Linux ■ IPsec mit PGP und X.509 Zertifikaten

Projektarbeit

Dozent: Dr. Andreas Steffen

Studenten: Andreas Hess
Patric Lichtsteiner
Roger Wegmann

Abgabetermin: 21. Juli 2000

1 INHALTSVERZEICHNIS

2 ZUSAMMENFASSUNG

Unsere Aufgabe bestand darin, eine VPN-Verbindung zwischen PGPnet unter Windows NT und FreeS/WAN unter SuSE Linux 6.4 aufzubauen. Die Authentifikation sollte neben Preshared Keys und PGP-Keys auch mit X.509-Zertifikaten funktionieren. Während PGPnet diese Authentifikations-mechanismen bereits unterstützte, musste FreeS/WAN um die X.509-Authentifikation erweitert werden. Als Grundlage diente uns die Diplomarbeit "Virtual Private Network mit sicherem Tunnel durchs Internet" von Olivier Gärtner und Berkant Uenal.

Als erstes haben wir eine VPN-Verbindung zwischen zwei PGPnet-Hosts aufgebaut. Dies hat mit allen Authentifikationsmethoden problemlos funktioniert. Anschliessend stellten wir eine VPN-Verbindung zwischen PGPnet und FreeS/WAN mit Preshared Keys her. Auch hier sind keine grösseren Probleme aufgetreten. Für die Authentifikation mit PGP-Keys mussten wir zuerst einen Patch für FreeS/WAN installieren und die Konfigurationsdateien anpassen. Nach einigen Problemen mit der Kompilation des für die Ermittlung der Schlüsselparameter benötigten Hilfsprogramms, hat auch dies funktioniert.

Nun begann unsere eigentliche Arbeit: Der FreeS/WAN-Quelltext sollte erweitert werden, sodass X.509-Zertifikate für die Authentifikation unterstützt werden. Als Erstes mussten wir herausfinden, welche Teile des Quelltexts an der gegenseitigen Authentifikation der Hosts beteiligt sind. Wir stellten fest, dass der Mechanismus für die Authentifikation beinahe identisch mit dem der PGP-Keys ist. Dadurch konnten wir den Quelltext in der gegebenen Zeit erfolgreich anpassen. Die grösste Hürde stellte die Analyse der X.509-Zertifikate für die Ermittlung der Schlüsselparameter dar. Dieses Problem konnten wir mit den OpenSSL-Funktionen und der Entwicklung eines eigenen ASN.1-Parsers lösen.

Schlussendlich konnten wir eine VPN-Verbindung mit X.509-Zertifikaten zwischen PGPnet und dem von uns modifizierten FreeS/WAN erfolgreich aufbauen.

Für uns war diese Arbeit sehr lehrreich. Wir konnten das im Fach Kommunikationssysteme gelernte Wissen über Secure Network Communication und ASN.1- bzw. BER-Codierung vertiefen. Ausserdem haben wir gelernt, einen umfangreichen OpenSource Quelltext zu analysieren und anzupassen.

Winterthur, den 21. Juli 2000

Andreas Hess Patric Lichtsteiner Roger Wegmann

3 AUFGABENSTELLUNG

3.1 BESCHREIBUNG

Mit dem OpenSource Paket "Free S/WAN" kann Linux um einen vollwertigen IPsec Stack ergänzt werden. In vielen Firmen versieht ein Linux-Server seinen Dienst als Firewall / Security-Gateway, mit der Aufgabe das Intranet von der Aussenwelt abzuschotten. Man möchte jedoch auf der Basis von authentisierten und verschlüsselten IPsec Verbindungen den eigenen Mitarbeitern von zu Hause aus oder von unterwegs mit mehrheitlich Windows-basierten Clients den Zugriff auf die firmeninternen Ressourcen ermöglichen. Als IPsec Windows-Client eignet sich dabei PGPnet, das sich leicht installieren und konfigurieren lässt.

Linux Free S/WAN wurde kürzlich durch Kai Martius um einen Patch ergänzt, der es erlaubt, die Authentisierung zwischen einem PGPnet Client und einem Linux Security-Gateway auf der Basis von PGP Zertifikaten durchführen zu können. Diese Projektarbeit soll versuchen, diesen Patch auf X.509 Zertifikate zu erweitern.

3.1.1 Aufgaben

- Installation von PGPnet Version 6.5.1i oder Version 6.5.3 unter Windows NT, Installation von Free S/WAN Version 1.3 unter Linux
- Integration des PGP Patch von Kai Martius
- Configuration und Betrieb einer IPsec Verbindung zwischen einem PGPnet Client und einem Linux FreeS/WAN Security-Gateway
- Analyse des PGP Patches und der Free S/WAN Struktur
- Studium der OpenPGP und X.509 Zertifikat-Formate
- Lösungsvorschlag für die Unterstützung von X.509 Zertifikaten durch Free S/WAN
- Falls machbar, Realisierung des Lösungsvorschlags

3.1.2 Infrastruktur / Tools

- Raum: E416
- Rechner: 4 PCs mit SuSE Linux 6.4 / Windows NT 4.0

3.1.3 SW-Tools

- PGPnet 6.5.1i oder 6.5.3
- FreeS/WAN Version 1.3 mit Patch von Kai Martius

4 EINLEITUNG

4.1 EINFÜHRUNG

Sicherheit war schon immer ein Grundbedürfnis der Menschheit. Dies hat sich im Internetzeitalter, in dem der gläserne Mensch immer mehr zur Wirklichkeit wird, nicht verändert. Im Gegenteil, das Sicherheitsbedürfnis ist grösser denn je.

Ein möglicher Lösungsansatz für abhörsichere Verbindungen im Internetbereich ist Virtual Private Network (VPN).

Zu einer sicheren Verbindung gehört auch die Authentifikation der Endpunkte. IPsec, eine Implementation von VPN, erlaubt verschiedene Authentifizierungsmechanismen, unter anderem auch X.509-Zertifikate. X.509-Zertifikate werden bereits für verschiedene Zwecke verwendet, beispielsweise für signierte E-Mails. Aus diesem Grund ist es naheliegend, dass man sie auch für IPsec einsetzt.

Heute wird Linux in Firmen häufig als Firewall eingesetzt. Leider existiert zur Zeit jedoch noch kein Linux-Firewall, der IPsec mit X.509-Zertifikaten unterstützt.

4.2 HINWEISE ZUR BENÜTZUNG DIESER DOKUMENTATION

Unsere Dokumentation ist in folgende Kapitel unterteilt:

- Grundlagen
  Dieses Kapitel gibt einen groben Überblick über die Techniken, die von VPN angewandt werden. Dieses Wissen wird bei den folgenden Kapiteln vorausgesetzt.
- Entwicklungs- und Testumgebung
  Um die folgenden Kapitel anschaulicher zu gestalten, zeigen wir hier dem Leser kurz unsere Netzwerktopologie. Alle nachfolgenden Beispiele beziehen sich auf diese Testumgebung.
- Grundkonfiguration
  Hier werden die Grundeinstellungen von PGPnet und FreeS/WAN erklärt.
- Anpassen des Quelltexts für die Verwendung von X.509-Zertifikaten
  Dieses Kapitel beschreibt die Massnahmen, die nötig waren, damit FreeS/WAN mit X.509-Zertifikaten umgehen kann.
- Konfiguration der verschiedenen Authentisierungsmethoden
  Als nächstes wird Schritt für Schritt erklärt, wie die verschiedenen Authentisierungsmethoden in PGPnet und FreeS/WAN konfiguriert werden können.

- Ausblick und Weiterentwicklung
  Hier werden einige Ideen aufgeführt, in welchen Richtungen man diese Projektarbeit weiterführen könnte.

5 GRUNDLAGEN

5.1 VIRTUAL PRIVATE NETWORK (VPN)

Der Begriff "Virtual Private Network" beschreibt die Absicht, über ein öffentliches Netz eine sichere (private) Verbindung zwischen zwei oder mehreren Hosts (Netzwerken) herzustellen. Beim öffentlichen Netz handelt es sich in den meisten Fällen um das Internet. Obwohl das Internet eine billige und einfach Art darstellt, ein WAN zu realisieren, wurde es in der Vergangenheit in Bereichen, in denen Sicherheit eine vorrangige Rolle spielt, kaum eingesetzt. Der Grund ist der vom Internet verwendete Protokollstack, der bis zum Transport Layer hinauf keine Sicherheitsvorkehrungen implementiert hat (gilt nur für IPv4). Mit der Einführung von VPN konnte das Sicherheitsproblem von Internetverbindungen behoben werden und es wurde beispielsweise möglich, Aussendienstmitarbeiter und ganze Netzwerke von Zweigstellen über das Internet sicher mit dem Firmennetzwerk zu verbinden. Das Secure Internet Protocol (IPsec) ist eine Implementation von VPN und wird auf den folgenden Seiten kurz beschrieben.

5.2 SECURE INTERNET PROTOCOL (IPSEC)

Das Ziel von IPsec ist eine sichere Kommunikation über das Internet zu gewährleisten. Das beinhaltet folgende Punkte:

- Integrität der Daten sicherstellen
- Authentifikation der Daten unterstützen
- Vertraulichkeit der Daten und der Verbindung gewährleisten
- Zugriffskontrolle auf Benutzerebene ermöglichen

Damit die Vertraulichkeit der Verbindung - dazu gehört die Anonymität des Senders und Empfängers - gewährleistet werden kann, muss IPsec schon beim Network Layer in den Protokollstack eingefügt werden. Würden die Daten erst weiter oben im Protokollstack von IPsec bearbeitet, könnte jeder Router, der an der Verbindung beteiligt ist, den Sender und den Empfänger einer Nachricht feststellen.

Der Network Layer im Internet Protokollstack wird in den meisten Fällen durch das Internet Protokoll IPv4 realisiert. IPsec hat also die Aufgabe, IPv4 um ein Sicherheitsprotokoll zu erweitern. Beim Internet Protokoll IPv6 wäre das nicht mehr notwendig, weil in IPv6 bereits ein Sicherheitsprotokoll integriert ist.

Bis zum Transport Layer hinunter ist IPsec völlig transparent. Als Sicherheitsprotokoll des Network Layers bietet es Schutz für alle Daten und Applikationen in den darüber liegenden Schichten, ohne das die Applikationen angepasst werden müssen.

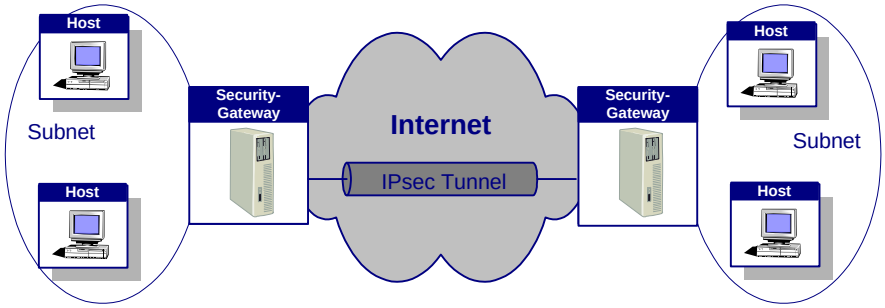
5.2.1 Zwei Verbindungsmodi

IPsec stellt für eine Verbindung den Transport und den Tunnel Mode zur Verfügung.

5.2.1.1 Transport Mode

Im Transport Mode stellen die beiden Hosts eine direkte Verbindung zueinander her. Der Transport Mode kann zwar die Integrität, die Authentizität und die Vertraulichkeit der Daten sicherstellen, nicht aber die Vertraulichkeit des Senders und Empfängers. Dies ist nicht möglich, weil die beiden Hosts direkt über das Internet miteinander kommunizieren und somit ihre Identität öffentlich preisgeben. Weil die beiden Hosts direkt miteinander kommunizieren, müssen sie sich selbst um die Sicherheit kümmern.

5.2.1.2 Tunnel Mode



IPsec Tunnel Mode

Für den Tunnel Mode wird auf beiden Seiten der Verbindung ein Security-Gateway benötigt. Die beiden Hosts, die miteinander kommunizieren wollen, befinden sich hinter den Security-Gateways. Die Pakete der Hosts werden von den Security-Gateways in neue Pakete verpackt und erst dann über das Internet geschickt. Mit dieser Technik sind nur die Informationen der Security-Gateways im Internet sichtbar und die Informationen des ursprünglichen Senders und Empfängers bleiben verborgen, wenn diese Daten verschlüsselt werden. Somit kann der Tunnel Mode zusätzlich zu den Sicherheitsvorkehrungen im Transport Mode auch die Vertraulichkeit des Senders und Empfängers gewährleisten. Weil im Tunnel Mode die Security-Gateways für das Einhalten der Sicherheitsbestimmungen zuständig sind, müssen sich die Hosts nicht um die Sicherheit kümmern. Die Security-Gateways stellen sozusagen einen sicheren Tunnel für die Hosts bereit.

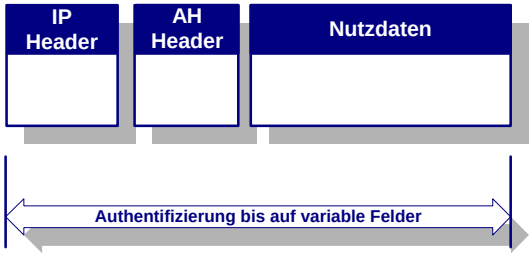
5.2.2 Zwei Sicherheitsfunktionen

Die Sicherheitsfunktionen von IPsec werden durch den IP Authentication Header und die IP Encapsulated Security Payload implementiert. Beide Funktionen können sowohl im Transport als auch im Tunnel Mode eingesetzt und in Kombination verwendet werden.

