//C=DE/O=../..]`.

Ein kompletter IKE-Config-Abschnitt ist in dem folgenden Beispiel zu sehen:

```
[ufqdn/ralf@spenneberg.net]
Address=        10.0.2.200
Netmask=        255.255.255.0
Nameserver=     10.0.2.1
WINS-server=    10.0.2.1
```

Listing 6.33 IKE-Config mit dem `isakmpd`

Damit der `isakmpd` auch den IKE-Config Modus aktiviert, ist in der Beschreibung der Phase 1 die Angabe des Flags `ikecfg` erforderlich.

```
[ISAKMP-peer]
Phase=          1
Configuration=  aggrmode
Flags=          ikecfg
```

6.5.7 Fazit

`isakmpd` aus dem OpenBSD-Projekt ist ein sehr mächtiger IKE-Daemon. Er stellt eine ernst zu nehmende Alternative zu `racoon` aus dem KAME-Projekt dar. Beide IKE-Daemonen sind in der Lage im Main Modus und Aggressive Modus mit Preshared Keys und X.509 Zertifikaten den Aufbau von VPN Verbindungen zu ermöglichen. `isakmpd` bietet mit der zusätzlichen Unterstützung des IKE Config Modus die Möglichkeit, virtuelle IP Adressen an die Clients zu verteilen. Leider bieten beide noch nicht die Unterstützung des NAT Traversal oder die Verteilung virtueller IP Adressen mit DHCP-over-IPsec. Hierzu ist im Moment nur FreeS/WAN mit den entsprechenden Patches in der Lage.

7 Aufbau heterogener Virtueller Privater Netze

Der Aufbau eines Virtuellen Privaten Netzes mit Linux ist recht einfach. Es kann zwar auch hier zu Problemen kommen, jedoch sind sie meist relativ einfach und schnell zu lösen, da an beiden Enden des Tunnels häufig identische Implementierungen sind.

Schwieriger wird der Aufbau von IPsec VPNs, wenn es sich um unterschiedliche IPsec-Implementierungen handelt, die eingesetzt werden. Trotz der Verfügbarkeit von Standards kommt es besonders bei dem IKE Protokoll immer wieder zu Problemen in der Interoperabilität. In diesem Kapitel werden einige Lösungen für die Anbindung von weiteren Systemen an ein Linux-VPN genannt.

7.1 Einleitung

Häufig werden Virtuelle Private Netzwerke mit unterschiedlichen Produkten realisiert. Dies kann verschiedene Gründe haben. Möglicherweise handelt es sich um eine VPN-Lösung zwischen zwei verschiedenen Firmen, die unterschiedliche Produkte einsetzen. Hier wird jede Firma das Produkt wählen, das für sie am günstigsten in der Anschaffung und Unterhaltung ist.

In vielen Fällen soll über das VPN aber auch ein Anschluss für Telearbeiter angeboten werden. Hierbei sind wieder viele verschiedene Möglichkeiten gegeben. Sie können über eigenständige Router, die eine IPsec Unterstützung bieten, angeschlossen werden oder das eingesetzte Betriebssystem kann selbst die IPsec Verbindung aufbauen. Da viele moderne Betriebssysteme inzwischen über einen eigenen IPsec Stack verfügen, ist dies die einfachste und preisgünstigste Variante.

In diesem Kapitel wird die Interoperabilität mit den verschiedenen Windows-Betriebssystemen, Cisco und Checkpoint-Produkten dargestellt. Insbesondere die hier beschriebenen einzelnen Einstellungen können immer wieder durch die Hersteller geändert werden. In den Fällen, in denen Sie nicht mit den hier beschriebenen Werten erfolgreich sind, oder bessere alternative Wege ermittelt haben, würde ich mich sehr über eine E-Mail an *ralf.spenneberg@mut.de* freuen.

7.2 Interoperabilitätsprobleme

Auch wenn IPsec ein Internet Standard ist, so zeigt bereits die tägliche Erfahrung, dass es für jede Regel auch Ausnahmen gibt. Dies gilt insbesondere für die IPsec Protokolle. Die IPsec-Protokolle ESP und AH selbst sorgen in den wenigsten Fällen für Probleme. Die verwendeten Verschlüsselungs- und Authentifizierungsverfahren sind aber bereits häufig Anlass für Interoperabilitätsprobleme. Die meisten Probleme treten jedoch beim IKE Protokoll auf. Hier existieren sehr viele verschiedene Parameter, die von den verschiedenen Implementierungen unterschiedlich gesetzt werden.

Da gerade das IKE Protokoll diese Probleme erzeugt, konzentrieren sich die meisten Interoperabilitätstests auf dieses Protokoll. So wird bei der jährlichen IPsec-Konferenz in Frankreich meist auch ein IKE Interoperabilitätstest verschiedener Hersteller durchgeführt (*http://www.hsc.fr/ressources/ipsec/ipsec2001/*).

Schließlich implementieren einige Hersteller gerade im IKE Protokoll eigene proprietäre Erweiterungen, um ihren Kunden zusätzliche Funktionen zu bieten und die Verwendung der häufig kostenlosen Client Software mit freien IPsec Implementierungen zu verhindern.

Die folgenden Kapitel zeigen den Aufbau von VPN Verbindungen der entsprechenden Produkte mit Linux und geben zusätzliche Hinweise.

7.3 Microsoft Windows 98/Me/NT

Diese Betriebssysteme verfügen von Haus aus über keine Unterstützung für die IPsec Protokolle. Für den Aufbau eines Virtuellen Privaten Netzwerks hat Microsoft bei der Veröffentlichung dieser Systeme das Point-to-Point-Tunneling Protokoll (PPTP) verwendet.

Wenn diese Betriebssysteme als Client in einem Linux basierten VPN eingesetzt werden sollen, existieren zwei mögliche Ansätze:

1. Für den Aufbau des VPNs wird das PPTP Protokoll verwendet. Es existiert ein Open-Source-PPTP Server (*http://www.poptop.org*) für Linux, der in der Lage ist mit 128 Bit verschlüsselte VPN Verbindungen aufzubauen. Vorteil dieser Lösung ist die Tatsache, dass auf dem Client keine zusätzliche Software installiert werden muss. Als Nachteil wiegt jedoch die zusätzliche Installation und Pflege des PPTP Daemons auf dem Linux System und die geringere Sicherheit des PPTP Protokolls (*http://www.counterpane.com/pptp.html*).

2. Für den Aufbau des VPNs wird das IPsec Protokoll verwendet. Dann wird eine zusätzliche Software auf dem Windows Client benötigt, die in der Lage ist einen IPsec Tunnel aufzubauen. Hier existieren eine ganze Reihe von Werkzeugen, deren Interoperabilität mit Linux getestet wurde:
   - SSH Sentinel: *http://www.ssh.com/products/security/sentinel/*
   - F-Secure VPN+: *http://www.f-secure.com/products/vpnplus/index.shtml*
   - SoftRemote VPN: *http://www.securecomputing.com/index.cfm?skey=343*
   - Kostenloser Microsoft MSL2TP Client: *http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/l2tpclient.asp*. Hierfür ist auf der Linux Seite zusätzlich die Unterstützung des L2TP-Protokolls erforderlich. Die Konfiguration dieses Protokolls wird im Kapitel 9, »Fortgeschrittene Konfiguration« erläutert.
   - Weitere Werkzeuge werden auf der unten angegebenen FreeS/WAN Seite aufgeführt.

Da sich die verschiedenen Werkzeuge stark in ihrer Konfiguration unterscheiden, kann hier nicht auf die Konfiguration im einzelnen eingegangen werden. Es existieren jedoch im Internet eine Vielzahl von Anleitungen, die die Konfiguration beschreiben. Als zentrales Repositorium dieser Dokumente kann die FreeS/WAN-Seite *http://www.freeswan.org/freeswan_trees/freeswan-2.01/doc/interop.html* genutzt werden.

7.4 Microsoft Windows 2000 und Windows XP

Microsoft hat mit der Version Windows 2000 begonnen, seine Betriebssysteme mit einem IPsec Stack auszustatten. Dieser IPsec Stack ist in der Lage, die Protokolle AH und ESP für die Authentifizierung und Verschlüsselung von IP Paketen zu verwenden. Zusätzlich wurde das IKE Protokoll implementiert, das mit Preshared Keys oder X.509 Zertifikaten eine Authentifizierung des VPN Peers durchführen kann. Für die Speicherung der X.509 Zertifikate verfügen diese Betriebssysteme nun über einen zentralen Zertifikatsspeicher, der auch von anderen Programmen, wie dem Internet Explorer, genutzt werden kann.

Die Verwaltung dieses IPsec Stacks und des IKE Protokolls erfolgt über den IP Security Policies Wizard (Abbildung 7.1). Dieser Wizard wird über die Microsoft Management Console gestartet (Abbildung 7.2).

Abbildung 7.1 Der Windows 2000 IP-Security-Policy-Wizard

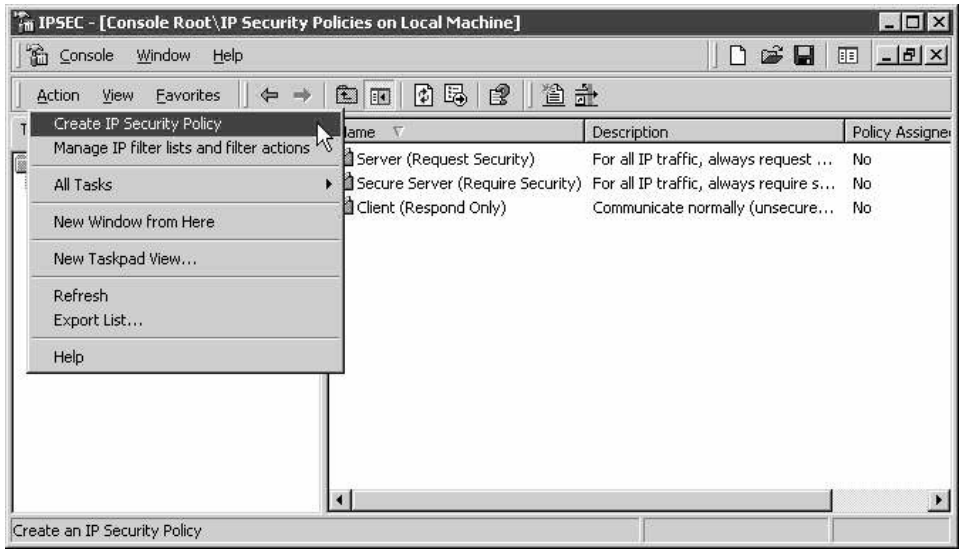


Abbildung 7.2 Die Microsoft-Management-Console startet den IPsec-Wizard

Obwohl es sich bei dieser Anwendung um einen »Zauberer« handelt, ist die Konfiguration eines IPsec-Tunnels mit diesem Werkzeug sehr umständlich und fehlerträchtig. Der geneigte Leser findet auf der Homepage von Jean-

Francois Nadeau (*http://jixen.tripod.com/*) eine ausführliche Anleitung mit 900 kByte Screenshots.

Wird dieser Wizard eingesetzt, so verlangt er die fixe Angabe der eigenen IP Adresse für die Konfiguration des IPsec-Tunnels. Handelt es sich bei dem Microsoft Windows System um einen Roadwarrior, der über eine unbekannte und sich ständig ändernde IP Adresse verfügt, so muss der Administrator (ein Benutzer verfügt nicht über ausreichende Rechte) jedesmal, nachdem eine neue IP Adresse vergeben wurde, die IPsec Policies neu konfigurieren. Dadurch ist der Wizard in einer derartigen Umgebung unbrauchbar.

Um in einer Roadwarrior Umgebung eingesetzt zu werden, bestehen auf Windows 2000/XP/2003 nun grundsätzlich zwei verschiedene Möglichkeiten:

- Installation eines alternativen IPsec Stacks mit grafischem Administrationswerkzeug, das ein Roadwarrior Szenario unterstützt. Hier sind dieselben Programme verfügbar, die bereits im letzten Kapitel über Windows 98/ME/NT aufgezählt wurden. Auf diese Werkzeuge soll hier nicht eingegangen werden.
- Verwendung des nativen IPsec Stacks mit einem alternativen Konfigurationswerkzeug. Markus Müller hat entdeckt, dass Microsoft für die Konfiguration des IPsec Stacks auch ein Kommandozeilenwerkzeug zur Verfügung stellt. Hierfür hat er einen kleinen Wrapper geschrieben, der mit diesem Werkzeug automatisch den nativen IPsec Stack konfiguriert.

7.4.1 Markus Müllers ipsec.exe

Markus Müller (*http://vpn.ebootis.de*) hat mit dem Kommando `ipsec.exe` ein Open Source Werkzeug geschaffen, das in der Lage ist, mit nativen Windows 2000/XP Mitteln die komfortable Konfiguration eines IPsec Tunnels zu erlauben. Dabei unterstützt dieses Werkzeug auch die Konfiguration des Windows Clients als Roadwarrior, da es selbstständig in der Lage ist die aktuelle IP Adresse zu ermitteln und automatisch zu verwenden. Ist im Vorfeld die Einwahl ins Internet erforderlich, so kann das Werkzeug dies veranlassen.

Die beste Eigenschaft dieses Werkzeuges stellt aber die Konfiguration dar. Markus Müller verwendet eine Konfigurationssyntax, die der FreeS/WAN-Syntax fast auf den I-Punkt gleicht.

```
conn %default
      dial=<providername>
```

```
conn VPN
      left=%any
      right=3.0.0.1
      rightsubnet=10.0.1.0/24
      rightca="C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
       OU=Wireless-VPN, CN=RootCA, E=ralf@spenneberg.net"
      network=auto
      auto=start
      rekey=1800S/30000K
      authmode=MD5
      pfs=yes
```

Listing 7.1 Konfigurationsdatei `ipsec.conf` für `ipsec.exe`

Im Folgenden werden die notwendigen Schritte für den erfolgreichen Aufbau eines VPNs vorgestellt.

Installation der notwendigen Programme

Vorraussetzung für den Einsatz des Programms von Markus Müller ist die Installation von Windows 2000 oder XP. Wenn als Plattform Windows 2000 eingesetzt wird, so ist zusätzlich die Installation des Servicepacks mindestens der Version 2 erforderlich. Dieses Servicepack 2 erweitert die in Windows 2000 eingebauten Verschlüsselsalgorithmen um starke Verfahren. Als Windows 2000 ursprünglich international ausgeliefert wurde, waren die Exportbeschränkungen der USA noch in Kraft und verhinderten den Export starker Kryptografie. Windows XP enthält bereits die entsprechenden starken Kryptoalgorithmen.

Weiterhin ist die Installation des Kommandozeilenclients (Command Line Interface, CLI) für die Konfiguration der IPsec-Richtlinien erforderlich. Im Fall von Windows 2000 ist dieses Werkzeug nicht auf ausgelieferten CDs enthalten, sondern Bestandteil des Windows 2000 Ressource-Kits. Speziell das Werkzeug `ipsecpol.exe` wird jedoch von Microsoft zum Download unter *http://agent.microsoft.com/windows2000/techinfo/reskit/tools/existing/ipsecpol-o.asp* zur Verfügung gestellt.

Der IPsec Richtlinien CLI für Windows XP ist auf Installations CD im Verzeichnis `\SUPPORT\TOOLS` enthalten. Nach dem Start des Programmes `setup.exe` in diesem Verzeichnis ist die komplette Installation für die Installation des Programms `ipseccmd.exe` auszuwählen.

Nach der Installation des entsprechenden CLI Werkzeuges in einem geeigneten Verzeichnis (zum Beispiel `C:\Programme\VPN`), kann es durch das Werkzeug von Markus Müller genutzt werden. Dieses Werkzeug muss nun ebenfalls installiert werden. Hierzu kann es in binärer Form (*http://vpn.*

ebootis.de/package.zip, empfohlen) oder als Quelltext (*http://vpn.ebootis.de/source.zip*) geladen werden. In binärer Form ist es sofort lauffähig und muss lediglich im Verzeichnis `c:\Programme\VPN` ausgepackt werden. Wurde der Quelltext geladen, so muss das Programm zunächst mit einem geeigneten Compiler übersetzt werden.

Auf der Linux Seite ist die Installation einer IPsec Lösung erforderlich. Hierbei existieren die meisten Erfahrungen mit FreeS/WAN, jedoch ist es auch möglich eine Verbindung zu `racoon` oder `isakmpd` aufzubauen. Wird FreeS/WAN eingesetzt, so wird stark die Verwendung von Super-FreeS/WAN empfohlen. Sollen die Patches manuell hinzugefügt werden, so ist der Einsatz des X.509-Patches und des Delete-SA/Notify SA Patches sinnvoll.