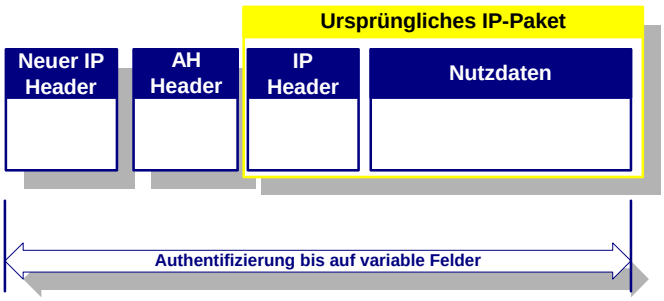
5.2.2.1 IP Authentication Header (AH)

Der IP Authentication Header (RFC 2402) wird nach dem IP-Header in das IP-Paket eingefügt und enthält u.a. die Informationen, mit denen ein IP-Paket authentifiziert werden kann. Mit dem AH werden die statischen Felder im IP-Header und alle nachfolgenden Nutzdaten mittels eines Hashs vor Veränderungen geschützt. Die variablen Felder dürfen nicht in den Hash einbezogen werden, da sie am Ziel andere Werte besitzen und der Hash somit niemals übereinstimmen würde. Der angewandte Algorithmus und der Schlüssel für die Authentifikation wird von der Security Association (siehe Kapitel 5.3) vereinbart.

Der AH kann im Transport und im Tunnel Mode eingesetzt werden. Im Transport Mode wird das ursprüngliche Paket und im Tunnel Mode das vom Security-Gateway erzeugte Paket mit dem AH versehen.



AH im Transport Mode



AH im Tunnel Mode

In beiden Fällen wird die Vertraulichkeit der Daten und der Verbindung mit dem AH aber nicht sichergestellt (keine Verschlüsselung).

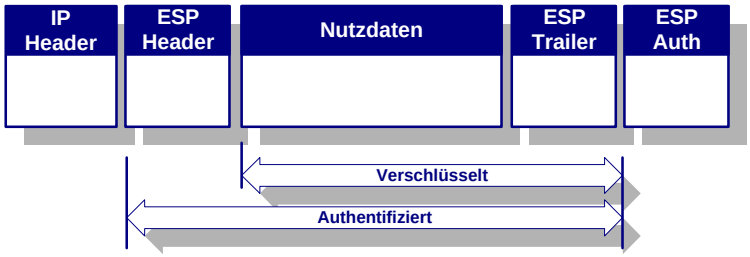
5.2.2.2 IP Encapsulated Security Payload (ESP)

Die IP Encapsulated Security Payload (RFC 2406) sorgt mittels Verschlüsselung für die Vertraulichkeit des Datenverkehrs. Der Algorithmus für die Verschlüsselung und der Schlüssel werden beim Verbindungsaufbau und während der Verbindung von der Security Association (siehe Kapitel 5.3) festgelegt. Alle Daten, die auf den IP-Header eines Pakets folgen, werden durch die verschlüsselten Daten ersetzt. Zwischen dem IP-Header und den verschlüsselten Daten wird ein ESP-Header eingefügt, und am Ende des Pakets folgt noch ein verschlüsselter ESP-Trailer. Optional kann ESP auch die Authentifikation übernehmen, was aber bei Verwendung des

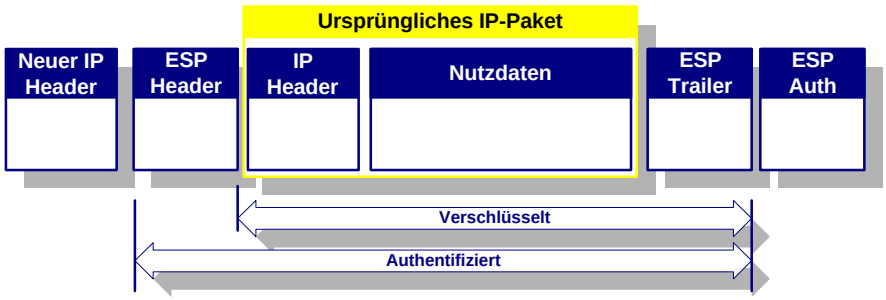
AH überflüssig ist. Zudem schützt die ESP-Authentifikation die statischen Informationen (Sender und Empfänger) im IP-Header nicht.

Auch ESP kann im Transport und im Tunnel Mode eingesetzt werden. Im Transport Mode wird das ursprüngliche Paket und im Tunnel Mode das vom Security-Gateway erzeugte Paket mit dem oben beschriebenen Verfahren modifiziert.

Beim Tunnel Mode wird das gesamte ursprüngliche Paket, das Bestandteil des vom Security-Gateway erzeugten Pakets ist, verschlüsselt. Der Sender und der Empfänger des ursprünglichen Pakets ist also nicht mehr ersichtlich und somit ist nicht nur die Vertraulichkeit der Daten sondern auch der Verbindung gewährleistet.



ESP im Transport Mode



ESP im Tunnel Mode

5.3 SECURITY ASSOCIATION (SA)

Eine Security Association (u.a. RFC 2408 und 2409) ist ein Vertrag zwischen den Endpunkten (Hosts oder Security-Gateways) einer sicheren Verbindung (z.B. IPsec). Der Vertrag, der für jede Verbindung neu ausgehandelt werden muss, beinhaltet die Parameter, welche für die sichere Kommunikation benötigt werden:

- Authentifikationsmechanismus
- Verschlüsselungsalgortihmus
- Hashalgorithmus
- Diverse Schlüssel für die Authentifikation und das Ver- und Entschlüsseln der Daten
- Gültigkeitsdauer der Schlüssel
- Zeit bis die SA erneuert werden muss

Beinahe alle Parameter der SA werden beim Verbindungsaufbau automatisch ausgehandelt. Um die Sicherheit der Verbindung zu erhöhen, werden die Schlüssel in bestimmten Zeitabständen auch während der Verbindung neu ausgehandelt. Der Ablauf, wie eine SA aufgebaut wird und die Parameter ausgehandelt werden, ist im folgenden Kapitel genauer beschrieben.

Weil nicht für alle Hosts die gleichen Parameter gelten dürfen, muss der Anwender in der Lage sein, bestimmte Parameter für jede Verbindung manuell konfigurieren zu können - so soll zum Beispiel ein bestimmter Host überhaupt keine SA mit dem eigenen Host eingehen dürfen. Wenn die Möglichkeit der manuellen, individuellen Konfiguration auf Hostebene gewährleistet ist, so wird IPsec auch der Forderung nach einer Zugriffskontrolle auf Benutzerebene (Hostebene) gerecht.

5.4 INTERNET KEY EXCHANGE (IKE)

Das Internet Key Exchange Protokoll (RFC 2409) beschreibt nicht nur den Austausch der Schlüssel einer IPsec SA - das Protokoll definiert den gesamten Aufbau der IPsec SA, also das Aushandeln und das Verteilen der Parameter, sowie die benötigten Mechanismen, um diesen Austausch zu sichern.

5.4.1 IKE Phase 1 - Main Mode

Das Austauschen der Parameter stellt an den IKE die grössten Anforderungen. Folgende Probleme gilt es zu lösen:

- Wie können die Parameter sicher über das Internet übertragen werden?

Dieses Problem lässt sich lösen, indem die Parameter verschlüsselt übertragen werden.

- Aber wie können zwei Hosts etwas verschlüsselt übertragen, wenn sie keine Informationen und folglich auch keine Schlüssel voneinander haben?

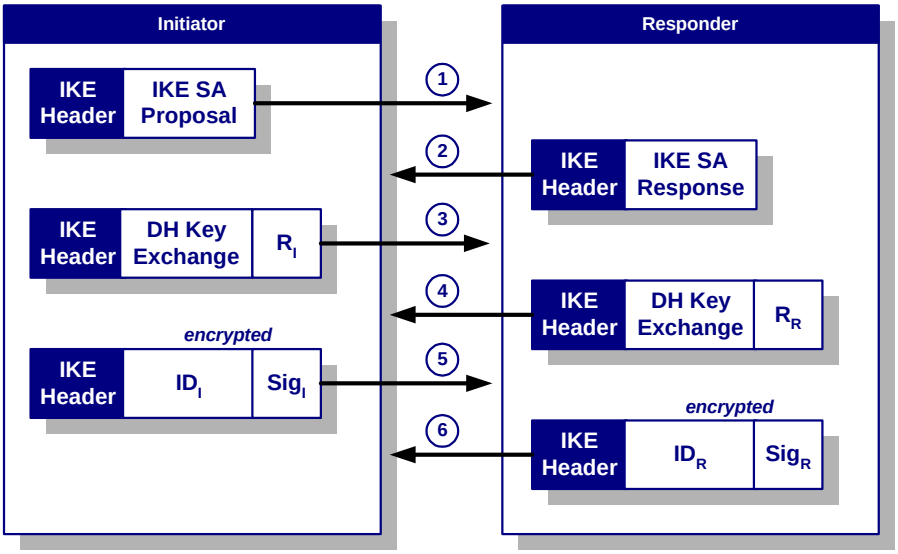
Dieses sehr heikle Problem kann mit dem Diffie-Hellman Key-Exchange Algorithmus [21] sehr einfach gelöst werden. Der Algorithmus ermöglicht das Erzeugen eines Secret Keys, der auf beiden Seiten (Hosts) bekannt ist, ohne dabei Informationen über das Internet zu übertragen, die es einem Angreifer ermöglichen würden, den Secret Key herauszufinden.

- Nun könnten die Parameter bereits gesichert übertragen werden, aber wer soll die Authentizität der Hosts garantieren?

Dieses Problem kann beim IKE u.a. mit Digital Signatures oder Preshared Keys gelöst werden. Ausser beim Verfahren mit Preshared Keys, bei welchem die beiden Host ein zuvor vereinbartes Geheimnis teilen, wird von den Hosts etwas benötigt, das nur der richtige Host kennt - dieses etwas ist meist der Private Key eines Schlüsselpaares.

Nun können sich die Hosts gegenseitig authentifizieren und besitzen auch einen gemeinsamen Schlüssel, mit dem sie die Parameter der SA sicher übertragen können.

Die oben beschriebenen Mechanismen zur Sicherung des Parameteraustauschs stellen selbst eine SA dar, die im IKE Protokoll Internet Security Association and Key Management Protocol (ISAKMP) SA genannt wird. Der hier beschriebene Vorgang bis die ISAKMP SA steht, wird IKE Phase 1 - Main Mode genannt.



IKE Phase 1 - Main Mode

Der oben dargestellte Ablauf soll kurz beschrieben werden:

- Schritt 1
  Der Initiator schickt dem Responder einen (oder mehrere) Vorschläge von SA-Konfigurationen. Dazu gehören Parameter wie der Authentifikationsmechanismus, der Verschlüsselungsalgorithmus der und Hashalgorithmus.
- Schritt 2
  Der Responder wählt einen Vorschlag aus und schickt diesen zurück.
- Schritt 3 und 4
  In diesen beiden Schritten findet der Diffie-Hellman Key-Exchange zwischen dem Initiator und dem Responder statt. Zusätzlich werden bestimmte Daten[1] der SA (in der Darstellung als R_I und R_R bezeichnet) übertragen.
- Schritt 5 und 6
  Der mit Diffie-Hellman Key-Exchange erzeugte Secret Key und die zusätzlich übertragenen Daten (R_I und R_R) werden benötigt, um einen weiteren Schlüssel zu erzeugen. Dieser Schlüssel wird als Initialisierungswert für die Bildung eines Hashs herangezogen. Dieser Hash wird über bestimmte Daten[1] der SA gebildet und anschliessend mit dem Private Key des jeweiligen Hosts signiert. Die entstandene Signatur (Sig_I und Sig_R) und die ID der Hosts (ID_I und ID_R) wird nun übertragen. Die Daten für die Bildung des Hashs müssen auf beiden Seiten bekannt sein, damit der andere Host den Hash resp. die Signatur überprüfen kann.

[1] Um welche Daten es sich handelt, wird hier nicht erklärt.

Weil beide Seiten im Besitz des in Schritt 3 und 4 erzeugten Secret Keys sind, werden ab Schritt 5 und 6 die Daten verschlüsselt übertragen.

Zusätzlich zur Signatur und der ID kann auch ein Zertifikat übertragen werden. Wird kein Zertifikat übertragen, müssen die Hosts in der Lage sein, anhand der empfangenen ID das Zertifikat resp. den Public Key des anderen Hosts zu ermitteln.

Nun können die Hosts mit dem Public Key die empfangene Signatur überprüfen und die Authentizität des anderen Hosts sicherstellen.

An dieser Stelle muss ein kleiner Nachteil dieses Authentifikationsmechanismus erwähnt werden: Wie aus der Darstellung ersichtlich ist, wird im Schritt 5 und 6 eine Identifikation der Hosts übertragen, obwohl die Authentizität der Hosts noch nicht sichergestellt ist - bis zum Schritt 6 könnte es sich beim Responder auch um einen Host ohne oder mit einer falschen Authentifikation handeln. Der Responder wäre folglich in der Lage, die Identität des Initiators zu ermitteln, ohne sich selbst ausweisen zu müssen.

5.4.2 IKE Phase 2 - Quick Mode

Nach dem Abschluss der Phase 1 (Main Mode) ist die Authentizität der Hosts und die Vertraulichkeit der Verbindung bereits gewährleistet. Aus diesem Grund ist die Phase 2 nicht mehr so aufwendig - der Quick Mode baut auf der zuvor erstellten ISAKMP SA auf.

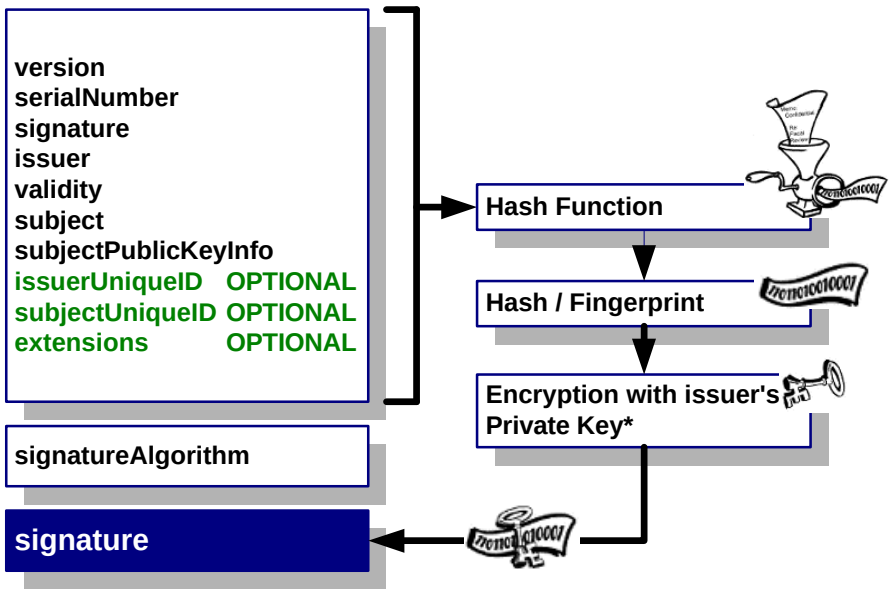
Erst der Quick Mode baut die eigentliche IPsec SA auf und handelt die verschiedenen Parameter für den AH, die ESP, den Authentifikations- und Verschlüsselungsmechanismus etc. aus.

Der Quick Mode kann während einer IPsec Verbindung wiederholt durchgeführt werden, um die Erneuerung der Schlüssel für die ISAKMP SA und die IPsec SA zu erzwingen. Damit die neuen Schlüssel in keiner Relation zu den vorher verwendeten Schlüsseln stehen, kann auch beim Quick Mode der Diffie-Hellman Key-Exchange Algorithmus angewandt werden, um neue Schlüssel zu erzeugen (Perfect Forward Secrecy).

5.5 X.509-ZERTIFIKAT

Beim X.509-Zertifikat geht es darum, einen Public Key seinem Inhaber zuzuordnen und dieser Zuordnung die benötigte Glaubwürdigkeit zu verleihen. Für die Glaubwürdigkeit, dass das Zertifikat echt ist, sorgt beim X.509-Zertifikat die Unterschrift einer sogenannten "Certification Authority" (CA), welche das Zertifikat ausstellt.

Ein X.509-Zertifikat ist folgendermassen aufgebaut:



Aufbau eines X.509-Zertifikats

Die wichtigsten Felder sollen kurz erläutert werden:

- signature, signatureAlgorithm
  Geben den Algorithmus an, mit dem das Zertifikat signiert wurde (beide Felder müssen den gleichen Algorithmus kennzeichnen)
- issuer
  Aussteller des Zertifikats (eine CA)
- validity
  Gültigkeitsbereich des Zertifikats (Datum von/bis)
- subject
  Distinguished Name des Inhabers des Zertifikats
- subjectPublicKeyInfo
  Der eigentliche Public Key

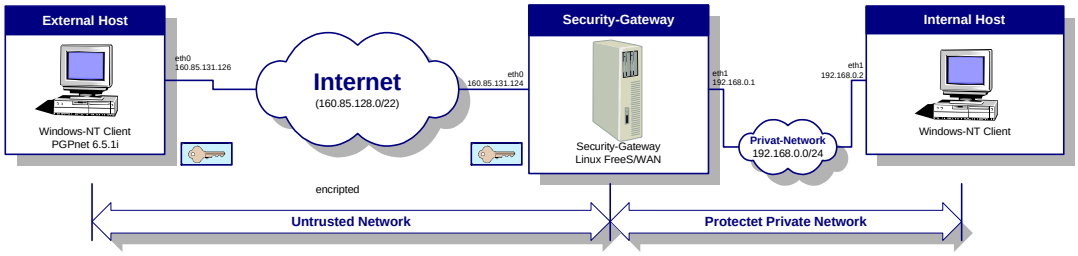
Die CA bildet nun einen Hash über diese Daten und verschlüsselt den Hash mit ihrem Secret Key. Der verschlüsselte Hash bildet die Signatur (Unterschrift) der CA und wird dem Zertifikat hinzugefügt. Die Signatur bestätigt die Richtigkeit der Daten.

Um die Gültigkeit eines Zertifikats zu überprüfen, muss die Signatur mit dem Public Key der CA entschlüsselt und mit dem Hash verglichen werden.

Eine CA kann ihr Zertifikat ebenfalls unterschreiben lassen. Dadurch entsteht eine Zertifizierungskette, an dessen oberster Stelle sich die Root CA befindet. Die Root CA unterschreibt ihr Zertifikat selbst. Jeder muss dieser Root CA vertrauen.

6 ENTWICKLUNGS- UND TESTUMGEBUNG

6.1 TOPOLOGIE



Topologie

Unsere Testumgebung gliedert sich im wesentlichen in zwei Teile, und zwar in ein geschütztes privates Netzwerk (192.168.0.0/24), welches nur über den Security-Gateway erreichbar ist und dem Internet, welchem nicht getraut wird. In unserem Falle ist dies das Schulnetz (160.85.128.0/22).

6.2 PGPNET-TO-FREES/WAN

Unser Ziel ist es nun von einem externen Windows NT Host (160.85.131.126) über das Internet via Security-Gateway (160.85.131.124 / 192.168.0.1) auf unseren internen Host (192.168.0.2) zuzugreifen. Dabei verwenden wir PGPnet auf dem externen Host, welcher uns eine sichere Kommunikation mit dem internen Host erlaubt. Es wird eine verschlüsselte Verbindung zwischen dem Security-Gateway auf FreeS/WAN-Basis und dem externen Host aufgebaut. Dabei übernimmt der Security-Gateway alle Sicherheitsfunktionen, was bedeutet, dass die internen Hosts keine zusätzliche Software benötigen. Die Verbindung bleibt also für die Hosts im geschützten privaten Netzwerk transparent.

In den folgenden Kapiteln werden wir genauer betrachten, wie eine solche Verbindung konfiguriert werden kann.

7 GRUNDKONFIGURATION

In diesem Kapitel wird die Grundkonfiguration von PGPnet und FreeS/WAN vorgenommen. Eine funktionierende Verbindung wird erst im Kapitel 9 erstellt.

7.1 PGPNET

Wie bereits erwähnt, wollen wir den einen Host unserer VPN-Verbindung mit PGPnet betreiben.

Um das korrekte Arbeiten unseres PGPnet-Clients zu testen, bevor wir mit dem FreeS/WAN Security-Gateway kommunizierten, installierten wir einen zweiten PGPnet-Client, mit welchem wir eine lauffähige PGPnet-zu-PGPnet-Verbindung erzeugten.

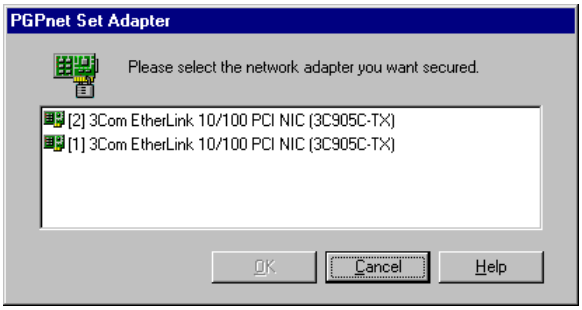
Für detaillierte Informationen zu PGPnet sollte die Online-Hilfe oder die deutsche Anleitung unter [17] verwendet werden.

7.1.1 Installation

PGPnet kann sehr einfach mit einem Setup-Programm installiert werden.

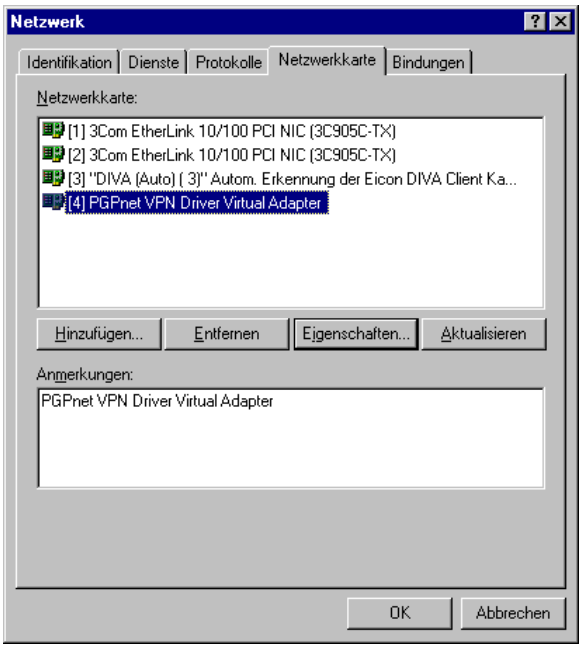
Nach einem Neustart muss die erste Entscheidung getroffen werden. PGPnet fragt, welche Netzwerkkarten gesichert werden sollen. Für unser Beispiel muss natürlich diejenige Netzwerkkarte ausgewählt werden, welche mit dem VPN-Partner verbunden ist.

Die Netzwerkkarten können auch im Nachhinein im Startmenu unter `Programme / PGP / Set Adapter` aktiviert bzw. deaktiviert werden.



PGPnet Set Adapter

In der Netzwerkkonfiguration von Windows NT finden wir anschliessend den virtuellen Adapter mit dem PGPnet VPN-Treiber.

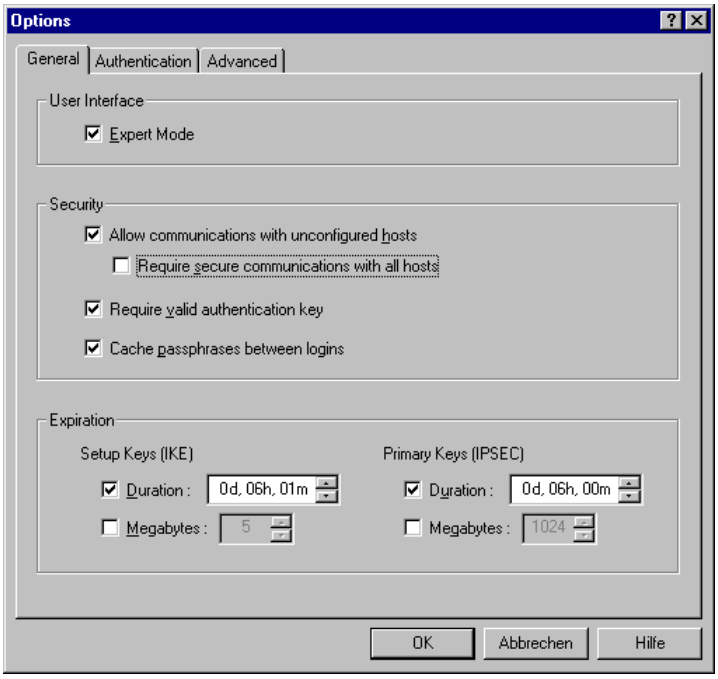


Netzwerkkonfiguration von Windows NT

7.1.2 General Option

Beim ersten Start von PGPnet mit `Start / Programme / PGP / PGPnet` erscheint ein Konfigurationsassistent. Weil wir die Einstellungen manuell vornehmen, kann der Assistent abgebrochen werden.

Wählen Sie den Befehl `View / Options` und nehmen Sie folgende Konfiguration vor:

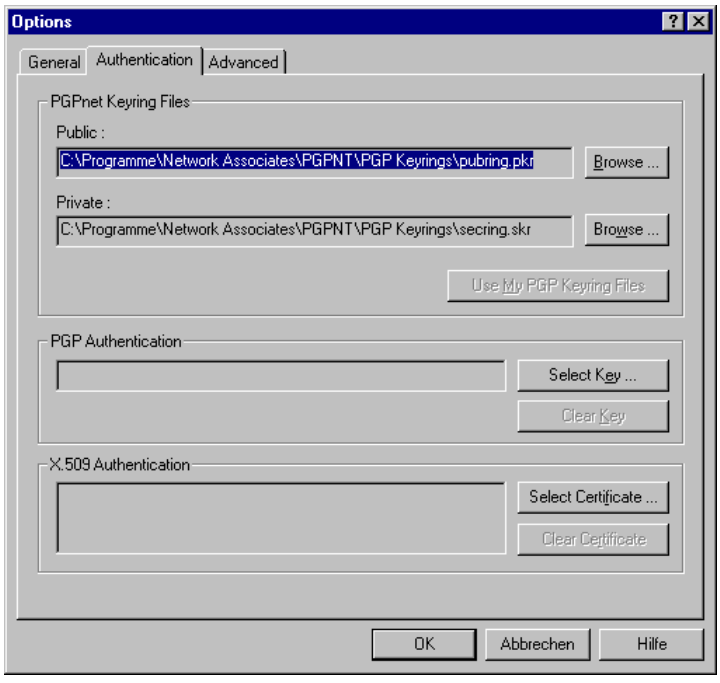


PGPnet Options: General

- PGPnet soll im `Expert Mode` ausgeführt werden (keine Assistenten).
- Wenn `Allow communications with unconfigured hosts` aktiviert ist, können auch Verbindungen zu Rechnern erstellt werden, die in PGPnet nicht konfiguriert sind. Dadurch muss nicht jeder Kommunikationspartner in die Hosts-Liste von PGP aufgenommen werden.
- Wird `Require secure communications with all hosts` aktiviert, so können nur sichere Verbindungen zu anderen Hosts erstellt werden. Wird die Option deaktiviert, kann ein Rechner zwar eine verschlüsselte Verbindung z.B. ins Firmennetz erstellen aber gleichzeitig auch eine unverschlüsselte Verbindung zu einem WWW-Server aufbauen.
- Ist die Option `Require valid authentication key` aktiviert, so akzeptiert PGPnet die Schlüssel von anderen Rechnern nur dann, wenn sie im lokalen Keyring als valid gekennzeichnet sind.
- Unter `Expiration` kann festgelegt werden, wie lange die Keys für die ISAKMP SA und IPsec SA gültig sind. Damit PGPnet mit FreeS/WAN kompatibel ist, muss die Dauer auf 6 Stunden gesetzt werden. Alternativ könnte auch eine Datenmenge angegeben werden, nach der die Keys erneuert werden.