Konfiguration mit PSKs

Die Konfiguration der Linux Seite mit einem Preshared Key wird in den entsprechenden Kapiteln ausführlich besprochen. Wichtig ist in Abhängigkeit der verwendeten Linux Lösung die Tatsache, dass Windows lediglich DES und 3DES zur Verschlüsselung und MD5 und SHA1 als HMAC unterstützt. Die Unterstützung von PFS ist optional. Diese Einschränkung muss bei der Konfiguration der Linux Seite berücksichtigt werden. Zur Authentifizierung wird ein Preshared Key genutzt.

Die Konfiguration des Windows Clients ist recht einfach. Zunächst wird die Konfigurationsdatei `C:\Programme\VPN\ipsec.conf` aus Listing 7.2 erzeugt.

```
conn %default
      dial=<providername>

conn VPN
      left=%any
      right=3.0.0.1
      rightsubnet=10.0.1.0/24
      presharedkey=Passphrase
      network=auto
      auto=start
      rekey=1800S/30000K
      authmode=MD5
      pfs=yes
```

Listing 7.2 Die Windows-Konfigurationsdatei `ipsec.conf` für die Verwendung von Preshared Keys

Die Implementierung eines IPsec Tunnels mit Preshared Keys ist unter Windows sehr einfach. Daher kann dies auch als erster Schritt nur empfohlen werden. Sobald dieser Tunnel funktioniert, kann die Authentifizierung auf X.509 Zertifikate umgestellt werden.

Sobald die Konfigurationsdatei erzeugt wurde, kann sie mit dem Werkzeug `ipsec.exe` in Windows umgesetzt werden. Hierzu wird der Befehl aufgerufen.

```
ipsec
IPSec Version 2.1.4 (c) 2001,2002 Marcus Mueller
Getting running Config ...
Microsoft's Windows XP identified
Host name is: <Rechner>
No RAS connections found.
LAN IP address: <IP Adresse>
Setting up IPSec ...

Deactivating old policy...
Removing old policy...

Connection VPN:
MyTunnel : <IP Adresse>
MyNet : <IP Adresse>/255.255.255.255
PartnerTunnel: 3.0.0.1
PartnerNet : 10.0.1.0/255.255.255.0
PFS : y

Auto : start
Auth.Mode : MD5
Rekeying : 3600S/50000K
Activating policy...
```

Nun ist die Richtlinie geladen. Damit ist aber noch kein Tunnel gestartet worden. Windows startet den Tunnel erst bei Bedarf, und zwar mit einem Ping. Dabei kann es sein, dass zu Beginn der Ping noch nicht erfolgreich ist, da der Tunnel noch nicht steht. Dann erscheint auf dem Bildschirm die Meldung »Negotiating IP Security«. Wenn diese Meldung nicht durch einen erfolgreichen Ping nach einigen Sekunden abgelöst wird, wurde ein Fehler in der Konfiguration gemacht. Hier hilft dann nur eine Kontrolle der Konfigurationsdateien und ein Blick in die Protokolle. Microsoft beschreibt im Knowledge-Base-Artikel 257225 (*http://support.microsoft.com:80/support/kb/articles/Q257/2/25.ASP*), wie die Protokollierung unter Windows aktiviert wird (`Oakley.log`). Hierzu ist der folgende Eintrag in der Registrierung erforderlich.

```
HKEY_LOCAL_MACHINE\SYSTEM\
CurrentControlSet\Services\PolicyAgent\Oakley\EnableLogging=1
```

Listing 7.3 Aktivierung der Protokollierung unter Windows

Das Windows Betriebssystem baut den Tunnel auch automatisch nach einiger Zeit wieder ab. Dabei sendet es eine Delete SA Notification. Sinnvollerweise unterstützt die Linux Seite diese Nachricht. FreeS/WAN benötigt dafür einen Patch.

Wenn der Aufbau des Tunnels mit Preshared Keys erfolgreich ist, kann die Authentifizierung auf X.509 Zertifikate umgestelllt werden.

Konfiguration mit einem X.509 Zertifikat

Die Verwendung von X.509 Zertifikaten im Kombinations mit Windows 2000 und Windows XP ist häufig zu Beginn mit Fehlern verbunden. Daher ist es sinnvoll zunächst die Konfiguration des Tunnels mit Preshared Keys zu testen. Sobald dies funktioniert, kann die Authentifizierung auf X.509 Zertifikate umgestellt werden.

Die Konfiguration der Linux-VPN-Lösungen mit X.509 Zertifikaten wird an anderen Stellen in diesem Buch ausführlich besprochen. Hier ist es nur erforderlich, die bereits im letzten Abschnitt angesprochenen Einschränkungen zu berücksichtigen.

Damit der Windows-Client nun X.509 Zertifikate nutzen kann, benötigt er einen privaten Schlüssel, das entsprechende X.509 Zertifikat und das Zertifikat der CA, die die Zertifikate ausgestellt hat. Windows unterstützt für den einfachen Import dieser Informationen das PKCS#12-Format. Diese Dateien werden üblicherweise mit der Endung `.p12` versehen. Die meisten grafischen CAs unterstützen direkt den Export der Informationen in diesem Format.

ACHTUNG

Der Kommandozeilenbefehl `ipsec.exe` kann nicht mit Umlauten und einigen anderen Sonderzeichen im DN der CA umgehen. Daher sollten derartige Zeichen bei der Erzeugung der CA nicht im DN verwendet werden!

Liegen die Dateien bereits einzeln vor, können sie aber auch mit dem Kommando `openssl` in dieses Format exportiert werden.

```
# openssl pkcs12 -export -in WinClient_cert.pem \
 -inkey WinClient_req.pem -certfile cacert.pem \
 -out WinClient.p12
```

Beim Aufruf dieses Befehls muss der Benutzer bis zu zwei Kennwörter eingeben. Das erste Kennwort wird verwendet, um den mit einer Passphrase geschützten privaten Schlüssel lesen zu können. Das zweite Kennwort wird als Exportkennwort verwendet, um den privaten Schlüssel erneut zu verschlüsseln. Dieses Kennwort ist beim Import in den Windows-Zertifikatsspeicher erforderlich.

Diese Datei muss nun auf den Windows-Rechner übertragen werden und kann dort durch einen Doppelklick importiert werden. Dabei ist die Angabe des Export-Kennwortes erforderlich. Beim Import ist es wichtig, dass die Zertifikate in den richtigen Speichern abgelegt werden. Der Windows Zertifikatsspeicher sollte normalerweise in der Lage sein, die Rolle der Zertifikate automatisch zu erkennen. Daher kann beim Import `automatisch` ausgewählt werden (Abbildung 7.3).

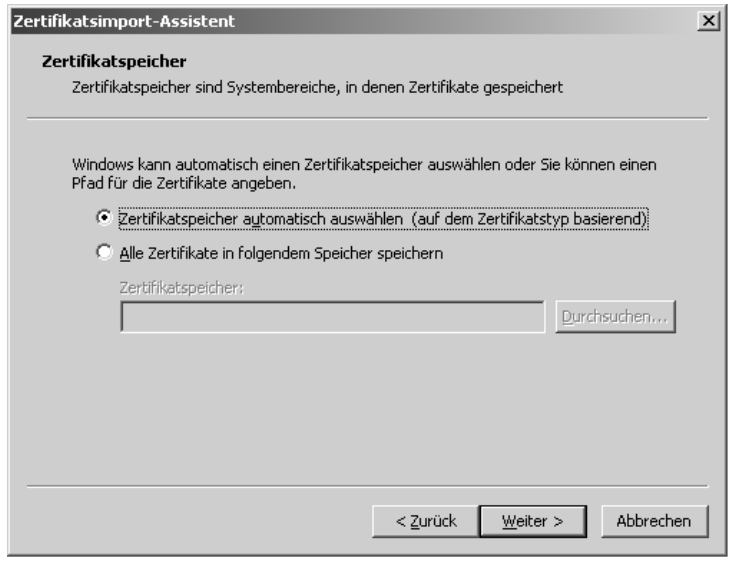


Abbildung 7.3 Automatischer Import der Zertifikate

Anschließend sollte jedoch kontrolliert werden, ob tatsächlich die Zertifikate in die entsprechenden Speicher kopiert wurden. Hierzu hat Markus Müller im Paket eine MMC Konfiguration unter dem Namen `ipsec.msc` abgespeichert. Ein Doppelklick auf diese Datei öffnet die MMC mit dem Zugriff auf den Zertifikatsspeicher (Abbildung 7.4). Wenn anschließend die Zertifikate nicht gefunden werden können, ist der Import zu wiederholen und dabei die Zertifikatsspeicher von Hand auszuwählen. Anschließend können die Zerti-

fikate mit Drag-and-Drop in die richtigen Speicher verschoben werden. Das Rechnerzertifikat muss sich im Speicher »Eigene Zertifikate« befinden, das Zertifikat der CA im Speicher »Vertrauenswürdige Stammzertifikate«.

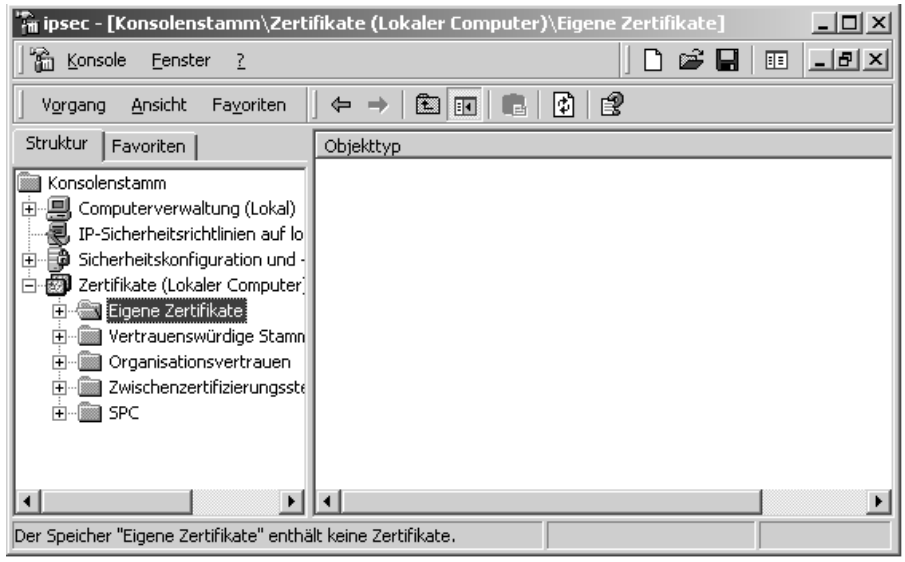


Abbildung 7.4 Anzeige der Zertifikatsspeicher mit der MMC

Ein Doppelklick auf die entsprechenden Zertifikate sollte zusätzlich deren Gültigkeit bestätigen. Sie sollte überprüft werden, da häufiger Zeit-Probleme auftreten, wenn die Zertifikate auf einem Linux-Rechner erzeugt werden und dieser eine andere Zeitzone verwendet als der Windows-Rechner. Die Zertifikate sind dann möglicherweise unter Windows noch nicht gültig!

Schließlich muss noch die Konfigurationsdatei `ipsec.conf` angepasst werden. Hier ist die Zeile `presharedkey` durch eine Zeile `rightca` zu ersetzen (siehe auch Listing 7.1).

```
rightca="C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
OU=Wireless-VPN, CN=RootCA, Email=ralf@spenneberg.net"
```

Hierbei ist darauf zu achten, dass das Subjekt der CA und nicht des Zertifikates angegeben wird! Das Subjekt der CA kann mit dem folgenden Befehl ausgelesen werden.

```
# openssl x509 -in cacert.pem -noout -subject
```

Enthält das Subject der CA eine E-Mail-Adresse, so ist, je nach verwendetem Betriebssystem, in dem Subject das vorangestellte `Email=` durch `E=` zu erset-

zen. Diese Ersetzung ist aber nicht immer erforderlich. Dies muss je nach verwendetem System getestet werden.

Sobald die Konfiguration unter Linux und Windows angepasst wurde, kann wieder der Befehl `ipsec.exe` gestartet werden um die Richtlinie zu laden. Anschließend steht der Tunnel wieder bei Bedarf zur Verfügung.

7.5 Checkpoint Firewall-1 NG

Die Firma Checkpoint (*http://www.checkpoint.com*) hat mit den Produkten Firewall-1 und VPN-1 auch VPN Lösungen im Programm. Diese kommerziellen Lösungen werden trotz ihres hohen Preises in vielen Umgebungen eingesetzt. Häufig sind die guten Schulungsmöglichkeiten auch durch Drittanbieter hierfür ein Grund. Einige Anwender sehen in Checkpoint auch den Marktführer im Segment Firewall Software. Dieser Abschnitt beschreibt die wesentlichen Einstellungen und Konfigurationen, damit ein Linux Client eine VPN Verbindung zur Checkpoint Firewall-1 NG aufbauen kann.

Die Checkpoint Firewall-1 NG unterstützt die folgenden Algorithmen für die Verschlüsselung der IKE-Verhandlungen: DES, 3DES, CAST-128 und AES-256. Die Authentizität kann mit MD5 und SHA-1 überprüft werden. Es werden die Diffie Hellmann Gruppen 1, 2 und 5 unterstützt. Die Datenverschlüsselung kann mit DES, 3DES, CAST-128 und AES-{128,256} erfolgen. Für die Authentifizierung kann die Checkpoint FW-1 Preshared Keys als auch RSA-Signaturen (X.509 Zertifikate) nutzen.

Bei der Konfiguration des IPsec Tunnels sind einige weitere Hinweise zu beachten. Wenn die Checkpoint Firewall ein X.509 Zertifikat für die Authentifizierung benutzt, dann verwendet sie als eigene ID ihre IP Adresse, obwohl das Zertifikat ein anderes Subjekt trägt.

Teilweise entstehen Fehler, wenn die Checkpoint FW-1 die IPsec SAs neu aushandeln möchte. Sinnvollerweise wählt man hier auf der Linux Seite eine kürzere Lebensdauer. Dann beginnt der Linux IKE Daemon das Rekeying.

Am einfachsten ist die Anwendung mit einem Preshared Key. Hier ist jedoch zu beachten, dass die Checkpoint FW-1 den Aggressive Modus beherrscht. Um den Preshared Key auf der Checkpoint FW-1 einzutragen sind zunächst die Netzwerke zu konfigurieren. Anschließend kann in den Eigenschaften des VPN Moduls das IKE Protokoll ausgewählt und editiert werden. Dort werden die anzuwendenden Algorithmen und der Preshared Key aktiviert und das Kennwort eingegeben. Die restliche Konfiguration des FW-1 entspricht der üblichen VPN Konfiguration.

Wenn X.509 Zertifikate verwendet werden sollen, so besteht das erste Problem darin, dass dem Autor keine Methode bekannt ist, wie ein auf FW-1 erzeugtes X.509 Zertifikat für externe Systeme exportiert werden kann. Daher ist es erforderlich, eine zweite CA zu erzeugen, die anschließend in der FW-1 als OPSEC CA importiert wird. Dieser Vorgang wird sehr gut auf den Seiten von AERASEC (*http://www.fw-1.de/aerasec/ng/vpn-freeswan/CPNG+Linux-FreeSWAN.html*) mit Unterstützung einiger Screenshots erläutert.

Das auf der Seite von AERASEC beschriebene Verfahren lässt sich erfolgreich auf FreeS/WAN, `racoon` und `isakmpd` anwenden.

7.6 Cisco

Cisco ist wie FreeS/WAN und `isakmpd` einer der Teilnehmer des IKE-Interoperabilitätstestes, der am 5. November 2001 auf der IPsec 2001 Konferenz in Frankreich durchgeführt wurde (*http://www.hsc.fr/ressources/ipsec/ipsec2001/#results*). Dort wurde die Interoperabilität unter Beweis gestellt. Lediglich der `isakmpd` benötigte für die Verwendung von X.509 Zertifikaten eine lokale Kopie dieser Zertifikate.

Zusätzlich wurden die folgenden Probleme erkannt:

- Cisco IOS akzeptiert keine längere Lebensdauer, als zuvor im Cisco IOS konfiguriert wurde. Sinnvollerweise verwenden beide Systeme die identische Lebensdauer.
- Der Cisco VPN Concentrator 3000 verwendet eine T61String-Kodierung für den DN des Zertifikates. Dies führt zu Problemen, die bei FreeS/WAN durch eine binäre Kodierung des DN in der Konfigurationsdatei gelöst werden können. Das Kommando `fswcert` (*http://www.strongsec.com*) kann verwendet werden, um die binäre Kodierung zu extrahieren.
- Cisco IOS ist nicht in der Lage mit 4096 Bit langen Schlüsseln zu arbeiten.
- Wenn `isakmpd` eingesetzt wird, sollte lediglich der 3DES-Algorithmus zur Verschlüsselung angeboten werden. Wenn `isakmpd` zwei verschiedene Proposals anbietet, werden unter Umständen von beiden Seiten unterschiedliche Proposals gewählt.