7.1.3 Authentication Options

Wählen Sie die Registerkarte `Authentication` im `Options`-Dialog aus.



PGPnet Options: Authentication

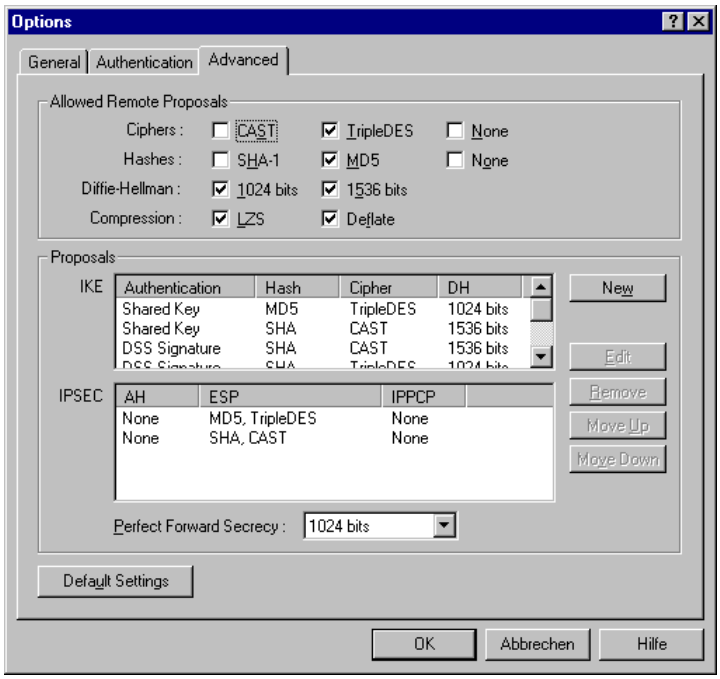
Unter `PGPnet Keyring Files` können die Public- und Secret-Keyringdateien für PGPnet bestimmt werden, welche die Public und Secret Keys enthalten. Diese Einstellungen können beibehalten werden.

Will man PGP-Keys zur VPN-Authentifikation einsetzen, wird unter PGP `Authentication` ein PGP-Key aus den PGPnet Keyring-Files bestimmt, der zur Authentifikation benutz wird. Beide Kommunikationspartner müssen hier den gleichen Schlüsseltyp angeben und verwenden, also in unserem Falle RSA.

Soll `X.509 Authentication` als Authentifikation genutzt werden, so kann dies hier gewählt werden. Beide Kommunikationspartner müssen die gleiche Root CA verwenden und das gleiche, von ihnen signierte und mit grösstem Trustlevel versehenen Root CA Zertifikat in ihrem PGPnet Pubring haben.

7.1.4 Advanced Options

Wählen Sie die Registerkarte `Advanced` im `Options`-Dialog aus.



PGPnet Options: Advanced

Unter `Allowed Remote Proposals` wird festgelegt, welche Verschlüsselungs-, Hash- und Authentifizierungsalgorithmen akzeptiert werden sollen. Weiter kann festgelegt werden, welche Keylängen des Diffie-Hellman Keys akzeptiert werden.

Der Abschnitt `Proposals` beinhaltet die eigenen Vorschläge, die man dem Partner zur Verschlüsselung und Authentifizierung (ISAKMP SA und IPsec SA) anbietet. Dabei kann man verschiedene Kombinationen aus dem zu verwendenden Verschlüsselungs- und Hashalgorithmus, der Datenkompressionsart und Keylänge des Diffie-Hellmann Keys definieren.

Damit PGPnet mit FreeS/WAN (und umgekehrt) eine Verbindung aufbauen kann, müssen die Einstellungen wie im Bild oben gezeigt, eingestellt werden.

7.2 FREES/WAN

Nun soll die Grundkonfiguration für unseren Security-Gateway auf FreeS/WAN-Basis erstellt werden.

7.2.1 Installation

Die Linux-Distribution von SuSE (ab Version 6.3) enthält bereits das FreeS/WAN-Paket und im mitgelieferten Kernel ist IPsec bereits aktiviert, dadurch muss der Kernel nicht neu kompiliert werden.

Wichtig für die Hosts im Subnet ist noch, dass das IP-Forwarding auf dem Security-Gateway eingeschaltet ist. Dieser Eintrag kann in der Datei `/usr/etc/rc.config` oder im Yast gemacht werden (`IP_FORWARD="yes"`).

7.2.2 FreeS/WAN-Patch für PGP-Key-Support

Damit FreeS/WAN PGP-Keys unterstützt, muss der Patch "PGP Key support by Kai Martius for FreeS/WAN Version 1.3" installiert werden. Damit das Update problemlos funktioniert, muss FreeS/WAN 1.3 (mit dem Source-Code) installiert sein (ist ab SuSE Linux 6.3 enthalten).

Der PGP-Key-Patch ist im X.509-Zertifikat-Patch bereits integriert. Wenn anschliessend der X.509-Zertifikat-Patch installiert werden soll, empfiehlt es sich, den PGP-Key-Patch nicht zu installieren.

7.2.2.1 Main.c patchen

- Wechseln Sie ins Verzeichnis `/usr/src/packages/SOURCES/freeswan-1.3/pluto` und kopieren das File `pluto.diff` des Patchs in dieses Verzeichnis
- Rufen Sie `patch -p1 < pluto.diff` auf
- Pluto mit dem Befehl `make` kompilieren
- Datei `pluto` nach `/usr/lib/ipsec` kopieren

7.2.2.2 Keyextractor kompilieren

Keyextractor ist ein Hilfsprogramm, mit dem ein Public oder Private Key in ein für FreeS/WAN lesbares Format aus einem PGP-Keyring extrahiert werden kann.

- Downloaden Sie "Peter Gutman's Crypto Library" [7]
- In der Datei `keymgmt/pgp_keys.c` muss die Funktion `readKey` sichtbar gemacht werden (`static`-Keyword entfernen)
- Die Quelltextdateien liegen im DOS-Format vor (Zeilenumbrüche als CR/LF codiert) und müssen ins Unix-Format umgewandelt werden. Verwenden Sie dazu das Kommando `dos2unix < [Quelldatei] > [Zieldatei]` oder `d2u [Quelldatei]` (wobei beim zweiten

Kommando auch Wildchars angegeben werden können)
Achtung, nur die `*.c` und `*.h`-Dateien dürfen gewandelt werden!

- Nun kann die Library mit `make` kompiliert werden.
- Jetzt müssen die beiden Dateien `libcrypt.a` und `libgmp.a` in das Sourceverzeichnis des Keyextractors kopiert werden.
- Das Makefile des Keyextractors muss noch angepasst werden, damit auf das richtige Verzeichnis der Crypto-Library verwiesen wird (Variable `CRYPTLIB`)
- Nun kann der Keyextractor mit `make` compiliert werden.

7.2.3 FreeS/WAN-Patch für X.509-Support

Damit FreeS/WAN auch X.509-Zertifikate unterstützt, muss der von uns erstellte Patch installiert werden. Dieser Patch setzt FreeS/WAN Version 1.3 (ungepatched) voraus, der oben beschriebene Patch für PGP-Keys ist bereits integriert.

- Wechseln Sie in das Verzeichnis
  `/usr/src/packages/SOURCES/freeswan-1.3/pluto`
  und kopieren das File `PlutoX509.diff` des Patchs in dieses Verzeichnis
- Rufen Sie `patch -p1 < PlutoX509.diff` auf
- Pluto mit dem Befehl `make` compilieren
- Datei `pluto` nach `/usr/lib/ipsec` kopieren
- Der Keyextractor ist für die Konfiguration der X.509-Zertifikate ebenfalls nötig. Details zur Kompilation sind oben beschrieben.

7.2.4 Logdateien

Sämtliche Aktionen von FreeS/WAN können geloggt werden. Dazu müssen in der Datei `ipsec.conf` die Variablen `plutodebug` bzw. `klipsdebug` entsprechend gesetzt werden (siehe Kapitel 9.1.1.1).

Die Logdatei kann mit folgendem Befehl in eine Datei geschrieben werden:

`ipsec barf > barf.txt` (barf=brechen/kotzen)

8 ANPASSEN DES QUELLTEXTS FÜR DIE VERWENDUNG VON X.509-ZERTIFIKATEN

8.1 IDENTIFIKATIONSTYP

Die Authentifikation mit einem X.509-Zertifikat ist sehr ähnlich wie die Authentifikation mit einem PGP-Key. Der Unterschied liegt darin, dass zur Identifikation nicht der Fingerprint des PGP-Schlüssels, sondern der Distinguished Name des Subjects im X.509-Zertifikat verwendet wird. Weil die Version 1.3 von FreeS/WAN keine X.509-Zertifikate unterstützt, mussten wir den Quelltext des IKE-Dämons Pluto anpassen.

Bei der Authentifizierung schicken sich die beiden Hosts gegenseitig ihre Identifikation. Mit dieser eindeutigen Identifikation kann ein Host sein Gegenüber identifizieren und die gewünschten Einstellungen für diese Verbindung vornehmen.

FreeS/WAN verwendet zur Auswahl der richtigen Verbindung die Identifikation im File `/etc/ipsec.conf`. Pluto liest dieses File beim Starten von `ipsec` ein und ermittelt für jede aufgeführte Verbindung den Identifikationstyp und die Identifikation der beiden Hosts anhand der Variablen `leftid` und `rightid`.

Die ersten zwei Zeichen der Variablen geben dabei den Typ der Identifikation an. Für die X.509 Zertifikate muss Pluto zusätzlich den Identifikationstyp ID_DER_ASN1_DN (siehe RFC 2407) erkennen und verarbeiten können. Dieser Typ steht für einen binär DER-codierten ASN.1 X.500 Distinguished Name, wie er beim Subject des X.509-Zertifikats vorkommt. Damit Pluto eine Identifikation vom Typ ID_DER_ASN1_DN im File `/etc/ipsec.conf` als solche erkennen kann, mussten wir eine neue Typenkennzeichnung für die Variablen `leftid` und `rightid` finden und diese Pluto bekannt machen. Wir haben uns für die Zeichen "@~" entschieden (ähnlich der Typenkennzeichnung ID_KEY_ID mit den Zeichen "@#"). Somit lautet eine Identifikation vom Typ ID_DER_ASN1_DN im File `/etc/ipsec.conf` beispielsweise: `leftid=@~305f31...`
Im Quelltext von Pluto musste dazu die Funktion `const char *atoid(char *src, struct id *id)` im File `freeswan-1.3/pluto/id.c` angepasst werden.

Nun kann Pluto zwar Identifikationen des Typs ID_DER_ASN1_DN im File `/etc/ipsec.conf` als solche erkennen, aber der vom anderen Host empfangene Identifikationstyp ID_DER_ASN1_DN wird immer noch verworfen, weil Pluto ihn nicht verarbeiten kann. Um dieses Problem zu beheben, mussten wir die Verarbeitungsschritte in Pluto für den Identifikationstyp ID_DER_ASN1_DN definieren. Weil die Authentifikation mit einem PGP-Key und einem X.509-Zertifikat gleich abläuft (abgesehen

von der Überprüfung des übermittelten Zertifikats), haben wir den Quelltext, der für die Authentifikation mit einem PGP-Key nötig ist, übernommen. Wir haben bei den Abfragen lediglich den Identifikationstyp ID_KEY_ID durch ID_DER_ASN1_DN ersetzt. Die Quelltextänderungen betreffen mehrere Files im Verzeichnis `freeswan-1.3/pluto` und werden hier nicht einzeln aufgeführt (siehe dazu das diff-File).

8.2 ZERTIFIKATSTYP

Neben dem Identifikationstyp wird bei einer X.509 VPN-Verbindung auch ein Zertifikatstyp übertragen. Dabei stellten wir fest, dass der übermittelte Zertifikatstyp nicht mit dem tatsächlich Typ des Zertifikats übereinstimmen muss, d.h. PGPnet vergleicht den Zertifikatstyp nicht mit dem Zertifikat. Die Vermutung liegt nahe, dass PGPnet weder den Zertifikatstyp noch das Zertifikat vom anderen Host überprüft. Die Vermutung konnten wir bestätigen, indem wir das von Pluto übermittelte Zertifikat im File `/etc/pgpcert` abänderten und die Verbindung trotzdem zustande kam. Wir vermuten darum folgendes: PGPnet sucht in seinem lokalen Keyring nach einem X.509-Zertifikat, dessen Distinguished Name (Subject) der vom anderen Host empfangenen Identifikation entspricht. Wenn das Zertifikat lokal gefunden und als gültig erkannt wurde, verwendet PGPnet den Public Key von diesem Zertifikat und kümmert sich nicht um das übermittelte Zertifikat. Kann das Zertifikat nicht lokal gefunden werden, kommt keine Verbindung zustande.

9 KONFIGURATION DER VERSCHIEDENEN AUTHENTISIERUNGSMETHODEN

Wir haben drei Tests durchgeführt, um einen PGPnet-Client mit FreeS/WAN zu verbinden. Bei diesen drei Tests verwendeten wir verschiedene Authentisierungsmethoden:

- Preshared Secrets
- PGP-Keys
- X.509-Zertifikate

In den folgenden Unterkapiteln werden die Konfigurationen für die Authentisierungsmethoden genauer beschrieben, wobei jeweils nur die Unterschiede zur vorherigen Methode erklärt werden.

9.1 PRESHARED SECRETS

Als erstes haben wir die VPN-Verbindung mittels eines Preshared Secrets aufgebaut. Die beiden Clients authentisieren sich in diesem Fall mit einem Passwort, das beiden bekannt ist.

9.1.1 Konfiguration FreeS/WAN

9.1.1.1 ipsec.conf

Die Datei `/etc/ipsec.conf` enthält die Konfiguration für den FreeS/WAN Security-Gateway.

In einem ersten Teil, dem `config setup`, werden Basiseinstellung gesetzt. Hier wird bestimmt, welche Interfaces für die Kommunikation verwendet werden, ob Informationen gelogged werden sollen und wie sich Pluto verhalten soll.

In einem zweiten Teil werden die verbindungsspezifischen Vereinbarungen getroffen. Dabei kann für jede einzelne Verbindung eine eigene Vereinbarung konfiguriert werden.

```
# /etc/ipsec.conf - FreeS/WAN IPSEC configuration file
# basic configuration
config setup
        # THIS SETTING MUST BE CORRECT or almost nothing will work;
        # %defaultroute is okay for most simple cases.
        interfaces=%defaultroute
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        klipsdebug=none
        plutodebug=all
        # Use auto= parameters in conn descriptions to control startup
        # actions.
        plutoload=%search
        plutostart=%search
```

Beschreibung zu den hier gesetzten Variablen:

- interfaces: Beschreibt die Zuordnung der physikalischen zu den virtuellen (gesicherten) IPsec-Interfaces. Im Normalfall genügt hier schon der Eintrag `%defaultroute`, wobei das konfigurierte Standardinterface verwendet wird. Wird das Interface explizit angegeben, so lautet der Eintrag z.B. `interfaces="ipsec0=eth1".` Sollen mehrere Interfaces angegeben werden, so ist zu beachten, dass diese durch ein Leerzeichen voneinander getrennt sind.
- klipsdebug: Soll der KLIPS Debugging Output gelogged werden, so wird hier `all` angegeben. Andernfall kann man diese Variable auf `none` setzen.
- plutodebug: Hier kann der Pluto Debugging Output mit `all` gelogged werden. Soll dies nicht geschehen, so bleibt der Eintrag leer oder wird mit `none` gekennzeichnet. Für unsere Testzwecke setzten wir diese Variable auf `all` für komplettes Logging.
- plutoload: Hier kann dem Pluto mitgeteilt werden, welche Verbindungen er zur Startzeit laden soll. Wird die Variable, wie in diesem Falle gezeigt, auf `%search` gesetzt, so werden alle connections mit den Einträgen `auto=add` und `auto=start` geladen.
- plutostart: Wenn hier `plutostart="%search"` eingetragen wird, startet Pluto automatisch die Verbindungen, welche mit `auto="start"` gesetzt wurden.

Nun definieren wir unterschiedliche Verbindungen. Da wir den Gateway, wie auch das dahinter liegende Subnetz erreichen möchten, müssen zwei Verbindungen deklariert werden:

```
# defaults for subsequent connection descriptions
conn %default
        # How persistent to be in (re)keying negotiations (0 means very).
        keyingtries=0
        # Parameters for manual-keying testing (DON'T USE OPERATIONALLY).

# connection
conn secure_gw
        authby=secret
        # Left security gateway, subnet behind it, next hop toward it.
        left=160.85.131.126
        leftsubnet=160.85.131.126/32
        leftid=160.85.131.126
        # Right security gateway, subnet behind it, next hop toward it.
        right=160.85.131.124
        rightsubnet=160.85.131.124/32
        rightid=160.85.131.124
        # Authorize this connection, but don't actually start it, at startup.
        auto=add

conn secure_subnet
        authby=secret
        # Left security gateway, subnet behind it, next hop toward it.
        left=160.85.131.126
        leftsubnet=160.85.131.126/32
        leftid=160.85.131.126
        # Right security gateway, subnet behind it, next hop toward it.
        right=160.85.131.124
        rightsubnet=192.168.0.0/24
        rightid=160.85.131.124
```

```
# Authorize this connection, but don't actually start it, at startup.
auto=add
```

- authby: Dies ist die Authentifikationsart der beiden Kommunikationspartner. Mögliche Einträge sind `secret` für Preshared Secrets oder `rsasig` für digitale RSA-Unterschriften. In unserem Fall nehmen wir natürlich `secret`, da unsere beiden Partner in diesem Versuch ein gemeinsames Passwort zur Authentifikation verwenden sollen.
- left/right: Wir definieren hier eine gültige IP-Adresse des linken bzw. rechten Hosts oder Security-Gateways. In unserem Falle ist dies der PGPnet-Client mit der IP-Adresse `160.85.131.126` und der FreeS/WAN-Gateway mit der IP-Adresse `160.85.131.124`.
- leftsubnet/rightsubnet:

  Die Subnets kennzeichnen wir in der `network/netmask` Notation. Da hinter dem PGPnet-Client kein Subnet vorhanden ist, verwenden wir für die `netmask` den Wert 32.
- leftid/rightid: Dieser Eintrag legt die Identifikation für die Authentifizierung fest. Bei der Preshared-Authentisierung wird der Kommunikationspartner anhand seiner IP-Adresse Identifiziert. Es kann anstelle der IP-Adresse auch ein gültiger Domainname mit einem führenden "@" eingetragen werden.
- auto: Wir setzen die Variable `auto` auf `add` und teilen damit Pluto mit, dass er zur Startzeit diese Verbindung starten soll. Dies bedingt, dass die Variable `plutoload` auf `%search` gesetzt wurde, damit Pluto beim Start nach den Verbindungen mit diesem Eintrag sucht und diese startet.

9.1.1.2 ipsec.secrets

Die Datei `/etc/ipsec.secrets` enthält die Konfigurationsparameter von FreeS/WAN, die geheim gehalten werden müssen. Bei einer Authentisierung mit Preshared Keys wird hier nur das Passwort konfiguriert. In unserem Beispiel heisst das Passwort "test".

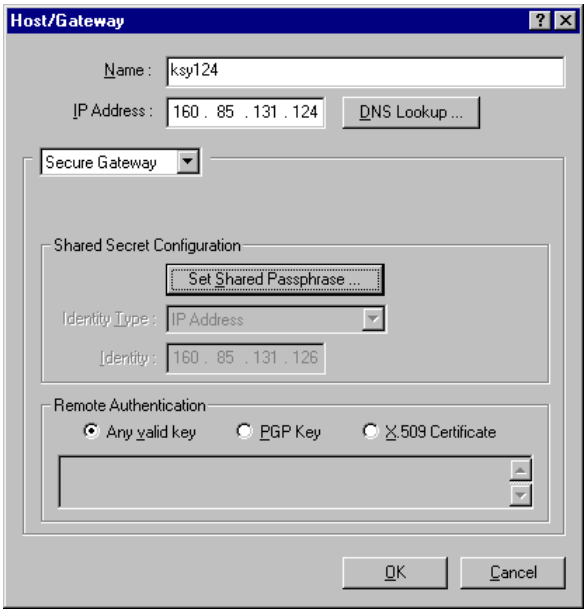
```
# This file holds shared secrets which are currently the only inter-Pluto
# authentication mechanism.  See ipsec_pluto(8) manpage.  Each secret is
# (oversimplifying slightly) for one pair of negotiating hosts.

# The shared secrets are arbitrary character strings and should be both
# long and hard to guess.

# Note that all secrets must now be enclosed in quotes, even if they have
# no white space inside them.

160.85.131.126 160.85.131.124 "test"
```

Weil diese Datei geheime Informationen enthält, sollte sie nur durch den FreeS/WAN Administrator zugänglich sein und für alle anderen lese- und schreibgeschützt bleiben.

9.1.2 Konfiguration PGPnet

Folgende Konfigurationsschritte sind nötig:

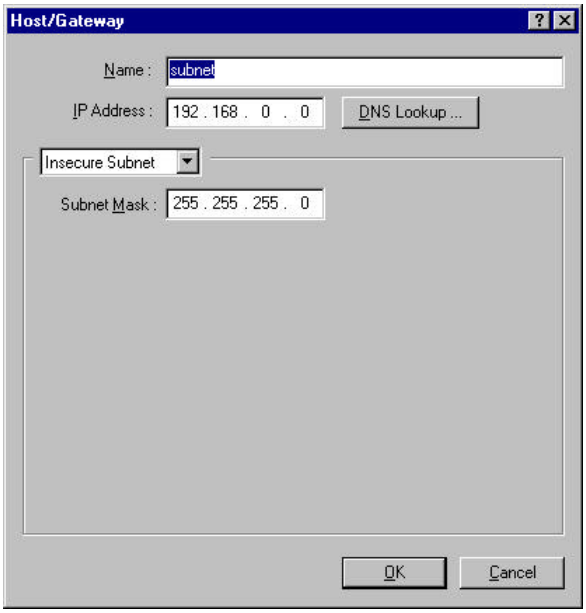
- Grundkonfiguration von PGPnet wie in Kapitel 7.1 beschrieben, durchführen
- Hosts-Fenster von PGPnet öffnen und `Add` wählen, um den Security-Gateway zu konfigurieren
- Konfiguration gemäss folgendem Fenster vornehmen (analog `ipsec.conf` von FreeS/WAN):



PGPnet Host/Gateway: Secure Gateway (mit Shared Secret)

Zusätzlich muss noch das Preshared Secret eingegeben werden. Dazu muss auf die Schaltfläche `Set Shared Passphrase` geklickt werden und das Passwort (in unserem Beispiel "test") eingegeben werden.

- Wiederum auf Add klicken, um Verbindung für das Subnet zu erstellen:



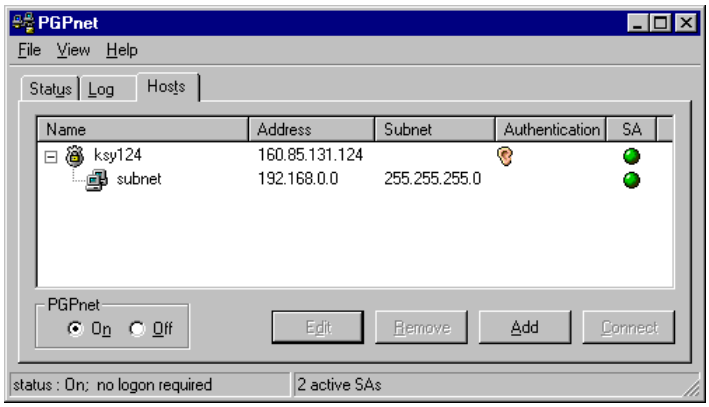
PGPnet Host/Gateway: Insecure Subnet

Damit ist auch die Konfiguration von PGPnet abgeschlossen.

9.1.3 Verbindung testen

Um die Verbindung zu testen, muss Pluto IPsec dem Befehl `ipsec setup start` gestartet werden. PGPnet kann mit einem Klick auf `On` im Hosts-Fenster aktiviert werden.

Eine Verbindung kann aufgebaut werden, indem die Gegenstelle angepingt wird. Beispielsweise mit `ping 160.85.131.124` auf dem PGPnet-Client, um den Gateway anzupingen oder `ping 192.168.0.1`, um die Verbindung in das Subnetz zu testen. In Gegenrichtung kann z.B. vom Gateway aus der PGPnet-Client mit `ping 160.85.131.126` erreicht werden.



PGPnet Hosts (mit aufgebauten SAs)

9.2 PGP-KEYS

Als nächstes soll eine Authentifikation mit PGP-Keys durchgeführt werden. Dadurch ist es nicht mehr nötig, dass den beiden Gegenstellen ein gemeinsames Passwort bekannt ist.

Voraussetzung ist, dass die Grundkonfiguration von PGPnet wie in Kapitel 7.1 beschrieben, und die Konfiguration für Preshared-Authentifikation wie in Kapitel 9.1.2 beschrieben, durchgeführt wurde. Es werden hier nur noch die Unterschiede gegenüber Preshared Keys erklärt. Weiter muss der Pluto-Patch für PGP-Keys oder X.509-Zertifikate installiert werden, wie dies in Kapitel 7.2.2 bzw. 7.2.3 beschrieben ist.

9.2.1 PGP-Keys erstellen

Zuerst benötigen wir für die beiden Kommunikationspartner (left und right) jeweils ein PGP-Schlüsselpaar, welches mit PGPnet bequem erzeugt werden können.

- Starten Sie das Programm `Start / Programme / PGP / PGPkeys`
- Wählen Sie den Befehl `Keys / New Key...`
  Für unser Beispiel haben wir folgende Informationen für die beiden Schlüsselpaare verwendet:

Parameter	left	right
E-Mail:	left@zhwin.ch	right@zhwin.ch
Key Pair Type:	RSA	RSA
Key Pair Size:	1024bit	1024bit
Expires:	Key Pair never expires	Key Pair never expires
Passphrase:	test	test

9.2.2 Konfiguration FreeS/WAN

9.2.2.1 ipsec.config

Damit FreeS/WAN mit PGP-Keys arbeitet, müssen einige Änderungen an der Konfigurationsdatei `ipsec.config` gemacht werden.

- Public Key von left einfügen:
  `keyextractor left out.txt pubring.pkr`
  `pubring.pkr` ist der Public-Keyring von PGP (befinden sich bei einer Standardinstallation im Verzeichnis
  `C:\Programme\Network Associates\PGPNT\PGP Keyrings`)
  Keyextractor zeigt eine Fehlermeldung an, dass der Secret Key nicht geöffnet werden konnte. Da FreeS/WAN den Private Key von left nicht braucht (oder nicht brauchen darf), hat dies keine negativen Auswirkungen.
  Der Inhalt der Datei `out.txt` kann nun unter der Variable `leftrsasigkey` eingefügt werden.
- `rightrsasigkey` kann auf `0x00` gesetzt werden, damit Pluto sein eigenes Zertifikat aus der Datei `/etc/pgpcert.pgp` liest.