Wenn diese Einschränkungen beachtet werden, ist es recht einfach eine VPN Verbindung aufzubauen. Dabei besteht die Möglichkeit die Authentifizierung sowohl mit Preshared Keys als auch mit X.509 Zertifikaten durchzuführen.

Ein typische Cisco IOS Konfiguration für die Authentifizierung mit einer PSK zeigt das folgende Listing.

```
crypto map VPN 30 ipsec-isakmp
 set peer 3.0.0.1
 set transform-set 3des-md5
 match address 130

crypto ipsec transform-set 3des-md5 esp-3des esp-md5-hmac

crypto isakmp key <passphrase> address 3.0.0.1


crypto isakmp policy 3
 encr 3des
 hash md5
 authentication pre-share
 group 2

access-list 130 permit ip 10.0.2.0 0.0.0.255 10.0.1.0 0.0.0.255
```

8 Aufbau einer Public Key Infrastruktur

Eine Public Key Infrastruktur (PKI) ermöglicht den sicheren Austausch von digitalen Signaturen, verschlüsselten Dokumenten, Authentifizierung, Autorisierung und vielen weiteren Funktionen, auch wenn mehrere Kommunikationspartner betroffen sind.

Dieses Kapitel stellt einige Werkzeuge vor, die Teile einer PKI oder auch eine ganze PKI implementieren. Zu Beginn werden zwei einfachere Werkzeuge vorgestellt, die grafische Certificate Authorities implementieren. Im letzten Abschnitt wird die Installation der OpenCA beschrieben. Hierbei handelt es sich um eine komplette PKI.

8.1 Einleitung

Eine Public-Key Infrastruktur erlaubt die zentrale Erzeugung, Signatur, Speicherung, Verwaltung und Verteilung von X.509 Zertifikaten und den privaten Schlüsseln. Eine PKI besteht aus drei Teilen:

- Certificate Authority (CA)
- Registration Authority (RA)
- Directory Service

8.1.1 Certificate Authority

Die Certificate Authority (CA) ist verantwortlich für die Ausgabe und Verwaltung der digitalen Zertifikate. Die CA übernimmt hierfür die folgenden Aufgaben:

- Zertifizierung des öffentlichen Schlüssels des Benutzers
- Veröffentlichung des Zertifikates
- Ausgabe von Zertifikatswiderruflisten (Certification Revocation Lists, CRLs)

8.1.2 Registration Authority

Die Registration Authority (RA) ist verantwortlich für die Aufzeichnung und Überprüfung aller Informationen, die die CA benötigt. Hierbei ist es besonders wichtig die Identität des Benutzers zu überprüfen, der die Unterzeichnung des Zertifikates beantragt. Dies kann selten durch einen Netzwerkdienst oder online erfolgen. Üblicherweise erfolgt die Identitätsüberprüfung durch eine Kontrolle des Ausweises.

8.1.3 Directory Service

Der Verzeichnisdienst (Directory Service) hat die Aufgabe die gültigen Zertifikate aller Benutzer in einem zentralen Repositorium zu speichern und zu veröffentlichen, so dass jeder Benutzer auf die Zertifikate der anderen Benutzer zugreifen kann. Zusätzlich veröffentlicht er die Zertifikatswiderruflisten (CRLs) oder erlaubt eine Überprüfung der Zertifikate mit dem Online Certificate Status Protocol (OCSP).

8.2 TinyCA

Die Perl-Anwendung TinyCA (*http://tinyca.sm-zone.net/*) von Stephan Martin ist eine einfache grafische Benutzeroberfläche für die Verwaltung einer CA. TinyCA ist in Perl/Gtk geschrieben und greift auf die Funktionen des OpenSSL-Paketes zurück.

TinyCA unterstützt fast sämtliche Operationen, die bei der Verwaltung einer CA auftreten können:

- Es sind eine unbegrenzte Anzahl von CAs möglich.
- X.509 Zertifikate können erzeugt und zurückgerufen werden.
- PKCS#10 Zertifikatsanfragen können importiert und signiert werden.
- Zertifikate können in den Formaten PEM, DER, TXT und PKCS#12 exportiert werden.
- Zertifikatsrückruflisten können in den Formaten PEM, DER und TXT exportiert werden.
- TinyCA bietet Unterstützung für Deutsch und Englisch.

8.2.1 Installation

Die Installation von TinyCA ist sehr einfach, da es sich um ein fertiges Perl Programm handelt. Auf der Homepage von TinyCA sind sowohl RPM als auch Debian Pakete neben dem Quelltextarchiv verfügbar. In Abhängigkeit der Distribution sind jedoch verschiedene Perl Pakete zusätzlich erforderlich, damit TinyCA gestartet werden kann. Der Autor hat unter *http://www.spenneberg.org/Certificate_Authorities/TinyCA* die Pakete, die unter Red Hat Linux 9 erforderlich sind, zusammengestellt.

8.2.2 Aufbau einer CA mit TinyCA

Der Aufbau einer CA ist mit dem Programm TinyCA sehr einfach. Direkt nach dem Start des Programms mit dem Befehl `tinyca` werden zunächst die Informationen der neu zu erzeugenden CA abgefragt (Abbildung 8.1). Wird die deutsche Sprache gewünscht, so muss vorher folgender Befehl die Umgebungsvariable setzen: `export LC_ALL=de_DE`.

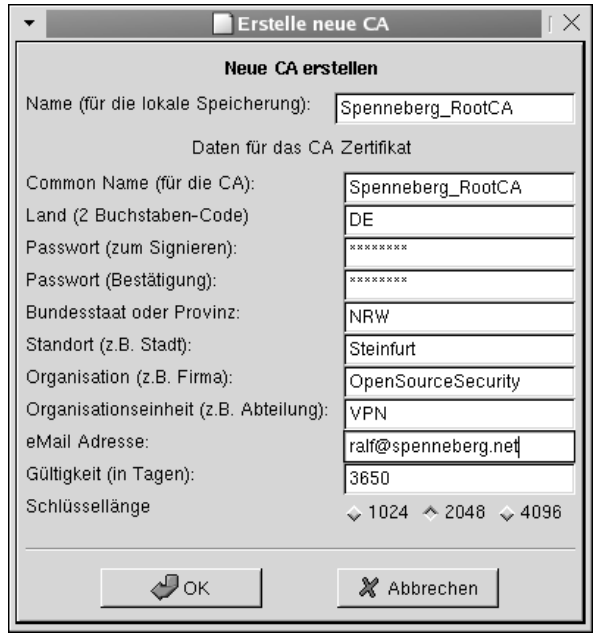


Abbildung 8.1 Erzeugung einer neuen CA

Anschließend werden weitere Einstellungen der CA abgefragt (Abbildung 8.2). Hier ist es insbesondere möglich, den `crlDistributionPoint` anzugeben. FreeS/WAN mit dem X.509 Patch ab Version 1.1.0 unterstützt den auto-

matischen Download der CRLs von den angegebenen Stellen. Hier können mehrere Distributionspunkte durch Semikolon getrennt angegeben werden. Der X.509 Patch unterstützt den Download mit den Protokollen `ldap://`, `http://`, `ftp://` und `file://`. Hierbei muss zusätzlich die Angabe `URI:` vorangestellt werden.

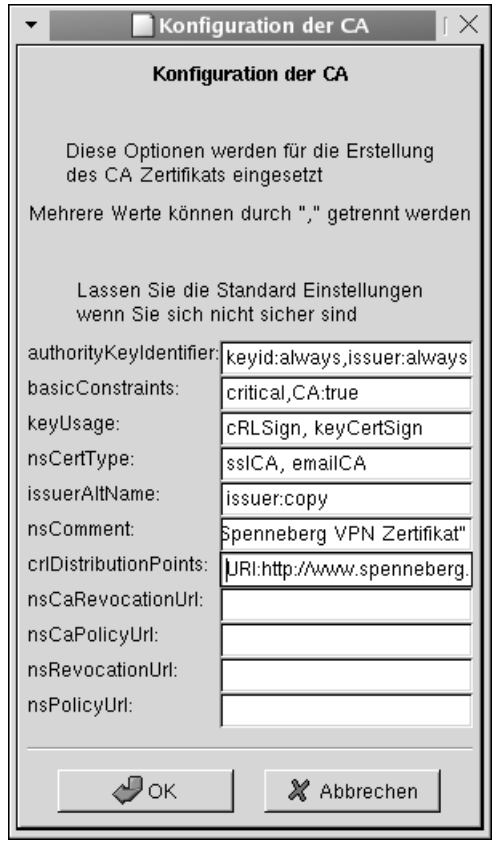


Abbildung 8.2 Weitere Eigenschaften der CA

Nun kann TinyCA für die Erzeugung neuer Zertifikate verwendet werden. Die grafische Oberfläche bietet nun Buttons und Registerkarten, die die Erzeugung und Verwaltung von Zertifikaten ermöglichen (Abbildung 8.3).

Um nun einen neuen Schlüssel und ein Zertifikat zu erzeugen wechselt der Benutzer zunächst auf die Registerkarte ANFORDERUNGEN. Dort wählt er den Button NEU. Im anschließenden Popup-Fenster kann er die Angaben für die Zertifikatsanforderung eingeben (Abbildung 8.4).

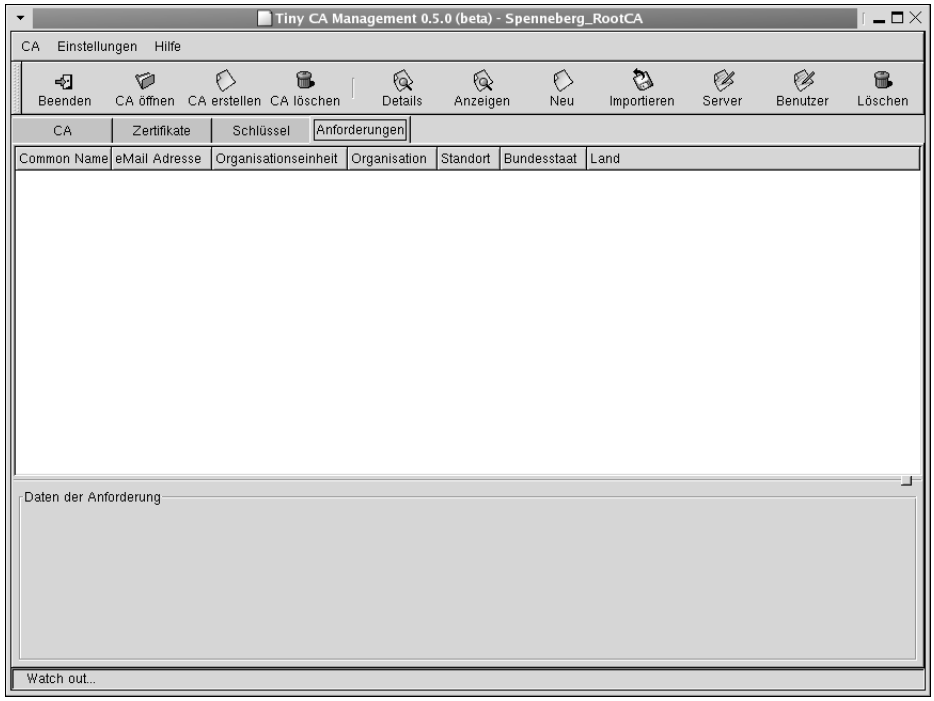


Abbildung 8.3 TinyCA ist nun einsatzbereit

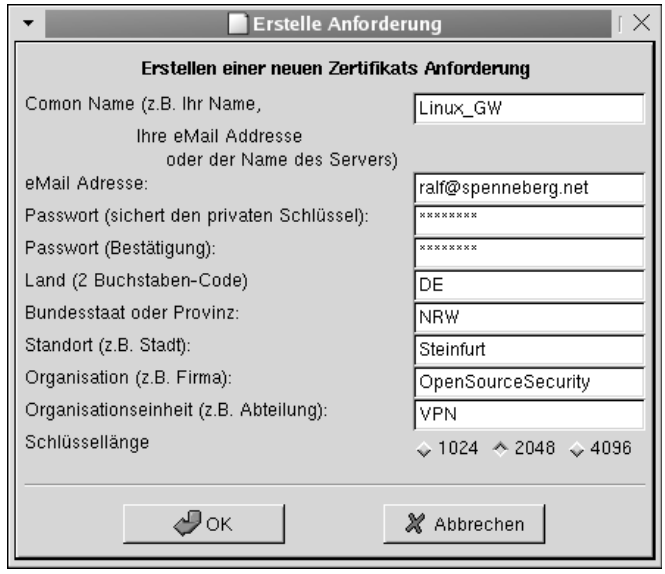


Abbildung 8.4 Erstellen einer neuen Anforderung

Nach Bestätigung der Anforderung mit OK erscheint diese in der Liste der grafischen Oberfläche. Die Buttons DETAILS und ANZEIGEN geben weitere Informationen über das Zertifikat.

Nun kann das Zertifikat signiert werden. TinyCA unterstützt hier Server und Benutzerzertifikate. Durch Auswahl des Buttons SERVER wird die Anfrage signiert. Dabei muss das Kennwort der CA eingegeben und die Gültigkeitsdauer bestimmt werden (Abbildung 8.5).

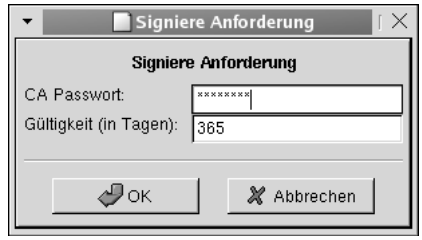


Abbildung 8.5 Unterzeichnen der Anforderung

Um ein Zertifikat zu widerrufen, kann auf der Registerkarte ZERTIFIKATE der Button WIDERRUFEN ausgewählt werden. Er fordert das Kennwort der CA an und widerruft das Zertifikat.

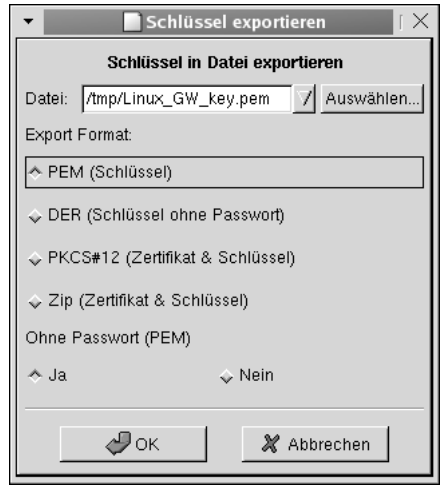


Abbildung 8.6 Export des Schlüssels

Damit der Schlüssel und die Zertifikate nun für den Aufbau von VPNs genutzt werden können, müssen sie exportiert werden. Zunächst soll der Schlüssel exportiert werden. Dazu ist auf der Registerkarte SCHLÜSSEL der

Button EXPORTIEREN auszuwählen. Im anschließenden Popup-Fenster können die Datei und das Format des Schlüssels angegeben werden (Abbildung 8.6).

Dabei ist es wichtig, dass der X.509 Patch für FreeS/WAN mit kodierten Schlüsseln umgehen kann. Sowohl `racoon` als auch `isakmpd` können dies nicht. Daher darf der Schlüssel hier nicht mit einem Kennwort geschützt werden (JA anwählen).

Nun kann das Zertifikat des Clients und das Zertifikat der CA exportiert werden. Hierfür ist auf den Registerkarten ZERTIFIKATE beziehungsweise CA der Button EXPORTIEREN beziehungsweise CA EXPORTIEREN auszuwählen (Abbildung 8.7).

Abbildung 8.7 Export des Zertifikates

Wenn als Client ein Microsoft Windows-Betriebssystem eingesetzt wird, so empfiehlt es sich den Schlüssel im PKCS#12 Format zu exportieren. Dies wird von TinyCA unterstützt. Dazu wird auf der Registerkarte ZERTIFIKATE das entsprechende Zertifikat ausgewählt. Nach der Auswahl von EXPORTIEREN wird das Format PKCS#12 benutzt (Abbildung 8.8).

Da dieses Format sowohl das Zertifikat als auch den privaten Schlüssel enthält, muss das Kennwort des Schlüssels eingegeben werden. Um den Schlüssel anschließend wieder zu schützen, kann ein Export Kennwort definiert werden. Sinnvoll ist es auch das Zertifikat der CA zur Datei hinzuzufügen, da der Windows Client dann sämtliche benötigten Informationen besitzt (Abbildung 8.9).