- Für die Variablen `leftid` und `rightid` müssen die Fingerprints der Public Keys verwendet werden (können in PGPnet ermittelt werden).
- `authby` auf `rsasig` setzen

Anschliessend sollte `ipsec.config` in etwa so aussehen:

```
# /etc/ipsec.conf - FreeS/WAN IPSEC configuration file

# More elaborate and more varied sample configurations can be found
# in doc/examples.

# basic configuration
config setup
        # THIS SETTING MUST BE CORRECT or almost nothing will work;
        # %defaultroute is okay for most simple cases.
        interfaces=%defaultroute
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        klipsdebug=none
        plutodebug=all
        # Use auto= parameters in conn descriptions to control
        # startup actions.
        plutoload=%search
        plutostart=%search

# defaults for subsequent connection descriptions
conn %default
        # How persistent to be in (re)keying negotiations (0 means very).
        keyingtries=0
        # Parameters for manual-keying testing (DON'T USE OPERATIONALLY).
        spi=0x200

# connection
conn secure_gw
        authby=rsasig
        # Left security gateway, subnet behind it, next hop toward it.
        left=160.85.131.126
        leftsubnet=160.85.131.126/32
        leftid=@#043BD47747F9D82EF11FC9156EAC590B
        leftrsasigkey=0x0111dc279bbe9d96667061c6eafa8aae747a67508a66494be526
                      b35c4e903ea011ce52dd8d40625da1b437ac5dafe5c45b5ed7d479
                      ce735babc5ad694c5f48b3a290d4442640b4fd9bccfd4e8f4530a8
                      95205fa89742d297098ff8221d8205c5ef5821e5a827a3e2e60bb3
                      b66bdc9381afcefeca13d79ca4e5fc5018d39d61019641
        # Right security gateway, subnet behind it, next hop toward it.
        right=160.85.131.124
        rightsubnet=160.85.131.124/32
        rightid=@#6B823E3EE3CDE659744BB4D23A7253CB
        rightrsasigkey=0x00
        # Authorize this connection, but don't actually start it, at startup.
        auto=add

conn secure_subnet
        authby=rsasig
        # Left security gateway, subnet behind it, next hop toward it.
        left=160.85.131.126
        leftsubnet=160.85.131.126/32
        leftid=@#043BD47747F9D82EF11FC9156EAC590B
        leftrsasigkey=0x0111dc279bbe9d96667061c6eafa8aae747a67508a66494be526
                      b35c4e903ea011ce52dd8d40625da1b437ac5dafe5c45b5ed7d479
                      ce735babc5ad694c5f48b3a290d4442640b4fd9bccfd4e8f4530a8
                      95205fa89742d297098ff8221d8205c5ef5821e5a827a3e2e60bb3
                      b66bdc9381afcefeca13d79ca4e5fc5018d39d61019641
        # Right security gateway, subnet behind it, next hop toward it.
        right=160.85.131.124
        rightsubnet=192.168.0.0/24
        rightid=@#6B823E3EE3CDE659744BB4D23A7253CB
        rightrsasigkey=0x00
        # Authorize this connection, but don't actually start it, at startup.
        auto=add
```

9.2.2.2 ipsec.secrets

Als nächstes kann der Secret Key von right in FreeS/WAN importiert werden:

- `keyextractor right out.txt pubring.pkr secring.skr`
  `pubring.pkr` bzw. `secring.skr` ist der Public-Keyring bzw. Secret-Keyring von PGP (befinden sich bei einer Standardinstallation im Verzeichnis `C:\Programme\Network Associates\PGPNT\PGP Keyrings`
  Anschliessend steht das Resultat steht in der Datei `out.txt`.
- Extrahierte Secret Key-Informationen in die Datei `ipsec.secrets` einfügen.

`ipsec.secrets` sollte nun folgendermassen aufgebaut sein:

```
# This file holds shared secrets which are currently the only inter-Pluto
# authentication mechanism.  See ipsec_pluto(8) manpage.  Each secret is
# (oversimplifying slightly) for one pair of negotiating hosts.

# The shared secrets are arbitrary character strings and should be both
# long and hard to guess.

# Note that all secrets must now be enclosed in quotes, even if they have
# no white space inside them.


: RSA {
        Modulus:
            0xc5c7b0f4b341e8bb4a680a7df7e89503d3dc8a2490687a01d8c3a155bcb07f
            116f78f0d891746ec854d8ebe4661a300748d3107f611f57df0dd1b6944804ec
            c92b1cff1ed0600c93a5a5303922566bd5cd25aa731f3195b6ddecc8d709d2db
            72e2fdb2d8dffa4fd585518b198aa159e89d1cab05e7ce7e4bcd866d6dec8250
            71
        PublicExponent: 0x11
        PrivateExponent:
            0x28b82bf624e7ec268f51a7cea3fb2dbd05f8b30787249196f7ec0aa0b5e81a
            293c98e64ab488e9a1b71d99fa5141a079f867b092b21d0a8fce23a596ffc4c7
            563a94994bbe80dce8f052e3cd7eb4a17e004fbb381cf8bd50e5a1d27cfee5c3
            3b9aa3d7635cd6aec869cca5cdb8b106886a9decea705dd01a55338fca3df1b1
            cf
        Prime1:
            0xc6fa61233d8478d504937b1a1fa40dfd8171639b06e130aed71e7fd0d8ed06
            f359a534ab54d1833d6925c0ff3458f9e1a3f620680ea76b1925b89c0d650171
            97
        Prime2:
            0xfe756c4287d762c0ed35f4efbfd74d744a3102a2f94f16a3ab732614c889d7
            14e361436f7f160def3edc127c42ec89706102e74fb75f698e2e8a3b8a360f7f
            37
        Exponent1:
            0x750bc0ab515cfbc89947b1d321abcbfe8860d12e040bfe84f702e1c6254022
            34cb521efb5f11d4bab65271870fbbde487e90c7c4bd536c2ce9032e9e77a67f
            0d
        Exponent2:
            0x86b6a2b9cf72071ad7ef638dfc26b088dbfbd43829a257479700b9cec48535
            a1a58dd8683438da335d839150d822df599cc54d48520555fffa856ad0b33561
            77
        Coefficient:
            0xaccc53b69ebd067258781c5d47900e9374f7f138f15ef8a2ad5479fe4ec7d5
            ed99643a36cb3ad4cb6b21e84d383fe5a78dc6d7f2dde5a4f6b45e3770c2f373
            d
  }
```

9.2.2.3 pgpcert.pgp

Als nächstes muss das PGP-Zertifikat vom right, aus dem Public-Keyring von PGP extrahiert werden, damit FreeS/WAN sein eigenes PGP-Zertifikat der Gegenstelle schicken kann.

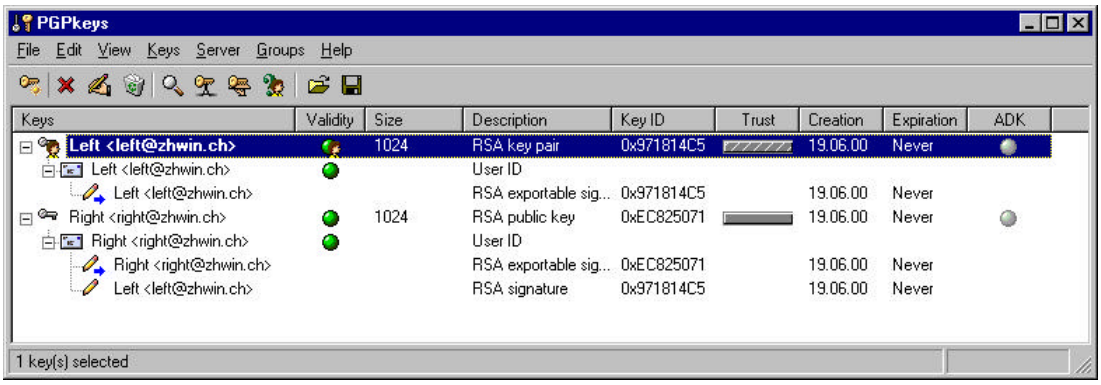
Da Pluto das ASCII-Format nicht unterstützt und PGP 6.x nur ASCII-Export zulässt, muss dies mit PGP Version 2.6.3 erfolgen (wird bei SuSE Linux 6.4 mitgeliefert).

- `pubring.pkr` aus dem Verzeichnis `C:\Programme\Network Associates\PGPNT\PGP Keyrings` (bei einer Standardinstallation von PGPnet) muss in `pubring.pgp` umbenannt werden und auf den right-Rechner kopiert werden.
- `pgp –kx right /etc/pgpcert.pgp pubring.pgp`
  Es erscheint eine Warnung, dass der Keyring mit einer neueren Version von PGP erstellt wurde, dies hat jedoch keine negativen Auswirkungen. PGP kann den Public Key (PGP-Zertifikat) trotzdem korrekt extrahieren.

9.2.3 Konfiguration PGPnet

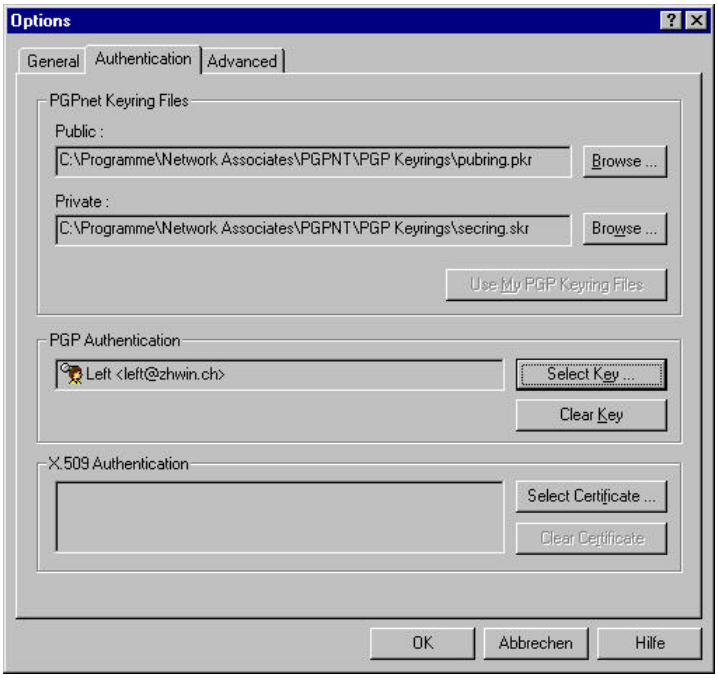
Folgende Anpassungen müssen bei PGPnet gemacht werden:

- Right-Private Key aus PGP löschen (da nur der Public Key gebraucht wird)
  Dies geht am einfachsten, indem man den Key exportiert (nur Public Key), anschliessend aus PGP löscht und dann den Public Key wieder importiert.
- Right-Public Key in PGP signieren (mit left-Secret Key), damit PGP dem right vertrauen kann.



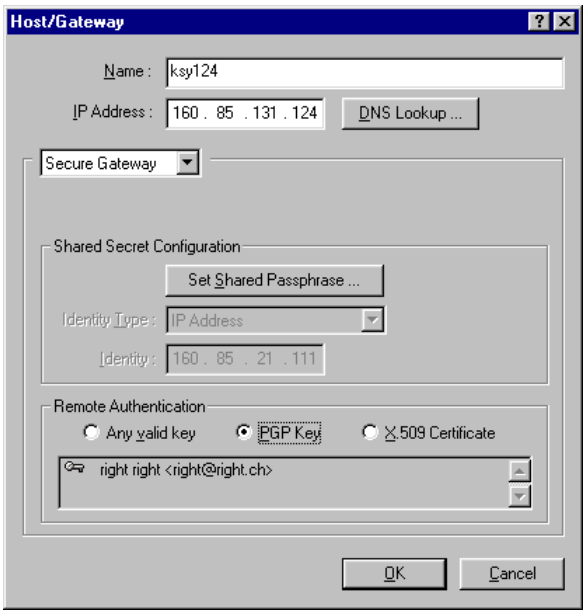
PGPkeys

- Im Hosts-Fenster von PGPnet den Befehl `View / Options / Authentication` wählen und den eigenen PGP-Secret Key auswählen (left):



PGPnet Options: Authentication

- Nun müssen noch die Connections im Fenster `Hosts` modifiziert werden. Öffnen Sie den Eintrag für den Gateway, klicken Sie auf `Clear Shared Passphrase` und wählen Sie anschliessend den PGP-Key vom right aus.

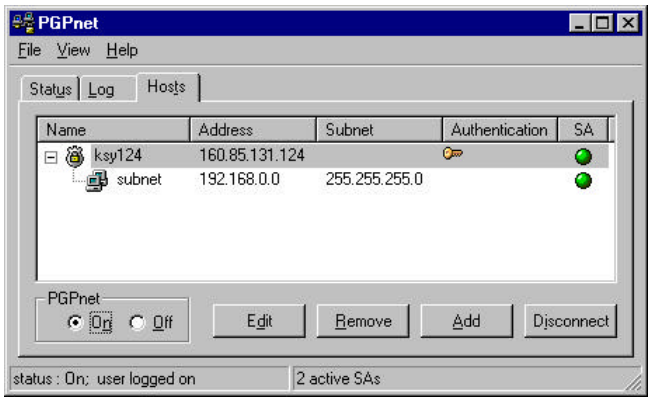


PGPnet Host/Gateway: Secure Gateway (mit PGP-Key)

- Der Eintrag für das Subnet muss nicht verändert werden.

9.2.4 Verbindung testen

Die Verbindung kann nun, wie im Kapitel 9.1.3 erklärt, gestartet und durch Anpingen der Gegenstellen getestet werden.



PGPnet Hosts (mit aufgebauten SAs)

9.3 X.509-ZERTIFIKATE

Als nächstes soll nun eine VPN-Verbindung erstellt werden, bei der die Authentifikation mit X.509-Zertifikaten erfolgt.

Vorausgesetzt wird wiederum die Installation und die Konfiguration von PGPnet und FreeS/WAN, wie dies in Kapitel 7 beschrieben wurde.

Da FreeS/WAN keine X.509-Zertifikate unterstützt, mussten wir einige Änderungen am Pluto-Quelltext vornehmen. Details dazu sind im Kapitel 8 (Theorie) und im Kapitel 7.2.3 (Installation) beschrieben.