Abbildung 8.8 Auswahl des PKCS#12 Formates für Windows Clients

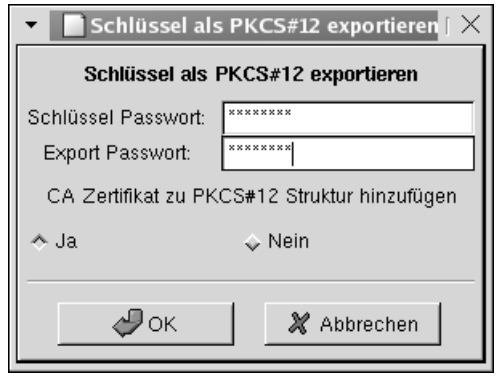


Abbildung 8.9 Hinzufügen des CA Zertifikates zur PKCS#12 Datei

Damit bietet die TinyCA alle Funktionen für die Erzeugung und Verwaltung von Zertifikaten für ein VPN. Leider bietet TinyCA keine Unterstützung der Zertifikatsseriennummer und keine direkte Veröffentlichung der Zertifikate oder der Zertifikatswiderruflisten auf einem Webserver oder LDAP-Server. Die automatische Veröffentlichung ist jedoch häufig auch mit Fehlern verbunden, ihr Fehlen kann daher verschmerzt werden.

Die TinyCA speichert alle Informationen im Verzeichnis `~/.TinyCA`. Dieses Verzeichnis sollte sinnvollerweise besonders geschützt werden. Möglicherweise sollte es sich um eine Verknüpfung auf eine Diskette handeln, die nach Bedarf entfernt und physikalisch sicher gelagert werden kann.

8.3 XCA

Die Anwendung XCA (*http://www.hohnstaedt.de/xca.html*) von Christian Hohnstädt erlaubt wie TinyCA die grafische Erzeugung einer CA, der Schlüssel und der Zertifikate. Dabei können die Zertifikate in den Formaten PEM und DER abgespeichert werden. Zusätzlich wird der Export von PKCS#12 und der Import von PKCS#7, PKCS#10 und PKCS#12 unterstützt. Die X.509.v3 Erweiterungen können genutzt werden. Zusätzlich unterstützt XCA die Verwendung der Seriennummern in den Zertifikaten. Dies ist wichtig, wenn nach Ablauf eines Zertifikates ein neues erzeugt werden soll. Die Erzeugung von Widerruflisten wird unterstützt und es können auch externe CRLs betrachtet und importiert werden.

8.3.1 Installation

Die Installation ist recht einfach. Auf der Sourceforge Projekt Seite *http://sourceforge.net/projects/xca* sind das Quelltextarchiv, Debian und Windows Programmpakete verfügbar. Bei der Übersetzung ist zu beachten, dass die folgenden Programmpakete erforderlich sind: QT-2.2.4 oder höher, BerkeleyDB 3.2 oder höher und OpenSSL-0.9.6 oder höher. Dann erfolgt die Übersetzung mit:

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

Bei der Übersetzung auf Red Hat Linux Systemen tritt jedoch ein Problem auf, da dort die beiden Include-Dateien `cxx_common.h` und `cxx_except.h` in den db[3|4] devel Paketen fehlen. Sie werden von Christian Hohnstädt aber unter *http://www.hohnstaedt.de/xca/* zur Verfügung gestellt. Ich pflege unter *http://www.spenneberg.org/* ein RPM-Paket für Red Hat Linux 9, welches eine einfache Installation ermöglicht.

8.3.2 Anwendung von XCA

Nach der Installation kann die Anwendung mit dem Befehl `xca` aufgerufen werden. Beim ersten Aufruf fordert die Anwendung den Benutzer auf, ein Kennwort für den Zugriff auf die Daten zu wählen und zweimal einzugeben (Abbildung 8.10).

Abbildung 8.10 XCA fordert das Kennwort für den Zugriff auf die Daten

Nun bietet das Werkzeug eine grafische Oberfläche mit mehreren Registerkarten (RSA Schlüssel, Unterschriftsanfragen, Zeritifikate, Vorlagen und Rücknahmelisten) und mehren Buttons (NEUER SCHLÜSSEL, EXPORT, IMPORT, IMPORT PFX (PKCS#12), DETAILS ANZEIGEN, LÖSCHEN und PASSWORT ÄNDERN). Abbildung 8.11 zeigt die grafische Oberfläche.

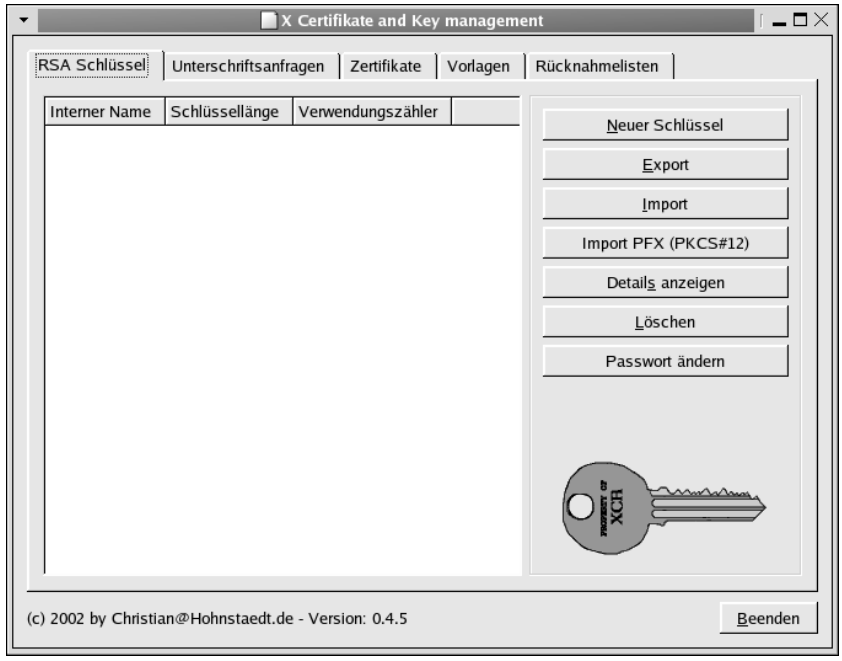


Abbildung 8.11 Die grafische XCA-Oberfläche

Um nun eine neue CA zu erzeugen, wird auf der Registerkarte ZERTIFIKATE der Button NEUES ZERTIFIKAT angewählt. Anschließend startet ein Wizard. Nach Bestätigung des Buttons NEXT können weitere Informationen über das Zertifikat eingegeben werden (Abbildung 8.12). Hier ist nun die Wahl der CA VORLAGE erforderlich. XCA verwendet Vorlagen beziehungsweise Templates, die im Vorfeld über die entsprechende Registerkarte angepasst werden können.

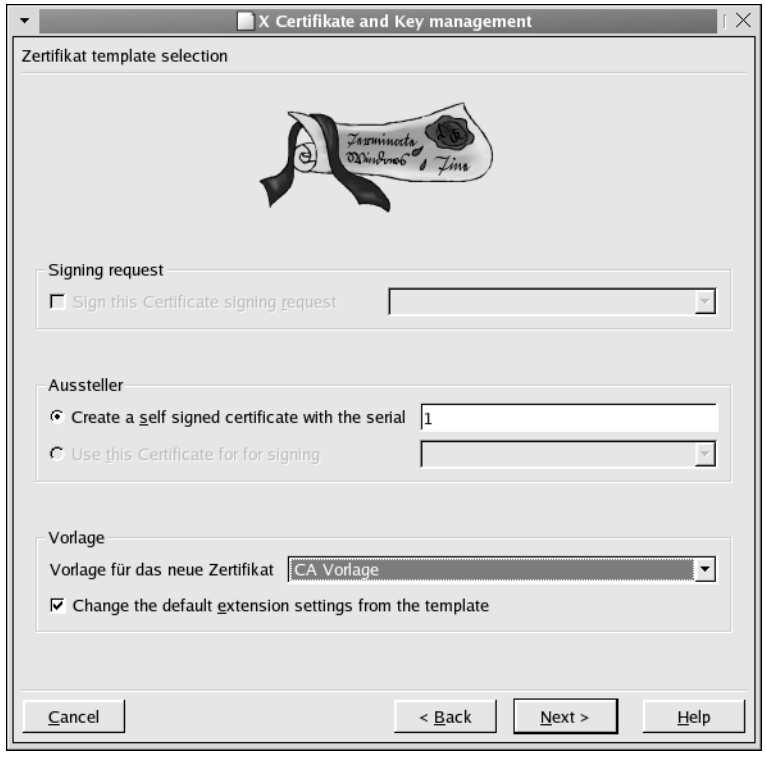


Abbildung 8.12 Auswahl eines CA Zertifikates

Nach Wahl von NEXT muss der Benutzer einen Namen für den Schlüssel eingeben und die Schlüssellänge bestimmen (Abbildung 8.13).

Nun muss der Benutzer noch die weiteren für das Zertifikat der CA erforderlichen Informationen eingeben. Hierzu gehören ein interner Name, Country Code, Country, Locality und so weiter (Abbildung 8.14).

Nun können weitere Eigenschaften der CA definiert werden. Hierbei handelt es sich um die Gültigkeitsdauer, den CRL Distribution Point und so weiter (Abbildung 8.15).

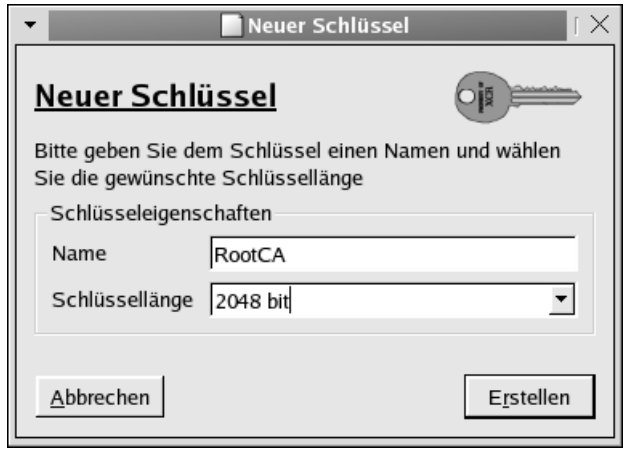


Abbildung 8.13 Wahl der Schlüssellänge

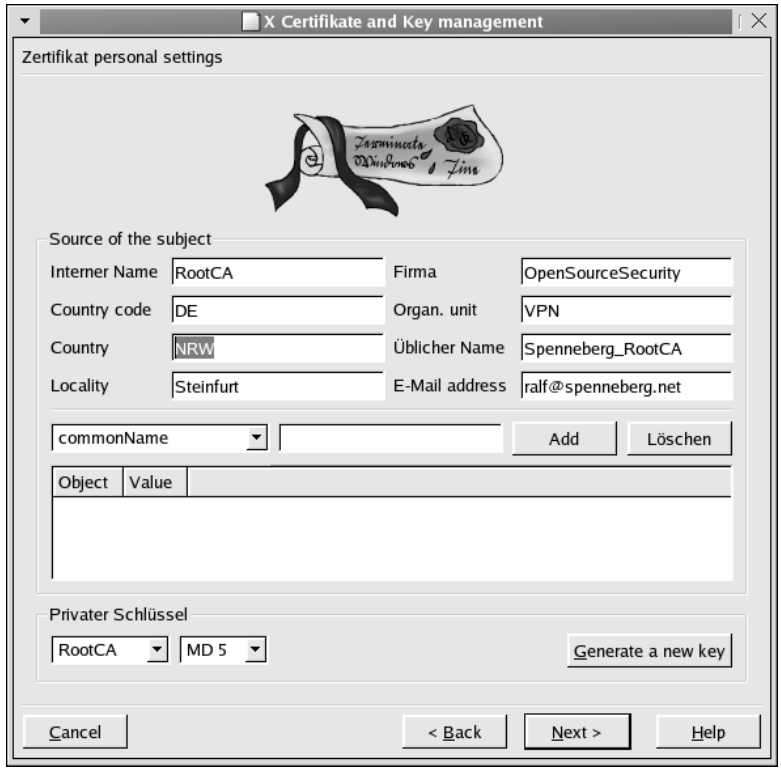


Abbildung 8.14 Beschreibung des CA Zertifikates

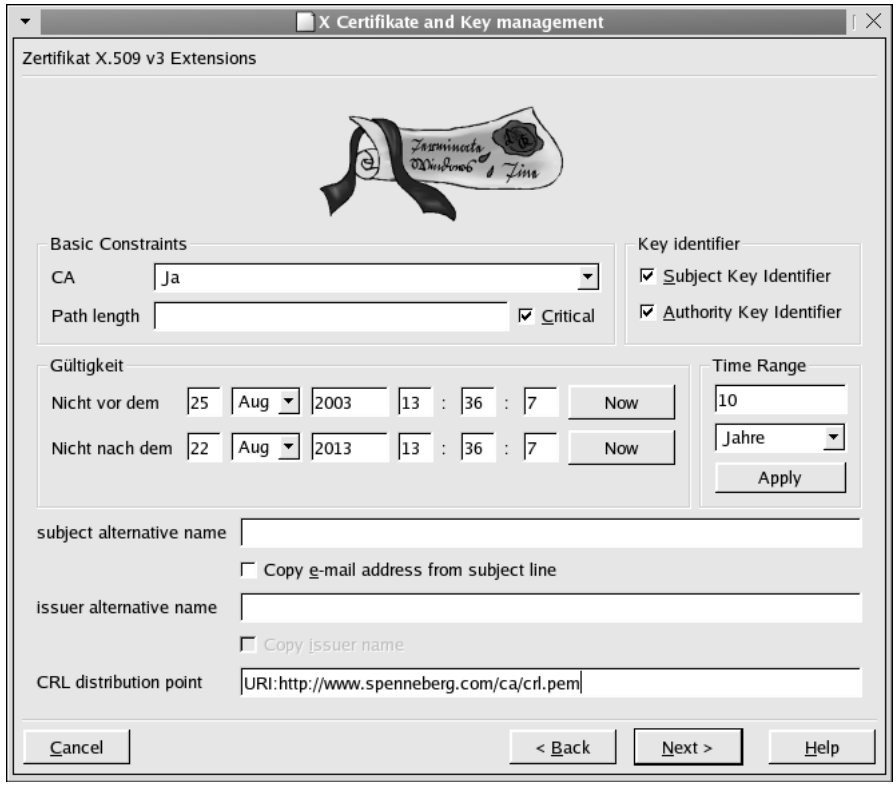


Abbildung 8.15 Wahl der Gültigkeitsdauer und der CRL-Liste

Anschließend kann der Benutzer die Rolle des Zertifikates einschränken. Damit kann das CA Zertifikat nur noch für diese Aufgaben genutzt werden. Eine Einschränkung kann die Sicherheit erhöhen, ist aber für den Einsatz als VPN-CA nicht zwingend erforderlich (Abbildung 8.16).

Im folgenden Fenster können Zertifikatsoptionen, die von Netscape definiert wurden und für den Einsatz im SSL Protokoll wichtig sein können, definiert werden. Dieser Dialog wird hier nicht gezeigt. Eine Auswahl von NEXT zeigt abschließend noch den Betreff des Zertifikates an und erlaubt mit Bestätigung des FINISH Buttons die Erzeugung der CA.

Nun kann das Zertifikat für einen VPN Client erzeugt werden. Hierzu wählt der Benutzer erneut den Button NEUES ZERTIFIKAT. Nach Bestätigung des ersten Fensters mit NEXT kann im zweiten Fenster die RootCA zur Signatur ausgewählt werden (Abbildung 8.17).