9.3.1 X.509-Zertifikate beantragen

Als erstes müssen bei einer Zertifizierungsstelle zwei X.509-Zertifikate beantragt werden (left und right). Normalerweise macht man dies mit einem Webbrowser, in unserem Fall Netscape Communicator. Als nächstes müssen die Zertifikate aus Netscape exportiert werden:

- Wählen Sie in Netscape Communicator `Communicator / Tools / Security Info / Certificates / Yours` (in der Liste auf der rechten Seite sollten nun die beiden beantragten Zertifikate left und right erscheinen)
- Wählen Sie ein Zertifikat aus und klicken `Export`
- Geben Sie ein Passwort für die P12-Datei an (Zertifikat wird im PKCS#12-Format verschlüsselt gespeichert)

Diese Dateien können nun in PGPnet importiert werden. Dazu wählt man im Fenster PGPkeys den Befehl `Keys / Import` und wählt die beiden Dateien aus.

9.3.2 Konfiguration FreeS/WAN

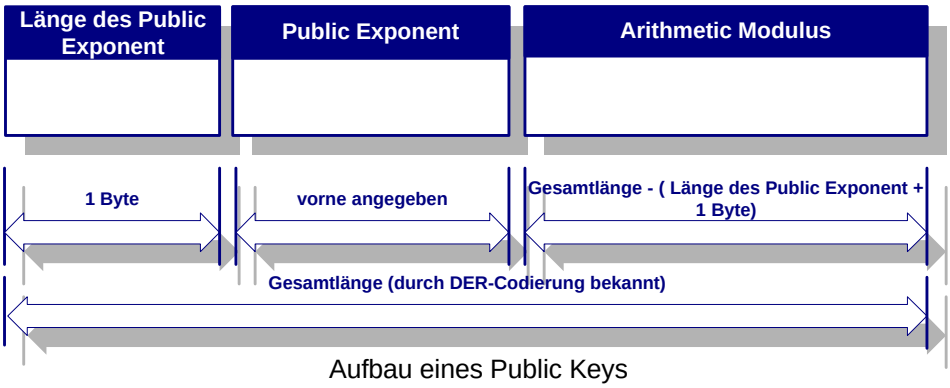
9.3.2.1 ipsec.config

Ebenfalls auf die gleiche Art wie im Kapitel 9.2.2.1 beschrieben, kann nun der Public Key von left extrahiert werden:

- `keyextractor left out.txt pubring.pkr`
  Eine Fehlermeldung erscheint, dass der Secret Key nicht geöffnet werden konnte. Dies hat jedoch keine negativen Auswirkungen, da FreeS/WAN den Privatekey des left nicht braucht.

  Leider hat sich hier ein Fehler im Keyextractor eingeschlichen:
  Der Public Key besteht aus einem Public Exponent und einem Arithmetic Modulus. Die Public Exponents sind öffentlich bekannt, wobei wir bei PGP auf den Wert 0x11 und hier auf 0x010001 gestossen sind. Wenn der Public Exponent nun nicht 0x11 ist, extrahiert der Keyextractor den Public Key falsch.

  Um dies zu verstehen, muss der genaue Aufbau eines Public Keys im X.509-Zertifikat bekannt sein:



Aufbau eines Public Keys

  Der extrahierte Public Key (aus out.txt) lautet in unserem Beispiel:

```
0x01010001a2ab3658bbc9c0ef1686d15f3906ccfd42dfed16eecedcce814c61d14790ac5
4c86de64f8c382c2bd46d0a3c9acbe15308af01d683a0dc8488e56fb1f9e08fa60f6dd143
996a8b870b6026196fd06721c14246399b5c7d7b9403da8b1ef99ce831711d505a780f944
239a0b350fd176f3a818694e90975b52158c9acbbf11f8f
```

  Wir wissen aus dem Private Key von left (siehe weiter unten), dass der Public Exponent 0x010001 lautet und somit 3 Bytes lang ist. Diesen Public Exponent finden wir tatsächlich auch im Public Key oben wieder (2. Byte bis 4. Byte), nur stimmt die angegebene Länge nicht (sollte 0x03 anstatt 0x01 sein).

  Der (verbesserte) Public Key von left kann nun in `ipsec.conf` eingefügt werden (Variable `leftrsasigkey`)

- `rightrsasigkey`: auf `0x00` setzen, damit Pluto sein Zertifikat aus `/etc/pgpcert.pgp` liest
- `authby` auf `rsasig` setzen

- Bei der `leftid` und `rightid` muss anstatt der IP-Adresse bei Preshared Secrets oder des Fingerprints bei PGP-Keys der Distinguished Name als ID verwendet werden. Dieser Distinguished Name ist im Feld "Subject" des X.509-Zertifikats gespeichert. Wir haben ein Programm entwickelt, dass diesen Distinguished Name aus einem Zertifikat herauslesen kann.

  - Als erstes muss das Zertifikat im PEM-Format, d.h. unverschlüsselt vorliegen (die P12-Dateien wurden weiter oben aus Netscape Communicator exportiert):
    `openssl pkcs12 -in right.p12 -out right.pem -nodes`
  - In der Datei right.pem muss alles ausser dem Zertifikat vom right gelöscht werden. Nachher sollte die Datei etwa folgenden Inhalt haben:

```
-----BEGIN CERTIFICATE-----
MIID/jCCAuagAwIBAgIBADANBgkqhkiG9w0BAQQFADBXMRMwEQYDVQQDEwpaSFcg
T3BlbkNBMSAwHgYDVQQLExdJbmZvcm1hdGlvbnN0ZWNobm9sb2dpZTERMA8GA1UE
ChMIemh3aW4uY2gxCzAJBgNVBAYTAkNIMB4XDTAwMDUyMjA3MjMwN1oXDTAwMDYy
MTA3MjMwN1owVzETMBEGA1UEAxMKWkhXIE9wZW5DQTEgMB4GA1UECxMXSW5mb3Jt
YXRpb25zdGVjaG5vbG9naWUxETAPBgNVBAoTCHpod2luLmNoMQswCQYDVQQGEwJD
SDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALs6goU3+YP/pOI8tmwC
EWANsx/iExSuOtiRi3F+DGRS68+k/S6eiYgA1dsX5ZIqr4s413MVNnTEG+Yc7EK/
d5A6HVpiIBT9BKUX41GSmjZ+iTfuSuwDqX5TaV8IdOR3WBsuHtj2glMhUwyGIY1R
t4dBgBQIs35/iCJUd/BLSNJK9532YF/TBKMOGdAdtXeg2Uud9BjIhF2YoBdkj6Zy
IK/BiOr/SFnCfwqUDXqsE6Kfp72pMWS7sezRpAH4MMSGRXnQhj1B/rVrp8tudUtp
FtOOEYt2w8FNnBItuQtKka0nFHIgW9ZJaIJa174CS04mRgATnVpAz1fuutxYjF1M
deUCAwEAAaOB1DCB0TAMBgNVHRMEBTADAQH/MB0GA1UdDgQWBBQ1ZhEt+wKLKQel
jQcDNuI9trQTSjB/BgNVHSMEeDB2gBQ1ZhEt+wKLKQeljQcDNuI9trQTSqFbpFkw
VzETMBEGA1UEAxMKWkhXIE9wZW5DQTEgMB4GA1UECxMXSW5mb3JtYXRpb25zdGVj
aG5vbG9naWUxETAPBgNVBAoTCHpod2luLmNoMQswCQYDVQQGEwJDSIIBADALBgNV
HQ8EBAMCAQYwCQYDVR0RBAIwADAJBgNVHRIEAjAAMA0GCSqGSIb3DQEBBAUAA4IB
AQCaoK3SQ8X19lb1d0hoPL9sa4ycpxEtyyy8y9rqSrHvCzUEL6TE+M0iBfsWo1OS
+IyUzFbzThjkzgyreatNcnWfYhRnmyPpgvFFDi7QxZ1IPjiYdz5Km5Q0fcvNrdRp
ywdyMomwa6Rpp0aXpjE+qhhOrpKT048UTYliXeOqryOombcrqQVa9ww2SaBdUc/9
Zt+Ot4iETPQybVijgKrtgNMPN58N+Qn8F3ZOf9gKYRqSiCsI837P0YZaIQ2ce5t6
qG/wlxnEXTXElhEGmcgOJIT/WFBwLSDgqR28j3rZff3qIbGMGlHu4oVsCvt3hyKt
xDTnEOq176mxmSNBhKbtZBJ2
-----END CERTIFICATE-----
```

  - Anschliessend muss in das DER-Format gewandelt werden:
    `openssl asn1parse -in right.pem -out right.der`
  - Nun kann der "Distinguished Name" extrahiert werden:
    `subjectextractor right.der right.sbj`

    Wenn alles geklappt hat, steht nun in der Datei `right.sbj` der Distinguished Name, der in die Datei `ipsec.conf` bei `rightid` eingefügt werden muss. Dabei muss noch der Präfix "@~" angegeben werden, damit Pluto weiss, dass es sich hier um einen Distinguished Name handelt.

    Der Distinguished Name für `leftid` kann analog erstellt werden.

Anschliessend sollte ipsec.conf in etwa so aussehen:

```
# /etc/ipsec.conf - FreeS/WAN IPSEC configuration file

# More elaborate and more varied sample configurations can be found
# in doc/examples.

# basic configuration
config setup
        # THIS SETTING MUST BE CORRECT or almost nothing will work;
        # %defaultroute is okay for most simple cases.
        interfaces=%defaultroute
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        klipsdebug=none
        plutodebug=all
```

```
        # Use auto= parameters in conn descriptions to control startup
        # actions.
        plutoload=%search
        plutostart=%search

# defaults for subsequent connection descriptions
conn %default
        # How persistent to be in (re)keying negotiations (0 means very).
        keyingtries=0
        # Parameters for manual-keying testing (DON'T USE OPERATIONALLY).
        spi=0x200

# connection
conn secure_gw
        authby=rsasig
        # Left security gateway, subnet behind it, next hop toward it.
        left=160.85.131.126
        leftsubnet=160.85.131.126/32
        leftid=@~305f311b301906092a864886f70d010901160c6c656674406c6566742e6
               3683112301006035504031309 6c656674206c6566743111300f060355040b
               13085a48572055736572310c300a060355040a13035a4857310b300906035
               5040613024348
        leftrsasigkey=0x03010001a2ab3658bbc9c0ef1686d15f3906ccfd42dfed16eece
                      dcce814c61d14790ac54c86de64f8c382c2bd46d0a3c9acbe15308
                      af01d683a0dc8488e56fb1f9e08fa60f6dd143996a8b870b602619
                      6fd06721c14246399b5c7d7b9403da8b1ef99ce831711d505a780f
                      944239a0b350fd176f3a818694e90975b52158c9acbbf11f8f
        # Right security gateway, subnet behind it, next hop toward it.
        right=160.85.131.124
        rightsubnet=160.85.131.124/32
        rightid=@~3063311d301b06092a864886f70d010901160e72696768744072696768
                742e6368311430120603550403130b726967687420726967687431113 00f
                060355040b13085a48572055736572310c300a060355040a13035a485731
                0b3009060355040613024348
        rightrsasigkey=0x00
        # Authorize this connection, but don't actually start it, at startup.
        auto=add

conn secure_subnet
        authby=rsasig
        # Left security gateway, subnet behind it, next hop toward it.
        left=160.85.131.126
        leftsubnet=160.85.131.126/32
        leftid=@~305f311b301906092a864886f70d010901160c6c656674406c6566742e6
               3683112301006035504031309 6c656674206c6566743111300f060355040b
               13085a48572055736572310c300a060355040a13035a4857310b300906035
               5040613024348
        leftrsasigkey=0x03010001a2ab3658bbc9c0ef1686d15f3906ccfd42dfed16eece
                      dcce814c61d14790ac54c86de64f8c382c2bd46d0a3c9acbe15308
                      af01d683a0dc8488e56fb1f9e08fa60f6dd143996a8b870b602619
                      6fd06721c14246399b5c7d7b9403da8b1ef99ce831711d505a780f
                      944239a0b350fd176f3a818694e90975b52158c9acbbf11f8f

        # Right security gateway, subnet behind it, next hop toward it.
        right=160.85.131.124
        rightsubnet=192.168.0.0/24
        rightid=@~3063311d301b06092a864886f70d010901160e72696768744072696768
                742e6368311430120603550403130b726967687420726967687431113 00f
                060355040b13085a48572055736572310c300a060355040a13035a485731
                0b3009060355040613024348
        rightrsasigkey=0x00
        # Authorize this connection, but don't actually start it, at startup.
        auto=add
```

9.3.2.2 ipsec.secrets

Als nächstes muss die FreeS/WAN-Datei `ipsec.secrets` konfiguriert werden. Wie bei den PGP-Keys im Kapitel 9.2.2.2 schon beschrieben, müssen dazu die Secret-Informationen mit Hilfe des Keyextractors extrahiert werden:

- `keyextractor right out.txt pubring.pkr secring.skr`
  Anschliessend steht das Resultat in der Datei `out.txt`.
- Extrahierte Secret Key-Informationen in die Datei `ipsec.secrets` einfügen.