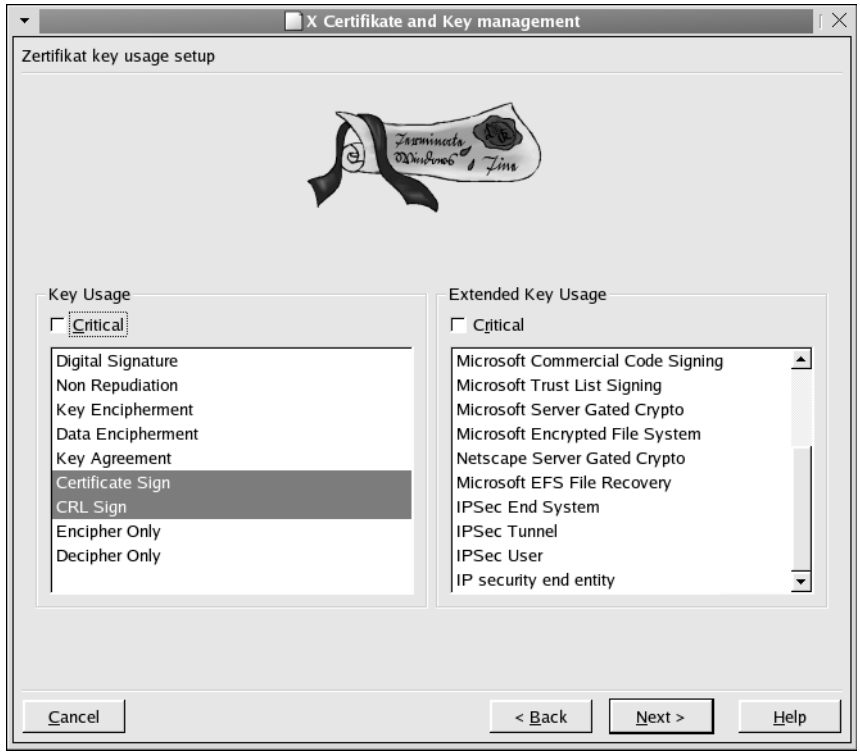


Abbildung 8.16 Einschränkung der Gültigkeit des Zertifikates

Nun wählt der Benutzer erneut (Abbildung 8.13) einen Namen für den Schlüssel (Linux_Client) und dessen Länge (2048 bit). Nach der Bestätigung mit ERSTELLEN können die Daten des Zertifikates wie die Gültigkeitsdauer eingegeben werden. Die Option SUBJECT ALTERNATIVE NAME (Abbildung 8.15) gibt die Möglichkeit einen alternativen Namen (`subjectAltName`) für das Zertifikat einzugeben. Der X.509 Patch kann diesen für den Zugriff auf das Zertifikat nutzen. `isakmpd` benötigt diesen Eintrag. Hier kann er bei der Erzeugung bereits gesetzt werden. Ansonsten enthält das `isakmpd` Paket den Befehl `certpatch` der ebenfalls diese X.509.v3 Erweiterung hinzufügen kann. Möglich sind hier die IP Adresse (`IP:217.160.128.61`) ein DNS Name (`DNS:www.nohup.info`) oder eine E-Mail-Adresse (`email:ralf@spenneberg.net`).

Nach Bestätigung mit NEXT kann der Gültigkeitsbereich des Zertifikates wieder eingeschränkt werden (Abbildung 8.16). Auch die Netscape Erweiterungen können für den Einsatz als VPN Zertifikat wieder mit NEXT übersprungen werden. Schließlich wird das Zertifikat mit FINISH erzeugt und die grafische Oberfläche stellt die beiden Zertifikate dar (Abbildung 8.18).

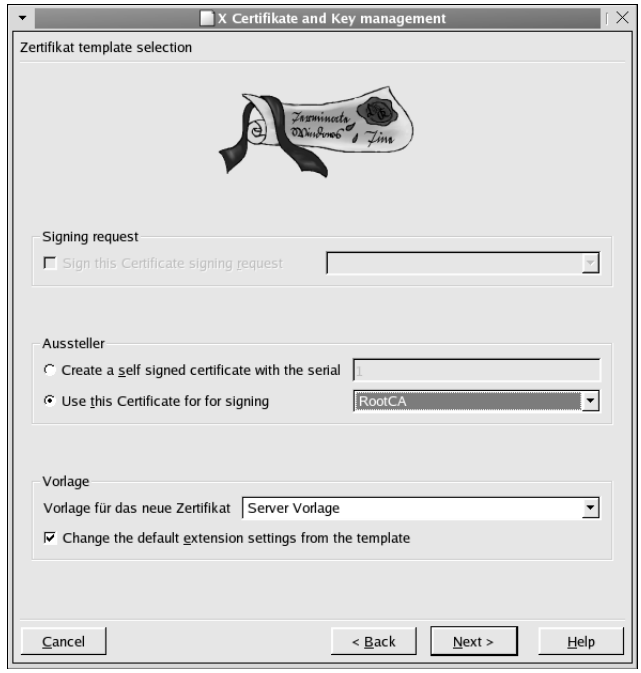


Abbildung 8.17 Auswahl der RootCA zur Signatur

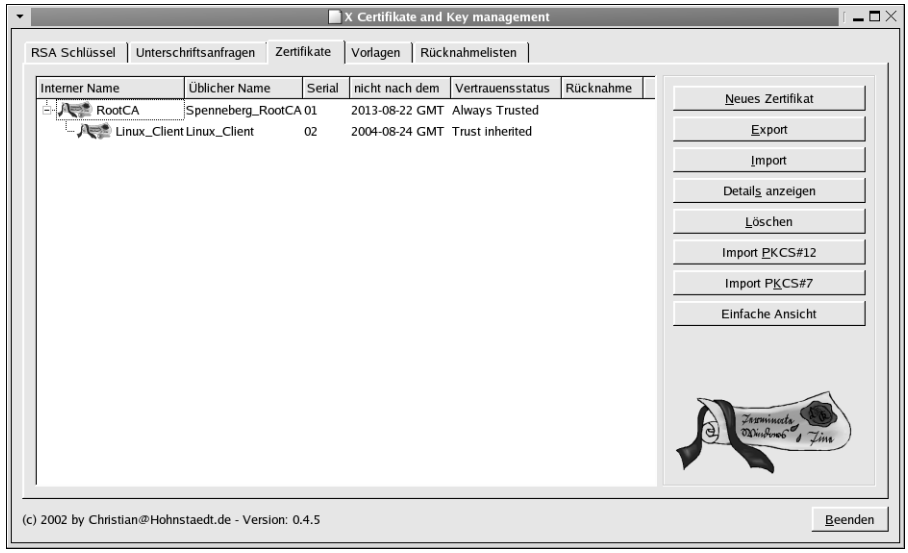


Abbildung 8.18 XCA mit einer CA und einem Zertifikat

In der Baumansicht ist die Abhängigkeit der Zertifikate voneinander sehr einfach nachvollziehbar. XCA kann auch Zertifikatsautoritäten erzeugen, die nicht selbst RootCA sind. Der FreeS/WAN X.509 Patch ist in der Lage derartige Zertifikathierarchien über mehrere Ebenen bis zur RootCA zu verfolgen.

Um die Schlüssel und die Zertifikate nun zu verwenden müssen diese exportiert werden. Hierzu wechselt man zunächst auf die Registerkarte RSA SCHLÜSSEL. Nach der Wahl von EXPORT kann die Datei für den Export angegeben werden (Abbildung 8.19). Hier ist wichtig, dass für die Verwendung durch den `racoon` oder den `isakmpd` IKE Daemon der Schlüssel nicht mit einem Kennwort geschützt wird.

Abbildung 8.19 Export des privaten RSA-Schlüssels im PEM-Format

Anschließend können die Zertifikate exportiert werden. Hierfür wird auf der Registerkarte ZERTIFIKATE das zu exportierende Zertifikat ausgewählt und mit EXPORT exportiert. Wenn es sich um ein Zertifikat für einen Windows Client handelt, empfiehlt es sich statt des PEM-Formates das PKCS #12 WITH CERTIFICATE CHAIN Format zu wählen (Abbildung 8.20). Dann wird auch der private RSA-Schlüssel und das Zertifikat der CA mit in die Datei aufgenommen.

Abbildung 8.20 Export des Windowszertifikates im PKCS#12-Format

Für den Schutz des privaten Schlüssels in der `.p12` Datei wird im nächsten Fenster bei Wahl des PKCS#12 Formates ein Export Kennwort angefragt. Bei einem Export im PEM Format ist dies nicht erforderlich.

8.3.3 Migration einer CA zum XCA-Werkzeug

Eine Besonderheit des XCA-Werkzeuges ist die Möglichkeit eine vorhandene CA, die zum Beispiel mit dem OpenSSL-Befehl `/usr/share/ssl/misc/CA` erzeugt wurde, in die XCA zu migrieren. Hierzu bietet XCA die Möglichkeit sowohl RSA Schlüssel als auch Zertifikate zu importieren.

So ist es möglich von einer kommandozeilen basierten CA auf eine bequem grafisch mit dem XCA Werkzeug zu administrierende CA umzusteigen.

8.4 OpenCA

Die OpenCA ist ein Open Source Projekt, das 1999 gestartet wurde, um eine Open Source Trust Center Software für UNIX zu erschaffen. Hauptsächlich besteht die OpenCA aus einem Web Interface, welches in Perl geschrieben wurde, OpenSSL für die kryptografischen Operationen und einer Datenbank, in der die Schlüssel, Zertifikate und Widerrufslisten gespeichert werden.

Zum heutigen Zeitpunkt unterstützt die OpenCA bereits die folgenden Funktionen:

- Verschiedene Schnittstellen: LDAP, RA, CA und die Protokolle SCEP und OCSP.
- Anmeldung mit Kennwort und Zertifikat (Zertifikate können auf einer Smartcard hinterlegt werden.).
- Flexible Steuerung der Certificate Subjects und der X.509.v3 Erweiterungen.
- Rückruf der Zertifikate gesteuert durch PIN Nummern oder mit digitalen Signaturen.
- Warnung bei Ablauf eines Zertifikates.
- Viele weitere Optionen.

Die Entwicklung der OpenCA ist aber noch nicht abgeschlossen. Im Moment wird fieberhaft an der Fertigstellung von Version 0.92 gearbeitet. Sie existiert im Moment nur im CVS und in Form von regelmäßigen Snapshots. Das Entwicklerteam erwartet die Fertigstellung bis Ende 2003.

Die Installation und Konfiguration der OpenCA könnte jedoch ein ganzes Buch füllen und würde dieses hier sprengen. Ich möchte daher auf die entsprechende Literatur im Internet und die Mailingliste verweisen. Auf der Homepage der OpenCA (*http://www.openca.org*) gibt es mehrere Dokumente, die die Installation und Konfiguration beschreiben. Sie sind auf der Seite *http://www.openca.org/openca/docs/* zu finden.

Ein weiteres Dokument, das die Installation einer älteren Version der OpenCA beschreibt, ist das Open-Source-PKI-Book. Sie befindet sich auf *http://ospkibook.sourceforge.net/*.

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.[1] 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 `adddns.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
      leftrsasigkey=0s....
      right=vpngateway.dyndns.org
      rightid=@vpnserver
```

```
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
spdflush;
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 priotisieren 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 kerneldir -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 priotisieren kann mit dem `tcc` 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 Standardtunnel 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.

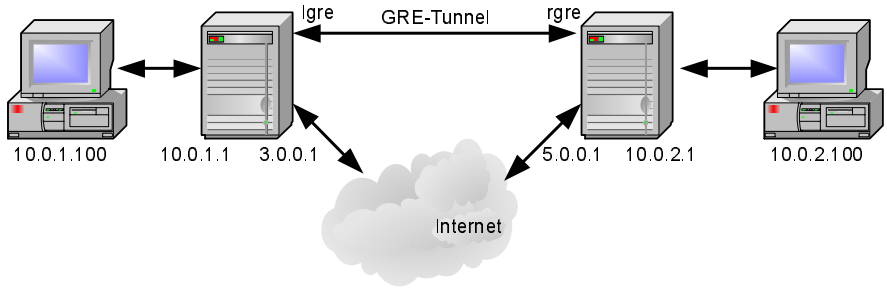


Abbildung 9.1 Ein GRE Tunnel zwischen dem linken und dem rechten Netzwerk

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.

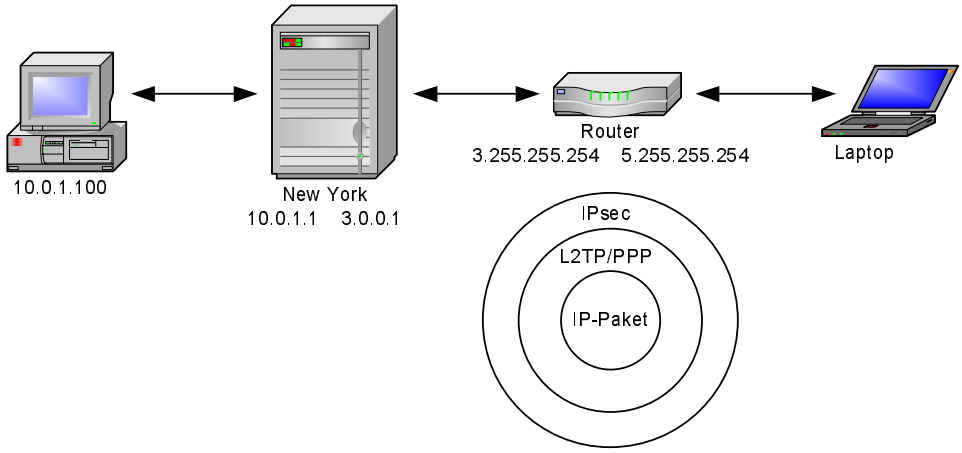


Abbildung 9.2 L2TP Verbindung mit einem Windows Client

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* 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 `mru 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 Traversals 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 Traversals

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`.[2]

```
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.
- **%all** 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.

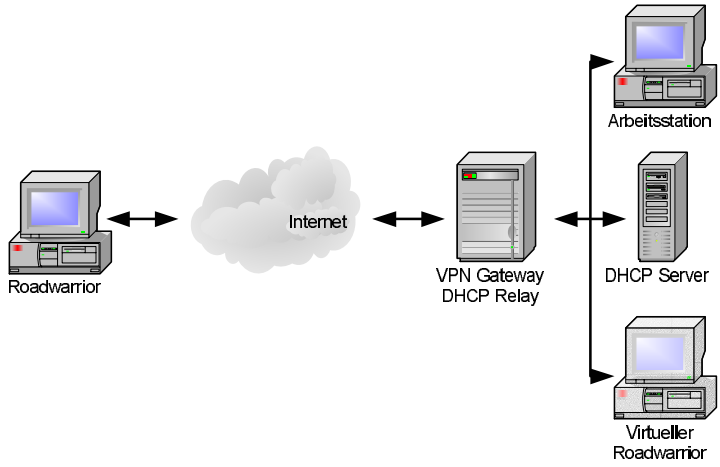


Abbildung 9.4 Der Client fordert zunächst per DHCP eine IP Adresse an, um später mit dieser Adresse über den Tunnel zu kommunizieren

ACHTUNG

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

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/dhcprelay/index.htm* zur Verfügung gestellt. Unter dieser URL findet der Leser auch ein DHCP Relay Howto, welches die Installation und Konfiguration beschreibt.

Die Software `dhcprelay-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>/dhcprelay-<version>.tar.gz
# cd dhcprelay-<version>
# ./configure
# make
# make install
```

Anschließend kann das DHCP Relay mit dem Skript `/etc/init.d/dhcprelay` 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/dhcprelay.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
DHCPSERVER="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 (`DHCPSERVER`) 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 Roadwarriorkonfiguration 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.

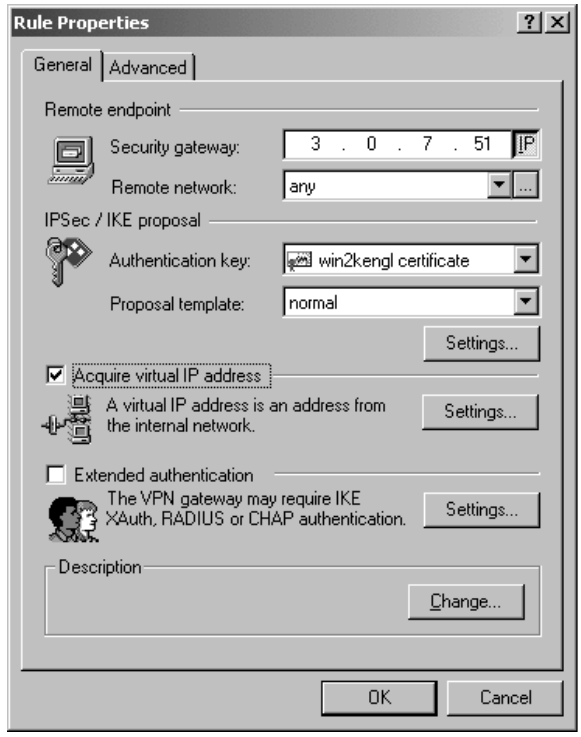


Abbildung 9.5 Aktivierung der virtuellen Adresse im SSH Sentinel

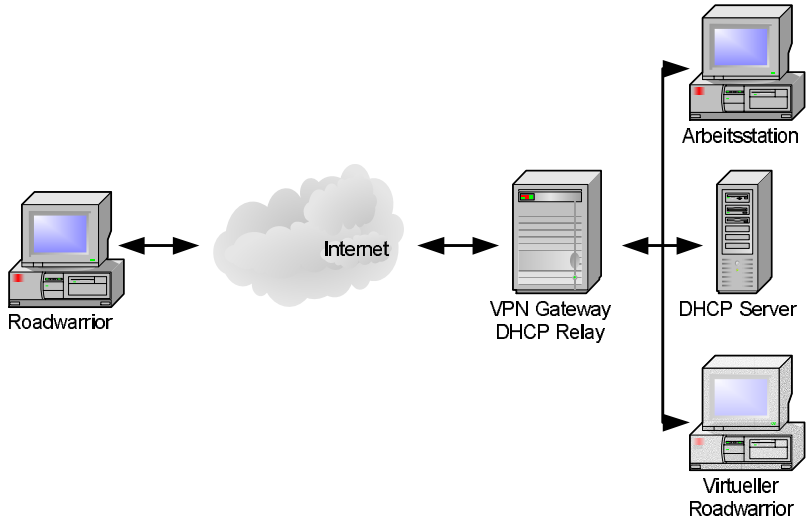


Abbildung 9.6 Auswahl des DHCP-over-IPsec Protokolls

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+l7MZzbAHH+lBllFX8T1K0Bsl
```

```
jBRRFldd85g93eUEXARuJOi2LLo1X5JwGabLYtju+fhyks358rvuzmDX+V/PUDgvnWq96COWU
BM71l9wv1gbtOvoHI2eDyUYZmhh7N3uKAfArmEnAIGdJxe5/FEF9h6ZhDn0a7bV3C/0571rCrn
OhV9c56LXGDIAnwS7g1QY/zPkzG4o+UVJcF/" "PwsPCJSY+m3iqfLXJXSkEvBL5+m9kPQLmy
CA7ezIvnPVp+x6ptQ7V1f50DbJQvmZ9uSS6/Bbq4nEMd5/U5d/crrbni+OkZel2UGtcFvZtV
vmyoj+lJ"
```

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+l7MZzbAHH+lBllFX8T1K0BsljBRRFldd85g93eUEXARuJOi2LLo1X
5JwGabLYtju+fhyks358rvuzmDX+V/PUDgvnWq96COWUBM71l9wv1gbtOvoHI2eDyUYZmhh7N3
uKAfArmEnAIGdJxe5/FEF9h6ZhDn0a7bV3C/0571rCrnOhV9c56LXGDIAnwS7g1QY/zPkzG4o+
UVJcF/" "PwsPCJSY+m3iqfLXJXSkEvBL5+m9kPQLmyCA7ezIvnPVp+x6ptQ7V1f50DbJQ
vmZ9uSS6/Bbq4nEMd5/U5d/crrbni+OkZel2UGtcFvZtVvmyoj+lJ"
```

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+l7MZzbAHH+lBllFX8T1K0BsljBR
RFldd85g93eUEXARuJOi2LLo1X5JwGabLYtju+fhyks358rvuzmDX+V/PUDgvnWq96COWUBM7
1l9wv1gbtOvoHI2eDyUYZmhh7N3uKAfArmEnAIGdJxe5/FEF9h6ZhDn0a7bV3C/0571rCrnOh
V9c56LXGDIAnwS7g1QY/zPkzG4o+UVJcF/" "PwsPCJSY+m3iqfLXJXSkEvBL5+m9kPQLmyCA7
ezIvnPVp+x6ptQ7V1f50DbJQvmZ9uSS6/Bbq4nEMd5/U5d/crrbni+OkZel2UGtcFvZtVvmyo
j+lJ"
```

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* 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 Gruppendateien `/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.kerneli.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 Austattung 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.[3]

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.

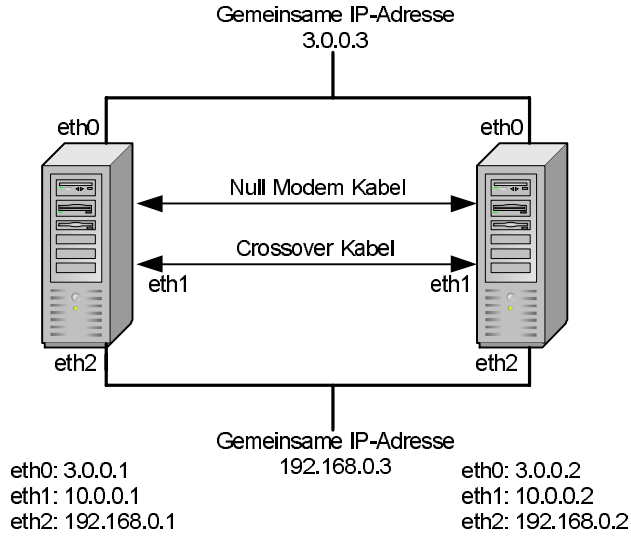


Abbildung 9.7 Der Heartbeat Cluster erlaubt den hochverfügbaren Aufbau von VPN Gateways

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 Betriebssytem 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 Betriebsystem 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
     rightnexthop=3.255.255.254
     rightid=@vpn.spenneberg.com
     rightrsasigkey=%cert
     left=5.0.0.1
     leftnexthop=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<lesegerätnr>:<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.

10 Fehlersuche

Dieses Kapitel widmet sich der lästigen aber häufig notwendigen Fehlersuche in Virtuellen Privaten Netzwerken. Hier gestaltet sich die Suche nach einem Fehler meist sehr schwierig, da die Nachrichten verschlüsselt übertragen werden. Bei einem Konfigurationsfehler verwirft möglicherweise eine Seite diese Pakete ohne Angabe eines Fehlers.

10.1 Werkzeuge

Es existieren eine Reihe von Werkzeugen, die bei der Fehlersuche helfen können. Das wichtigste Werkzeug ist sicherlich die Protokollierung durch die entsprechenden Systeme. Hier finden sich am ehesten die Fehlermeldungen und Hinweise auf Konfigurationsfehler. Sowohl FreeS/WAN als auch der `isakmpd` erlauben die Änderung des Debug-Levels im laufenden Betrieb.

10.1.1 FreeS/WAN Debugging

Bei FreeS/WAN kann mit den Befehl `ipsec whack` im laufenden Betrieb der Debug-Level verändert werden. Dabei kann mit der Option `–name` die Protokollierung nur für eine bestimmte Verbindung aktiviert werden. Folgende Protokollfunktionen existieren bei Pluto:

- **`–debug-raw`**
- **`–debug-crypt`**
- **`–debug-parsing`**
- **`–debug-emitting`**
- **`–debug-control`**
- **`–debug-klips`**
- **`–debug-dns`**
- **`–debug-all`** (nicht empfohlen)
- **`–debug-private`**
- **`–debug-none`** (schaltet das Debugging ab)

Für KLIPS existieren ähnliche Debug-Funktionen, die im laufenden Betrieb mit dem Befehl `ipsec klipsdebug` gesetzt werden können. Die Option `–all` schaltet das gesamte Debugging ein (nicht empfohlen). Mit der Option `–none`

kann es wieder abgeschaltet werden. Die Optionen `-set` und `-clear` erlauben das an- und abschalten einzelner Debug Funktionen.

- **`tunnel`** Tunnel Aktivitäten.
- **`tunnel-xmit`** Nur versendete Pakete in dem Tunnel.
- **`pfkey`** Kommunikation mit Pluto.
- **`xform`** Auswahl der Transform.
- **`eroute`** IPsec Routing Table.
- **`spi`** Änderungen in der SAD.
- **`esp`**
- **`ah`**
- **`ipcomp`**
- **`verbose`** Zusätzliche Informationen (nicht empfohlen).

Ein weiteres Feature bei der Fehlersuche in FreeS/WAN ist der Befehl `ipsec verify`. Dieser prüft die Konfigurationsdatei und die verwendeten Schlüssel auf ihre Syntax.

```
# ipsec verify
Checking your system to see if IPsec got installed and started correctly
Version check and ipsec on-path                                 [OK]
Checking for KLIPS support in kernel                            [FAILED]
Checking for RSA private key (/etc/ipsec.secrets)               ipsec
showhostkey: no pubkey line found -- key information old?       [FAILED]
Checking that pluto is running
whack: Pluto is not running (no "/var/run/pluto.ctl")           [FAILED]
DNS checks.
Looking for TXT in forward map: kermit.spenneberg.de            [MISSING]
Does the machine have at least one non-private address          [FAILED]
Two or more interfaces found, checking IP forwarding            [OK]
Checking NAT and MASQUERADING                                   [N/A]
```

Die Ausgabe muss jedoch mit Vorsicht gelesen werden, denn nicht jedes `MISSING` und jedes `FAILED` bedeutet, dass die Konfiguration Fehler aufweist. Dieser Befehl prüft auch ob sämtliche Schlüssel für die opportunistische Verschlüsselung (OE) im DNS eingetragen wurden. Wird die OE nicht verwendet, so können die entsprechenden Meldungen ignoriert werden. Teilweise kann dieser Befehl auch neue Parameter einzelner Patche nicht erkennen und richtig lesen. Dann kommt es ebenfalls zu Fehlermeldungen.

10.1.2 Debugging bei racoon

Bei `racoon` kann der Debug Level nur beim Start angegeben werden. Leider unterstützt `racoon` hier auch nur die Verwendung der allgemeinen Option `-d`, die den Level erhöht. Mehrere Angaben dieser Option erhöhen den Debug Level weiter. Ansonsten unterstützen `racoon` wie auch `isakmpd` die Ausführung im Vordergrund einer Konsole, so dass die Meldungen auf die Konsole geschrieben werden. Wird `racoon` nicht mit der Option `-F` angewiesen, im Vordergrund zu laufen, so protokolliert er über den `syslog`. Nach der Angabe der Option `-l logdatei` führt er die Protokollierung in der Datei durch. Wenn der Debug-Level geändert werden soll, muss `racoon` neu gestartet werden.

10.1.3 Debugging bei isakmpd

`isakmpd` unterstützt zusätzlich zur Angabe des Debug Levels bei seinem Start auch seine Konfiguration während der Ausführung. Hierfür erzeugt `isakmpd` bei seinem Start eine Named Pipe. Der Ort der Named Pipe kann mit der Option `-f` beim Start angegeben werden. Wird diese Option nicht verwendet, so verwendet `isakmpd` die Datei `/var/run/isakmpd.fifo`.

Durch das Senden des Befehls `D` an diese Named Pipe kann der Administrator den Debug Level konfigurieren. Im einzelnen stehen die folgenden Befehle zur Verfügung.

- **`D <class> <level>`** Dies stellt den Level für die angegebene Klasse (s.u.) ein.
- **`D T`** Hiermit wird das Debugging für alle Klassen abgeschaltet. Ein weiteres `D T` stellt das Debugging im alten Zustand wieder her.

`isakmpd` verwaltet die Debug-Informationen in Klassen. Hierbei definiert es die folgenden 11 Klassen: 0 (Misc), 1 (Transport), 2 (Message), 3 (Crypto), 4 (Timer), 5 (Sysdep), 6 (SA), 7 (Exchange), 8 (Negotiation), 9 (Policy) und A (Alle). Diesen Klassen können Werte zwischen 0 und 99 zugewiesen werden.

Um nun das Debugging für alle Klassen auf den Wert 40 zu setzen, kann der folgende Befehl genutzt werden:

```
# echo "D A 40" > /var/run/isakmpd.fifo
```

Über die Fifo können auch weitere Eigenschaften von `isakmpd` gesteuert werden. So ist es möglich Verbindungen zu starten oder zu beenden.

10.1.4 Weitere Werkzeuge für das Debugging

Das wichtigste Werkzeug ist wahrscheinlich der Befehl `ping`. Dieser Befehl prüft die Erreichbarkeit eines weiteren Rechners und ist sicherlich jedem Leser bekannt. Mit dem Befehl `ping` sollte vor allem die Erreichbarkeit des VPN-Peers vor der Aktivierung des VPNs getestet werden. Besteht bereits ohne VPN keine Konnektivität, so wird diese wahrscheinlich nicht durch den Aufbau des VPNs entstehen.

Weitere wichtige Befehle sind `traceroute`, `tcpdump` und `ethereal`. Insbesondere die letzten beiden Befehle sind besonders interessant, da sie die Angabe des verwendeten Algorithmus und des Schlüssels erlauben. Sie sind dann in der Lage die übertragenen IPsec Pakete zu entschlüsseln und im Klartext darzustellen. Dies kann bei der Fehlersuche sehr hilfreich sein. Der Befehl `tcpdump` verwendet hierfür die Option `-E algo:secret`.

10.2 Typische Fehler und ihre Ursachen

Im folgenden werden einige typische Fehlermeldungen und ihre häufigsten Ursachen vorgestellt und besprochen. Dies ist sicherlich keine erschöpfende Liste. Der Autor würde sich über Erweiterungen und Anregungen freuen und wird diese auf seiner Homepage in einer aktuellen Version einpflegen.

10.2.1 FreeS/WAN: gmp.h: No such file or directory

FreeS/WAN benötigt bei der Übersetzung die GNU Multi Precision Bibliothek. Diese Bibliothek ist bei den meisten Distributionen in zwei Paketen enthalten: `gmp` und `gmp-devel`. Die benötigte Datei `gmp.h` befindet sich in dem Paket `gmp-devel`.

10.2.2 Isakmpd: bitstring.h: No such file or directory

`isakmpd` benötigt bei der Übersetzung die BSD-Datei `bitstring.h`. Diese Datei ist bei den meisten Linux Distributionen nicht enthalten, und muss aus den BSD Quellen besorgt werden, wenn `isakmpd` selbst übersetzt werden soll.

10.2.3 FreeS/WAN: _updown: »route add -net ...« failed

Das Kommando `route` wird vom `_updown` Skript aufgerufen um nach der Aktivierung des Interfaces `ipsecX` die entsprechende Route zu setzen. Dieser

Fehler deutet darauf hin, dass die verwendeten Netzwerkadressen und das Gateway nicht existieren oder erreicht werden können. Prüfen Sie die Erreichbarkeit.

10.2.4 FreeS/WAN: ipsec_setup: Fatal error, kernel appears to lack KLIPS

FreeS/WAN besteht aus zwei Bestandteilen: `KLIPS` im Kernel und `pluto` im Userspace. Hier ist entweder die Installation von `KLIPS` nicht korrekt durchgeführt worden oder es wird gerade der falsche Kernel verwendet. Dieser Kernel verfügt über keine IPsec-Funktionalität.

10.2.5 FreeS/WAN: Abbruch einer PPP/PPPOE Verbindung setzt falsche Defaultroute

Wenn eine PPP- oder PPPOE-Einwahlverbindung durch einen Provider beendet wird, so wird das ppp0 Interface entfernt. Es bleibt aber die Defaultroute auf das `ipsec0` Interface gesetzt. Dadurch wählt sich der Rechner nicht wieder ein. Um dies zu verhindern, müssen die `/etc/ppp/ip-[up|-down].local` Skripte angepasst werden, damit nach dem Start des Interfaces auch `ipsec setup start` beziehungsweise nach der Beendigung auch `ipsec setup stop` ausgeführt wird.

10.2.6 FreeS/WAN: Die Interfaces ipsec4 und folgende sind nicht verfügbar

FreeS/WAN unterstützt von Haus aus nur maximal vier `ipsec` Interfaces (0-3). Um mehr Interfaces zur Verfügung zu haben, ist die folgende Änderung im Quelltext durchzuführen und der Kernel neu zu übersetzen:

```
#define IPSEC_NUM_IF    4
```

Listing 10.1 Datei: `klips/net/ipsec/ipsec_tunnel.h`

10.2.7 FreeS/WAN: INTERNAL ERROR

Die Protokolldatei enthält Einträge wie:

```
vpn-gateway pluto[3303]: INTERNAL ERROR: /proc/net/ipsec_eroute
line 852 source subnet field malformed: non-ipv6 address may not
contain `:'
```

Prüfen Sie ob Pluto und KLIPS unterschiedliche Versionen sind. Hierzu kann der Befehl `ipsec –version` verwendet werden:

```
# ipsec --version
Linux FreeS/WAN U2.00/K2.01
See `ipsec --copyright' for copyright information.
```

Hier ist Pluto (Userland, U) Version 2.00 während KLIPS (K) Version 2.01 ist.

10.2.8 FreeS/WAN: fatal error in »packetdefault«: %defaultroute requested but not known

Dieser Fehler tritt bei FreeS/WAN in den Versionen 2.0 aufwärts auf, wenn die Policy Groups nicht deaktiviert wurden. Die Deaktivierung der Policy Groups ist in 5.6, »FreeS/WAN 2.x« beschrieben.

10.2.9 FreeS/WAN: connect() for "/var/run/pluto.ctl" failed

Pluto wurde gestartet, aber stürzte ab, so dass er nicht läuft. Dies wird häufig durch einen Fehler in der Datei `ipsec.secrets` verursacht. Pluto reagiert auf Syntaxfehler in dieser Datei sehr häufig mit einem Absturz. Wichtig ist die Einhaltung der exakten Syntax.

10.2.10 FreeS/WAN: Nach dem Upgrade von 1.98 auf 1.99 oder 2.0 funktioniert der Tunnel nicht mehr

Wenn für die Authentifizierung der VPN Peers X.509 Zertifikate eingesetzt werden, so wurde bis einschließlich X.509 Patch Version 0.9.27 die Verwendung der Datei `/etc/x509cert.der` als Zertifikat unterstützt. Ab der Version 0.9.28 beziehungsweise 1.0.0 wird diese Datei nicht mehr automatisch von FreeS/WAN gelesen. Wurde das Zertifikat in der Datei `x509cert.der` gespeichert, so kann dennoch sehr leicht ein Upgrade durchgeführt werden. Hierzu wird diese Datei mit dem Parameter `leftcert` oder `rightcert` spezifisch geladen. FreeS/WAN kann das DER-Format ohne weitere Probleme einlesen.

10.2.11 Tunnel ist vorhanden aber ein Ping ist nicht möglich

Achtung – wenn ein VPN-Tunnel zwischen zwei Netzwerken aufgebaut wird, dürfen die Gateways zunächst diesen Tunnel nicht benutzen. Es ist also nicht möglich von einem Gateway in das andere Gateway oder Netzwerk einen Ping zu senden. Der Grund liegt in der Absender IP Adresse die das Gateway wählt. Der Routing Code wählt das externe Interface und daher die externe IP Adresse als Absender IP Adresse. Diese darf aber nicht im Tunnel genutzt werden. Daher wird das Paket verworfen. Entweder wird ein zusätzlicher Tunnel erzeugt, oder die Lösung aus 9.2.1, »Gateway Routing« genutzt.

11 Testumgebungen

Es bietet sich häufig an, bevor ein VPN in einem produktiven Netz genutzt wird, den Einsatz und die Konfiguration in einer Testumgebung zu evaluieren und zu prüfen. Dieses Kapitel stellt verschiedene Möglichkeiten für den Aufbau einer derartigen Testumgebung vor. Hierbei existieren grundsätzlich zwei verschiedene Möglichkeiten. Entweder werden sämtliche benötigten Rechner für den Aufbau des Testfalles in eigenständiger physikalischer Hardware realisiert. Dies benötigt jedoch umfangreiche Hardware Ressourcen, wie Rechner Kabel, Hubs, Switches und so weiter. Von Vorteil ist jedoch, dass direkt die Leistungsfähigkeit der eingesetzten Hardware in der VPN Lösung getestet werden kann. Die Alternative ist eine Virtualisierung der gesamten Testumgebung. Hierbei werden die benötigten Rechner mit Hilfe von Softwareprodukten emuliert. Es werden daher keine zusätzlichen Rechner oder Netzwerkhardware benötigt. Am bekanntesten ist sicherlich die Rechneremulation mit VMware. Dieses Produkt emuliert einen Intel PC und erlaubt die Installation eines beliebigen Betriebsystems. Wenn jedoch nur die Emulation eines Linux Rechners auf der Basis des Linux Betriebssystems gewünscht wird, so genügt in den meisten Fällen User Mode Linux. Diese Linux Variante stellt außerdem geringere Anforderungen an die genutzte Hardware als VMware.

11.1 Testumgebungen

Dieser Abschnitt stellt die in diesem Buch verwendete Testumgebung vor und erklärt ihren Aufbau. Diese Umgebungen sind sehr generisch gehalten worden und versuchen die realen Verhältnisse im Internet mit möglichst geringen Mitteln nachzustellen, um möglichst jedes in diesem Buch vorgestellte Szenario nachstellen zu können.

Im Grunde werden zwei verschiedene Testumgebungen benötigt:

- Eine einfache Testumgebung I, bei der zwei Netzwerke über zwei VPN Gateways miteinander kommunizieren. Das Internet wird durch einen zusätzlichen Router zwischen den VPN Gateways emuliert.
- Eine Testumgebung II, bei der einzelne Rechner mit dynamischen IP Adressen über einen Router auf ein VPN Gateway zugreifen.

Wenn die Testumgebungen aufgebaut werden, sollte vor dem Einsatz von FreeS/WAN getestet werden, ob diese Umgebungen funktionieren. Dazu kann mit `ping` die Konnektivität geprüft werden.

11.1.1 Testumgebung I

Diese Testumgebung (Abbildung 11.1) stellt den häufigsten Fall einer VPN Lösung in einem Unternehmen dar. Dieses Unternehmen verfügt über zwei lokale Netzwerke. Sie befinden sich in New York und Berlin. Das Netzwerk in New York verwendet die IP Adressen 10.0.1.0/24. Das Netzwerk in Berlin verwendet die IP Adressen 10.0.2.0/24. In beiden Netzen gibt es ein Standard Gateway. Es ist unter der IP Adresse 10.0.1.1 beziehungsweise 10.0.2.1 erreichbar. Beide Gateways verfügen über eine statische IP Adresse im Internet. New York verwendet die IP Adresse 3.0.0.1 mit einer Netzmaske von 255.0.0.0 und einem Standard Gateway von 3.255.255.254. Berlin verwendet die IP Adresse 5.0.0.1/8 mit dem Standard Gateway von 5.255.255.254.

New York und Berlin sind in der Realität über das Internet miteinander verbunden. Hier wird das Internet durch einen Router simuliert, der über zwei Netzwerkkarten mit den IP Adressen 3.255.255.254/8 und 5.255.255.254/8 verfügt.

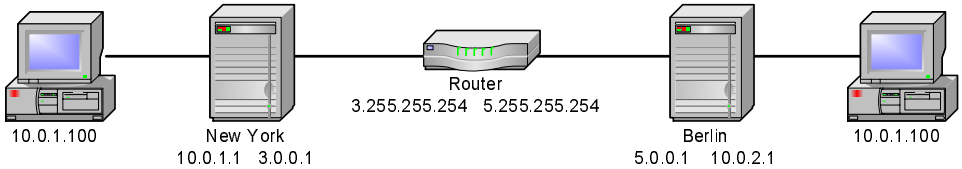


Abbildung 11.1 Testumgebung I

11.1.2 Testumgebung II

Diese Testumgebung wird für den Test eines Roadwarrior Szenarios und den Test des NAT Traversals benötigt. Hierbei gleicht die linke Hälfte des Aufbaus der Testumgebung I. Es existiert hier ebenfalls ein Netzwerk New York, das die IP Adressen 10.0.1.0/24 verwendet. Dieses Netzwerk ist über ein Gateway mit dem Internet verbunden. Das Gateway ist dazu mit zwei Netzwerkkarten ausgestattet. Die interne Karte verwendet die IP Adresse 10.0.1.1/24. Die externe Karte verwendet die IP Adresse 3.0.0.1/8 mit einem Standardgateway von 3.255.255.254.

Der Router simuliert erneut das Internet. Über den Router greifen nun Clients mit dynamischen IP Adressen auf New York zu. Hinter den Clients kann sich ein weiteres Netzwerk (Client1, 192.168.3.0/24) befinden. Client1 kann damit auch für den Test des NAT Traversal verwendet werden. Dazu

wird auf Client1 NAT aktiviert. Der VPN Aufbau erfolgt dann von den Rechnern NAT Client1 und NAT Client2 hinter Client1. Client2 ist ein weiterer Client mit dynamischer IP Adresse.

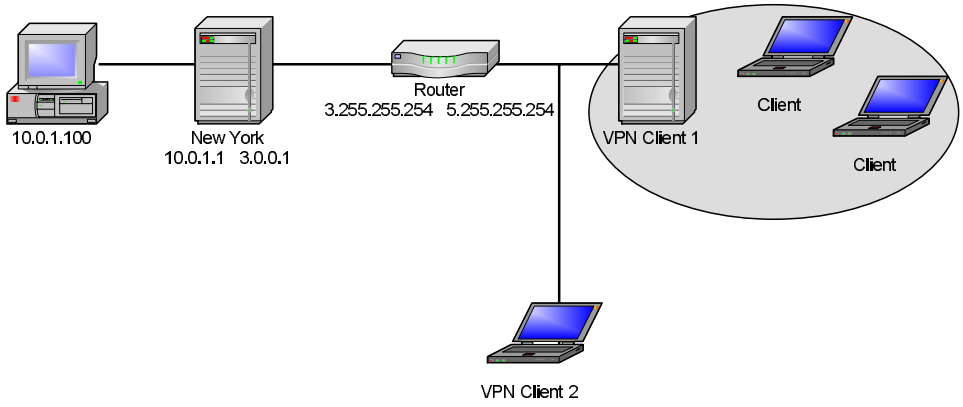


Abbildung 11.2 Testumgebung II

11.2 Physikalische Testumgebungen

Es ist möglich diese Testumgebungen physikalisch aufzubauen. Hierzu sind jedoch eine große Anzahl von Rechnern inklusive Hubs, Switches und Kabeln erforderlich. Ein physikalischer Testaufbau erlaubt jedoch eine Abschätzung der tatsächlichen Leistungsfähigkeit der Hardware. So kann geprüft werden, ob die ausgewählte Hardware später im Produktionseinsatz in der Lage ist die gewünschte Leistung zu erbringen.

Für erste Tests ist es jedoch sinnvoller und meist auch schneller und preiswerter, die entsprechenden Aufbauten zu emulieren.

11.3 VMware

VMware Workstation ist ein kommerzielles Produkt. Es wird von VMWare (*http://www.vmware.com*) hergestellt und stellt das kleinste Produkt aus einer Reihe mit dem VMware GSX Server und dem VMware ESX Server dar. Es erlaubt die Virtualisierung eines Intel PCs auf der Basis von Linux oder Win32 Bit Betriebsystemen. So besteht die Möglichkeit auf diesem virtualisierten PC ein weiteres Intel Betriebsystem (Windows, Linux, *BSD oder Solaris) zu in-

stallieren. VMware unterstützt dabei eine Vernetzung der virtuellen Rechner.

VMware Workstation ist als Evaluationsversion für 30 Tage erhältlich. Anschließend kostet eine Lizenz 299 Dollar.

Die Installation von Linux oder Windows Betriebsystemen erfordert anschließend keine weiteren Anpassungen. Es wird lediglich empfohlen, die VMware Tools zu installieren. Sie bieten zum Beispiel einen angepassten Grafikkartentreiber, um das Gastbetriebssystem im Vollbild Modus zu betreiben.

Für die Vernetzung bietet VMware die Erzeugung von virtuellen Netzwerken. Dazu werden auf dem Host Betriebsystem virtuelle `vmnetX` Karten erzeugt, die dem Host Zugang zu diesen Netzwerken geben.

Der Aufbau sämtlicher Testumgebungen ist mit VMware Workstation möglich. Für Einzelheiten lesen Sie bitte die VMware Bedienungsanleitung.

11.4 User-Mode-Linux

User-Mode-Linux (UML) ist ein Projekt (*http://user-mode-linux.sourceforge.net/* von Jeff Dike, das den Start eines Linux Betriebsystems auf einem bereits gestartetem Linux System ermöglicht. Auch diese Lösung bietet die Möglichkeit virtuelle Netzwerke aufzubauen. Dabei kann UML auch vollkommen von der Umwelt abgeschnittene Netzwerke aufbauen. Hierzu wird ein Switch Daemon zur Verfügung gestellt.

11.4.1 Kernelbau

Damit der Kernel im Usermode funktionieren kann, sind einige Änderungen erforderlich. Ein Großteil des Usermode Codes befindet sich bereits in den aktuellen Kerneln. Jedoch ist es für die Funktionalität und Stabilität von Vorteil, den aktuellsten Patch für diesen Zweck zu nutzen. Laden Sie zunächst den aktuellen Linux Kernel von *http://www.kernel.org* und den Usermode Patch von *http://user-mode-linux.sourceforge.net/dl-sf.html*. Hier werden Patches für den Linux Kernel 2.4, 2.5 und 2.6 vorgehalten.

TIPP

User Mode Linux unterstützt seit einiger Zeit ein Separate Kernel Address Space (SKAS). Diese Funktion beschleunigt einen User Mode Linux Kernel um 30 bis 50 Prozent. Jedoch ist hierfür ein Patch des Host Kernels erforderlich. Sie können weitere Informationen über SKAS auf *http://user-mode-linux.sourceforge.net/skas.html* nachlesen.

Entpacken Sie den Linux Kernel in einem geeigneten Verzeichnis und patchen Sie ihn mit dem Usermode Patch.

```
# mkdir /usermode
# cd /usermode
# tar -xjf />path</linux-<version>.tar.bz2
# cd linux-2.4.19
# bzcat />path</uml-patch-<version>.bz2 | patch -p1
# make ARCH=um oldconfig dep
```

Wenn Sie FreeS/WAN benutzen möchten, entpacken Sie nun FreeS/WAN in der gewünschten Version und wenden entsprechende FreeS/WAN Patches an. Anschließend starten Sie die FreeS/WAN Übersetzung. Wenn Sie den Linux Kernel 2.5 oder 2.6 mit dem nativen IPsec Stack nutzen wollen, ist dies nicht nötig.

```
# cd ..
# tar -xzf />path</freeswan-<version>.tar.gz
# cd freeswan-1.99
# make ARCH=um KERNELSRC=../linux-<version> insert mcf confcheck kernel
```

Hierbei kann es zu einem Fehler bei der Übersetzung kommen, da das FreeS/WAN Skript versucht ein `bzImage` zu generieren. Die Architektur `ARCH=um` erlaubt aber nur die Erzeugung eines Kernels `linux`. Um dies anzupassen ist die Datei `freeswan-<version>/Makefile.inc` zu editieren.

```
KERNEL=$(shell if expr " `uname -m`" : ' i.86' >/dev/null ; \
        then echo linux ; \
        else echo boot ; \
        fi)
```

Geben Sie für die Übersetzung nicht den Befehl `make kinstall` ein. Er wird versuchen den Kernel und die Programme direkt auf ihrem Host System zu installieren.

Wenn das Kernel Konfigurationswerkzeug startet, wählen Sie die Unterstützung für Kernelmodule ab, so das ein monolithischer Kernel erzeugt wird. Dies vereinfacht später die Verwendung des Kernels.

Nun wird der Kernel übersetzt. Dies wird eine Weile dauern. Anschließend sollten Sie den erzeugten Linux Kernel mit einem sinnvollen Namen sichern.

```
# cp ../linux-<version>/linux ../linux-<version>-freeswan-<version>
```

11.4.2 UML Installation

Für die Verwendung von User Mode Linux werden einige Werkzeuge auf dem Host benötigt. Dies sind die UML Utilities. Entweder Sie installieren das auf der User Mode Linux Downloadseite verfügbare RPM Paket oder Sie kompilieren die UML Utilities von Hand. Wenn Sie das RPM wählen, sollten Sie daran denken, dass mit diesem RPM auch ein aktueller User Mode Linux Kernel installiert wird. Er befindet sich im Gegensatz zu dem selbst übersetzten Kernel im Suchpfad Ihrer Shell.

Nun müssen Sie noch ein Dateisystem erzeugen. Auf der User Mode Linux Homepage werden einige vorgefertigte Root-Dateisysteme angeboten. Sie können diese aber auch relativ leicht selbst herstellen. Mit `mkrootfs` (*http://www.stearns.org/mkrootfs/*) und UML-Builder (*http://umlbuilder.sourceforge.net/*) stehen Ihnen zwei Anwendungen zur Verfügung, die die Erzeugung stark vereinfachen.

Zusätzlich ist auch eine Swap Partition erforderlich. Sie wird ebenfalls von den Werkzeugen erzeugt, kann aber auch mit dem folgenden Befehl erstellt werden.

```
# dd if=/dev/zero of=swapfs bs=1024k count=128
# mkswap swapfs
```

Auf der CD ist ein kleines einfaches Root Dateisystem enthalten. Es basiert auf der Red Hat Linux 9.0 Distribution.

11.4.3 Start von User-Mode-Linux

Nun können Sie User Mode Linux starten. Wählen Sie dazu ein Dateisystem aus. Anschließend können Sie mit dem folgenden Befehl eine User Mode Linux Session starten. Sie sollte dann automatisch über eine Netzwerkverbindung verfügen. Testen Sie sie, indem Sie sich anmelden (*login:root*, *kennwort:root*) und einen Ping auf ihre Hauptmaschine durchführen. Dort wird automatisch die Netzwerkkarte `tap0` mit der IP Adresse 3.255.255.254 aktiviert.

```
/<path>/linux umid="New-York" root=6200 ubd0=rootfs ubd7=swapfs
eth0=tuntap,,,3.255.255.254
```

11.4.4 Aufbau virtueller Netzwerke mit uml_switch

User Mode Linux bietet die Möglichkeit komplette eigenständige Netzwerke aufzubauen, die über einen Switch oder ein Hub miteinander verbunden sind. Diese Funktion wird vom Befehl `uml_switch` ausgeübt. Die Kommunikation zwischen den verschiedenen UML Sessions und dem `uml_switch` erfolgt dann über einen Socket. Der Befehl kennt die folgenden Optionen bei seinem Aufruf:

- **`-hub`** Der Befehl arbeitet als Hub und nicht als Switch.
- **`-unix`** *`socket`* Hiermit können mehrere Switche erzeugt werden. Jeder benötigt einen eigenen Kommunikationssocket. Der kann hier angegeben werden.
- **`-t`** *`tap-device`* Hiermit kann der Switch am Host angeschlossen werden. Dazu ist es erforderlich das Gerät `tapX` zuvor mit dem Befehl `tunctl -u` *`uid`* zu erzeugen. Diese Option ist nicht bei allen Versionen von `uml_switch` verfügbar.

Ein typischer Aufruf von `uml_switch` sieht dann wie folgt aus:

```
# uml_switch -unix /tmp/newyork.ctl
```

Beim Aufruf von User Mode Linux muss das entsprechende Interface, das an den Switch angebunden werden soll, angegeben werden mit:

```
eth1=daemon,mac-address,,/tmp/newyork.ctl,
```

Die *`mac-address`* muss durch eine eindeutige Adresse ersetzt werden, da ansonsten jeder Rechner, der mit dem Switch verbunden wird die identische Adresse erhält und keine Kommunikation möglich ist.

A Lizenzen

A.1 GNU GPL

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that

you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program. In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1

and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues),

conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program. If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License. 8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free

Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode: Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

B Die CD ROM zum Buch

Die CD ROM enthält zusätzliches Material, das dieses Buch gesprengt hätte, und die entsprechende Software, um die Beispiele in diesem Buch testen zu können.

B.1 RFC Dokumente

Die CD ROM enthält sämtliche Drafts und Standards, die für das Verständnis der eingesetzten Protokolle erforderlich sind. Außerdem sind verschiedene Dokumente auf der CD vorhanden.

B.2 Software

Die CD ROM enthält die aktuellsten Versionen von FreeS/WAN, `setkey`, `racoon` und `isakmpd`, die zum Zeitpunkt der Drucklegung verfügbar waren. Zusätzlich befinden sich auf der CD ROM RPM Pakete für den Einsatz dieser Software auf der Red Hat Distribution.

Außerdem befindet sich auf der CD ein Root Dateisystem basierend auf Red Hat Linux 9 für User-Mode-Linux.

C Glossar

3DES	Dieses symmetrische Verschlüsselungsverfahren wendet DES dreimal an. Hieraus resultiert ein Schlüssel von 112 oder 168 Bit Länge.
Advanced Encryption Standard	Ein symmetrisches Verschlüsselungsverfahren mit einer Schlüssellänge von 128 oder 256 Bit.
AES	*siehe* Advanced Encryption Standard
AH	*siehe* Authentication Header
Authentication Header	Dieses IPSec Protokoll garantiert die Unversehrtheit der übertragenen Daten. Es garantiert nicht die Vertraulichkeit.
Blowfish	Dieses symmetrische Verschlüsselungsverfahren erlaubt die Verwendung von bis zu 448 Bit langen Schlüsseln.
Data Encryption Standard	Dieses symmetrische Verschlüsselungsverfahren verwendet einen 56 Bit langen Schlüssel.
dDoS	*siehe* Distributed Denial of Service
Denial of Service	Ein Denial of Service ist die fehlende Verfügbarkeit eines Dienstes. Dies kann durch einen Absturz des Betriebssystems oder des Rechners, aber auch durch eine Überlastung des Systems hervorgerufen werden.
DES	*siehe* Data Encryption Standard
DHCP-over-IPsec	Diese Methode bietet die Möglichkeit automatisch IP Adressen an VPN Clients zu verteilen.
Distributed Denial of Service	Beim Distributed Denial of Service überlasten viele verteilte Rechner gleichzeitig einen einzelnen Rechner durch (meist gespoofte) Pakete.
DNS	*siehe* Domain Name Service
Domain Name Service	Dieser Dienst ist zuständig für die Auflösung von Rechnernamen in IP Adressen und umgekehrt. Er kann auch öffentliche RSA Schlüssel verteilen und wird für die opportunistische Verschlüsselung benötigt.

DoS	*siehe* Denial of Service
Encapsulated Security Payload	Dieses IPSec Protokoll garantiert die Sicherheit der übertragenen Daten. Hierzu werden die Pakete verschlüsselt und authentifiziert.
ESP	*siehe* Encapsulated Security Payload
File Transfer Protocol	Ein Protokoll, das für den Transport von Dateien verwendet wird. Die Dateien können in zwei verschiedenen Modi übertragen werden: aktiv und passiv. Bei der aktiven Übertragung, öffnet der Server eine Verbindung zum Client. Bei der passiven Übertragung öffnet der Client den Datenkanal.
Firewall	Ein Rechner oder eine Rechnerstruktur, die den Informationsfluss zwischen zwei Netzen entsprechend einer Sicherheitsrichtlinie überwacht und regelt.
FTP	*siehe* File Transfer Protocol
GnuPG	GNU Privacy Guard. Eine Open Source Alternative zu PGP (Pretty Good Privacy)
HTTP	*siehe* HyperText Transfer Protocol
Hub	Ein Repeater, der mehrere Netzwerksegmente physikalisch miteinander verbindet. Hierbei werden alle Pakete an alle angeschlossenen Segmente weitergeleitet.
HyperText Transfer Protocol	Das Applikationsprotokoll, das von Web Servern und Browsern für die Kommunikation genutzt wird.
ICMP	*siehe* Internet Control Message Protocol
IKE	*siehe* Internet Key Exchange
Internet Control Message Protocol	Dieses Protokoll wird für die Übertragung von Status- und Kontrollnachrichten verwendet.
Internet Key Exchange	Dieses Protokoll authentifiziert die Kommunikationspartner, ermittelt die Verschlüsselungs- und Authentifizierungsalgorithmen und Sitzungsschlüssel für die IP Sec Protokolle.
IP Adresse	Eine eindeutige nummerische Bezeichnung eines Teilnehmers in einem IP Netz.
IP	Internet Protocol

MD5 Message Digest Fünf. Ein krytographischer Prüfsummen Algorithmus.

MTU Maximum Transmission Unit

Multicast Paket Ein Paket, das an eine bestimmte Auswahl von Rechnern in einem Netzwerk gerichtet ist.

NAT Traversal NAT Traversal bietet die Möglichkeit die IPsec Protokolle über NAT Geräte einzusetzen. Hierzu werden die IPsec Pakete erneut in UDP Paketen eingepackt. So können die Pakete von NAT Geräten ohne Probleme weitergeleitet werden.

NAT Network Address Translation

Paketfilter Eine auf der Netzwerk- und Transportschicht implementierte Firewall. Die Informationen werden paketweise gefiltert und in Abhängigkeit vom Paket Header erlaubt oder verworfen.

PMTU Discovery Path Maximum Transmission Unit Discovery

Port Ein Port ist eine Nummer, die einen Kommunikationskanal des TCP oder UDP Protokoll bezeichnet. Dieser Port wird vom protokolleigenen Multiplexer zur Verfügung gestellt. Man unterscheidet privilegierte Ports (0-1023) und unprivilegierte Ports (1024-65535). Privilegierte Ports können nur mit root-Rechten benutzt werden.

Proxy Ein Proxy ist eine Anwendung, die auf Applikationsebene Verbindungen entgegen nimmt und aufbaut. Sie arbeitet als Man-in-the-Middle und kann auch Filterfunktionen übernehmen. So kann sie auch als Firewall eingesetzt werden.

RipeMD160 Ein kryptografischer Prüfsummen Algorithmus

Rjindael *siehe* Advanced Encryption Standard

Roadwarrior Ein Roadwarrior ist eine Person bzw. ein Rechner, der unter der Verwendung einer dynamischen IP Adresse auf ein VPN zugreift. Die IP Adresse kann daher nicht zur Authentifizierung des Roadwarrior genutzt werden.

SA *siehe* Security Association

Secure Shell Ein sicherer Ersatz für telnet, rsh, rcp, rexec und ftp. Sowohl die Authentifizierung als auch die Datenübertragung erfolgt verschlüsselt.

Security Association Eine Security Association definiert die zu verwendenden Algorithmen und Schlüssel für die Kommunikation zwischen zwei Rechnern. Eine Security Association ist immer unidirektional.

Security Policy Eine Security Policy definiert, wann welche Security Association anzuwenden ist.

SHA-1 Secure Hash Algorithm. Ein kryptografischer Prüfsummen Algorithmus

SP *siehe* Security Policy

Spoofing Eine Technik, bei der bestimmte Informationen gefälscht werden. Hierbei kann es sich um IP Adressen (IP Spoofing) IP/MAC Adresspaarungen (ARP Spoofing) und DNS/IP Paarungen (DNS Spoofing) handeln.

ssh *siehe* Secure Shell

SSL Die Secure Socket Layer wird von einigen Applikationsprotokollen genutzt, um eine authentifizierte und verschlüsselte Verbindung aufzubauen.

Switch Ein Switch ist ein Netzwerkgerät, das mehrere Segmente ähnlich einem Hub miteinander verbindet. Der Unterschied zu einem Hub, das die Pakete an alle angeschlossen Geräte weitersendet, ist, dass ein Switch das Paket nur an den entsprechenden Rechner mit der Ziel MAC Adresse weitergibt. Ein Switch ist also eine Art Router auf Layer 2.

TCP *siehe* Transmission Control Protocol

Transmission Control Protocol Dieses Protokoll garantiert die vollständige Zustellung aller Informationen in der richtigen Reihenfolge mit der höchsten möglichen Geschwindigkeit.

UDP *siehe* User Datagram Protocol

User Datagram Protocol Dieses Protokoll ermöglicht die Übertragung von einzelnen unabhängigen Nachrichten. Die Zustellung und die Reihenfolge des Nachrichtenempfangs wird nicht vom Protokoll garantiert.

D Bibliografie

Barrett, Daniel J., Richard E. Silverman: SSH: Secure Shell – Ein umfassendes Handbuch. 1. Aufl. Köln: O'Reilly 2001.

Bauer, Friedrich L.: Enzifferte Geheimnisse. Methoden und Maximen der Kryptologie. 3., überarbeitete Aufl. Berlin u. a.: Springer 2000.

Bellovin, William, Steven Cheswick: Firewalls und Sicherheit im Internet. 2., überarbeitete Aufl. Bonn u. a.: Addison-Wesley 1995.

Böhmer, Wolfgang: VPN. Virtual Private Networks. Die reale Welt der virtuellen Netze. 1. Aufl. München u. a.: Hanser 2002.

Cavallar, Stefania, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, Gérard Guilerm, Paul Leyland; et al.: Factorisation of a 512-bit RSA modulus;, In: Theory and Application of Cryptographic Techniques, *ftp://ftp.gage.polytechnique.fr/pub/publications/jma/rsa-155.ps*

Doraswamy, Naganand, Dan Harkins: IPSec. 1. Aufl. Bonn u. a.: Addison-Wesley 2000.

Hall, Eric A.: Internet Core Protocols: The Definitive Guide. 1. Aufl. Sebastopol u. a.: O'Reilly 2000.

Kahn, David: The Codebreakers. 2., überarbeitete Aufl. New York: Simon & Schuster Inc. 1997.

Lipp, Manfred: VPN – Virtuelle Private Netzwerke. Aufbau und Sicherheit. 1. Aufl. Bonn u. a.: Addison-Wesley 2001.

Marsh, Matthew G.: Policy Routing Using Linux. 1. Aufl. Indianapolis u. a.: SAMS 2001.

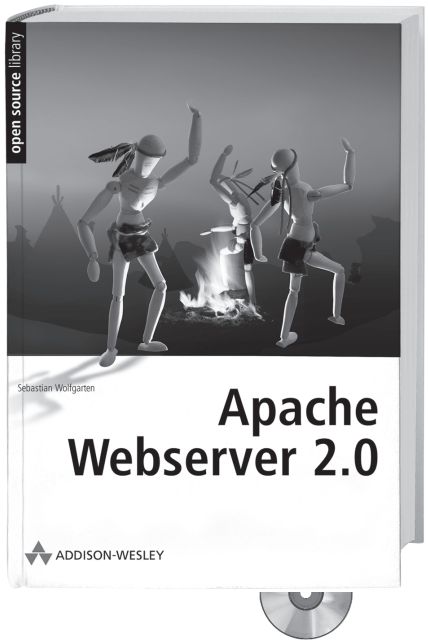
Dr. Pohlmann, Norbert, Markus a Campo: Virtual Private Networks. 2. überarbeitete Aufl. Bonn: MITP 2003.

Schneier, Bruce: Applied Cryptography. 2., überarbeitete Aufl. New York u. a.: John Wiley & Sons 1995.

Stevens, W. Richard: TCP/IP Illustrated. Bd. 1. 1. Aufl. Reading u. a.: Addison Wesley 1994.

Ziegler, Robert L.: Linux Firewalls. 2., überarbeitete Aufl. München: Markt+Technik Verlag 2002.

Stichwortverzeichnis



Apache Webserver 2.0

Installation, Konfiguration, Programmierung

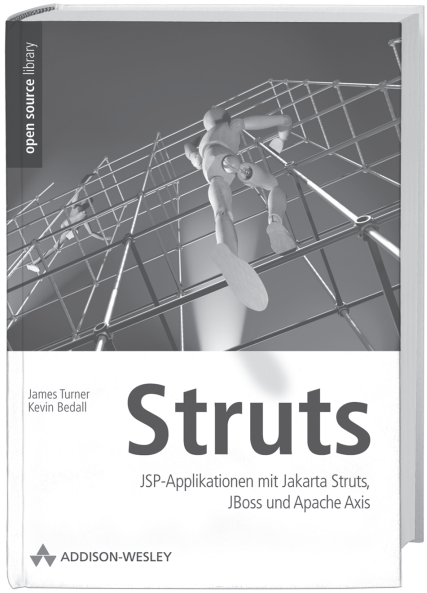
Sebastian Wolfgarten

Der mit dem Web tanzt! Der Apache Webserver, mit über 60 % der meistverbreitete Webserver überhaupt, ist in der Version 2 völlig überarbeitet worden. Hier erhalten Sie das umfassende Wissen zur Installation, Konfiguration und Programmierung des Apache Webserver 2.0. Neben den Neuerungen wie Multiprotokollsupport, Ein- und Ausgabefilter oder Laufzeitmodellen werden auch besonders Themen wie CGI, SSI, PHP, Perl, Tomcat und SSL berücksichtigt. Egal, ob Sie unter Linux, Windows, FreeBSD oder Solaris arbeiten – hier erfahren Sie, wie Sie den Apache 2.0 effizient einsetzen.

Open Source Library

758 Seiten, 1 CD-ROM
€ 39,95 [D]/ € 41,10 [A]
ISBN 3-8273-2039-9

THE SIGN OF EXCELLENCE



Struts

James Turner / Kevin Bedell

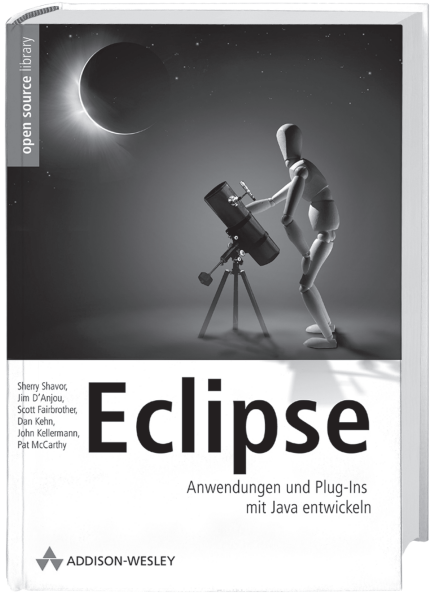
Auf Basis der Java-Technologie lassen sich mit dem Struts-Framework einfach und schnell komplexe, dynamische Websites entwickeln. Dieses Buch gibt einen grundlegenden Einstieg in die Welt der Web-Applikations-Entwicklung. Zahlreiche anschauliche und gut nachvollziehbare Beispiele erläutern die Features des Frameworks und führen schnell zum Erfolg.

Open Source Library

480 Seiten, 1 CD
€ 44,95 [D] / € 46,30 [A]
ISBN 3-8273-2113-1

www.addison-wesley.de

ADDISON-WESLEY



Eclipse

Sherry Shavor u. a.

Das US-Original dieses Buches ist die zur Zeit populärste Einführung in die Programmierung von und mit Eclipse. Die Autoren führen in den Gebrauch der IDE zur Entwicklung von Java-Anwendungen ein. Sie zeigen zudem ausführlich, wie eigene Plug-Ins zur Erweiterung von Eclipse geschrieben werden können. Das Buch ist aufgebaut wie ein Tutorial, jedes Kapitel entspricht einer Sitzung und enthält Beispiele und Übungen zur Vertiefung des Stoffes. Die beiliegende CD enthält neben der Eclipse-Software auch diese Übungsdateien sowie weitere Arbeitsbeispiele.

Open Source Library

850 Seiten, 1 CD
€ 59,95 [D] / € 61,70 [A]
ISBN 3-8273-2125-1