Beispiel:

```
# This file holds shared secrets which are currently the only inter-Pluto
# authentication mechanism.  See ipsec_pluto(8) manpage.  Each secret is
# (oversimplifying slightly) for one pair of negotiating hosts.

# The shared secrets are arbitrary character strings and should be both
# long and hard to guess.

# Note that all secrets must now be enclosed in quotes, even if they have
# no white space inside them.


: RSA {
        Modulus:
            0xa3eb2ec459ce862c4225c7a58dfabcf1a4d620f8fd4f5154ae03730d93023e
            48ad3017c6cba3deb7ddc52e65125d0347f061daef05bc731034da77a1bf4861
            d5c09e215e89431377f5f78394a730e28ffb13274c4a9cb8fbe67ecd2be3ab52
            b66c89e887859fc970b3f8bf86a30b53f9ce068d3fea08cb9029736c08c30756
            69
        PublicExponent: 0x010001
        PrivateExponent:
            0x249312514f748c6d8da8de5e2b4ada23d4d2ce72a01ab59bfe63d8a51bbb31
            6887f42629fe40272ec17600bf1e94d06999d24bc66f7237fe52bcb85ac7d27f
            b995a013176e45d8401824a9ed58de1f71de749f7f5b663d1c7267f15a9373f6
            453f8ff457c94946113641ff405f2de3d724ac60fae7e80208df61f72d12366a
            01
        Prime1:
            0xca9371911c3398ba0738a9c82e6cd98a0a3f81e73eaab0be6abab57787a34d
            54ad3d5832df99c7ff44c5b4af15c0ced38dbd0f31621cf5347e032198e4b8ef
            c1
        Prime2:
            0xcf25dcae41fd3ef0b1ee7c3537bcde9acd3a2184701bb06a2b3ef9e1b00952
            447627156bbf0e93e537c08a325102d478b74e471da8a00d5048757209533510
            a9
        Exponent1:
            0x14eb851ee73aacdb0943e0c4bb86a2308f3e5fbbf1ccced3db51cccbbb4632
            063c9dc92df7c979cdf2dc37fc9cd27c897c69b9d7efb9edd8adeac4dd754164
            01
        Exponent2:
            0x2d8176ca3d2ae1f1c454ccaa7035287dbf49dee5162faf0093c180e9685f93
            d80050b3aef8c8be0aa3c5c3562c5fbfc74915841fead4463875c27f2bb740b8
            09
        Coefficient:
            0x80fc66b2c1365b47205f801d2e026da634855bde2b969229e5746cf19d2b73
            384d49a1c87fc7a42a1ec352c8acbbe6d78f8761a49a2f668cebe2f222e7dc44
            dd

    }
```

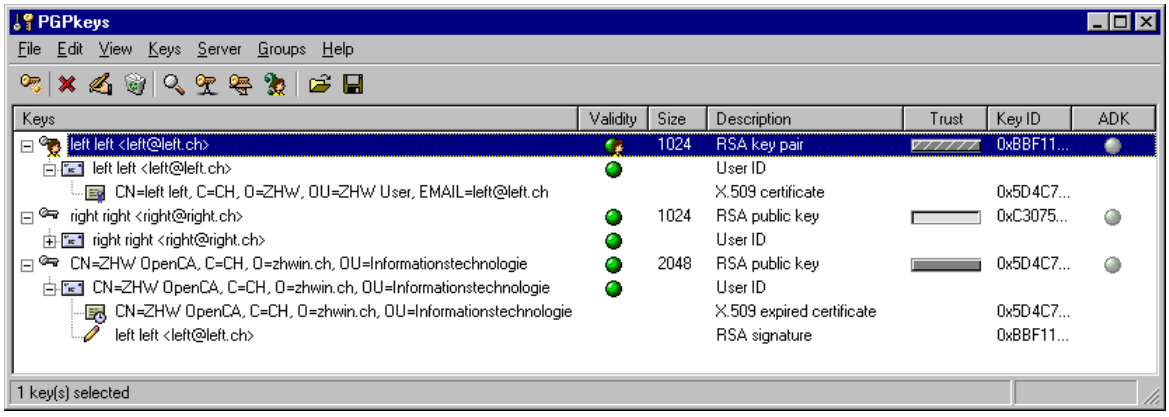
9.3.2.3 pgpcert.pgp

Bei der Authentifizierung mit PGP-Keys haben wir an dieser Stelle das PGP-Zertifikat von left extrahiert, damit FreeS/WAN sein eigenes Zertifikat mitschicken kann. Unsere Tests haben aber gezeigt, dass PGPnet dieses Zertifikat ignoriert, wenn es bereits im Keyring von PGP vorhanden ist - was bei uns ja der Fall ist. Aus diesem Grund muss das Zertifikat an dieser Stelle nicht neu extrahiert werden.

9.3.3 Konfiguration PGPnet

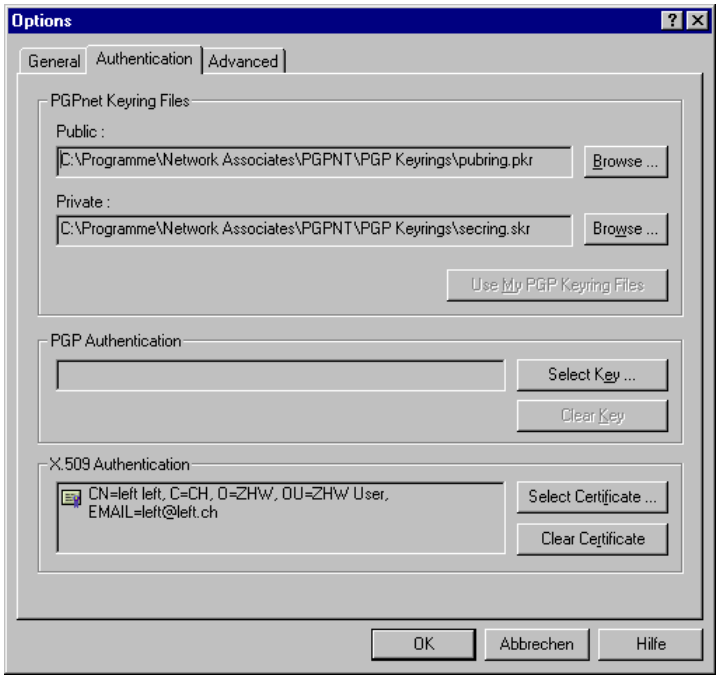
Nun muss PGPnet noch angepasst werden.

- Der Right-Private Key kann aus PGPnet auf dem left-Computer gelöscht werden (da nur der Public Key gebraucht wird)
  Das geht am einfachsten, indem man den Key exportiert (nur Public Key), anschliessend aus PGP löscht und dann den Public Key wieder importiert.
- Damit PGP der Zertifizierungsstelle vertrauen kann, muss das CA-Zertifikat in PGP signiert werden (mit Secret Key von left):



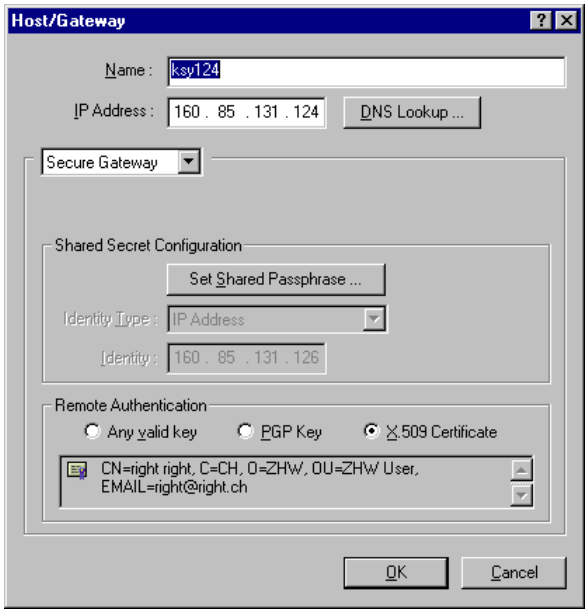
PGPkeys

- Im Hosts-Fenster von PGPnet unter `View / Options / Authentication` das eigene X.509-Zertifikat (left) auswählen:



PGPnet Options: Authentication

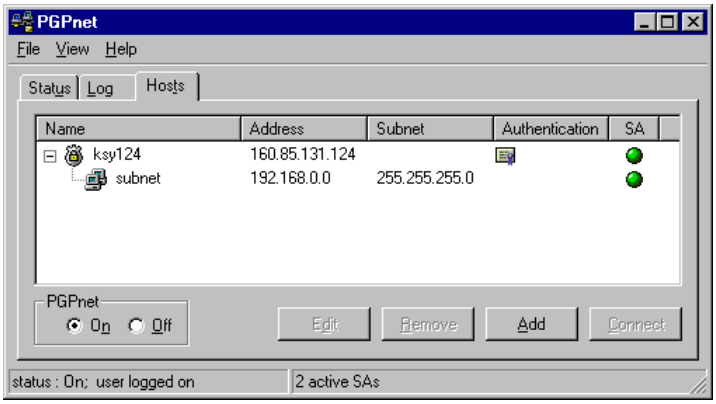
- Ebenfalls im Hosts-Fenster muss bei der Gateway-Connection das right-Zertifikat angegeben werden.



PGPnet Host/Gateway: Secure Gateway (mit X.509-Zertifikat)

9.3.4 Verbindung testen

Die Verbindung kann nun gestartet und durch Anpingen der Gegenstellen getestet werden.



PGPnet Hosts (mit aufgebauten SAs)

10 SCHLUSSWORT

10.1 BEURTEILUNG DER ARBEIT

Für diese Arbeit war ein sehr grosses Grundlagenwissen im Bereich der Secure Network Communication nötig. Nur schon die Analyse des Pluto-Barfs war für uns am Anfang recht schwierig. Wir hatten auch häufig Mühe, die richtigen Dokumentationen zu finden.

Als wir dann zur Analyse des Quelltexts übergingen, waren wir erst einmal von den mehreren tausend Zeilen Pluto-Quelltext überwältigt.

Trotzdem hat uns die Arbeit Spass gemacht - hatten wir doch einige grosse Erfolgserlebnisse. Für uns war diese Projektarbeit eine wertvolle Übung im Analysieren und Erweitern von OpenSource Quelltext. Zusätzlich haben wir im Bereich Linux und VPN viel dazugelernt

10.2 WEITERENTWICKLUNGS-MÖGLICHKEITEN

Wir haben unser Ziel erreicht und eine funktionierende VPN-Lösung mit X.509-Zertifikaten erstellen können. Trotzdem gibt es noch einige Punkte, die für uns unbefriedigend sind und noch verbessert werden sollten.

Bei der aktuellen FreeS/WAN-Implementation müssen die Key-Informationen recht mühsam mit dem Keyextractor extrahiert und in die `ipsec.secrets` bzw. `ipsec.conf`-Datei geschrieben werden.

Als einfachste Lösung könnte das Extrahieren und Einfügen der Key-Informationen in die FreeS/WAN-Konfiguration mit einem Shellscript realisiert und automatisiert werden. Mit dieser Lösung müssten keine Änderungen am FreeS/WAN-Quelltext gemacht werden.

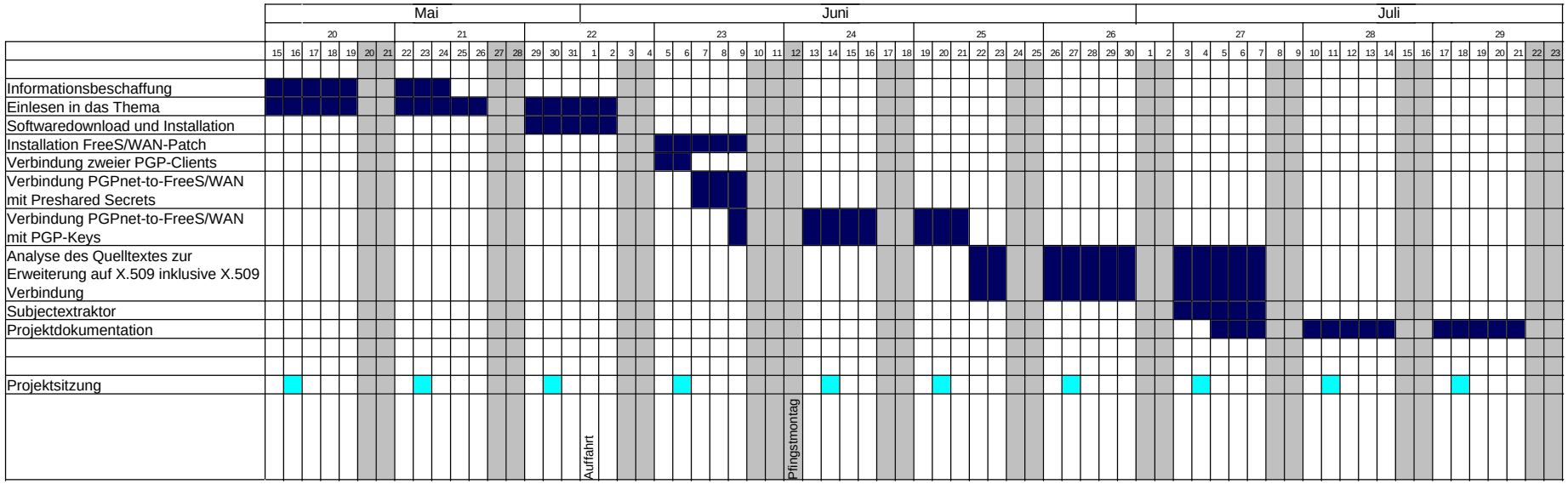
Optimaler wäre eine Lösung, bei der ein FreeS/WAN-Utility X.509-Zertifikate direkt im PKCS#12-Format importieren kann (das ist das Format, in welchem Netscape die Zertifikate exportiert). Die Schlüssel müssten nicht mehr manuell in die Konfigurationsdateien geschrieben werden.

Am besten wäre aber eine Schlüsselverwaltung nach dem Vorbild von PGPnet, bei welcher der eigene Secret Key und fremde X.509-Zertifikate importiert werden können. Idealerweise könnte diese Schlüsselverwaltung Zertifikate im PKCS#12-Format importieren. Zusätzlich müsste eine Liste von Root-Zertifikaten bestehen, denen man traut (wie dies z.B. in Netscape gelöst ist).

Weiter müsste eine Möglichkeit bestehen, Zertifikate z.B. anhand des Distinguished Name bei LDAP-Servern abzufragen. Dies ist nötig, wenn ein Client eine VPN-Verbindung aufbauen möchte, aber nicht die vollständige

X.509-Zertifizierungskette mitschickt. Um die Echtheit des X.509-Zertifikats zu überprüfen, muss FreeS/WAN sämtliche Zertifikate abfragen können, was nur möglich ist, wenn auf die LDAP-Server zugegriffen werden kann. In der `ipsec.config`-Datei wäre dadurch die Angabe des Public Keys des Kommunikationspartners nicht mehr nötig. Jeder, der dem FreeS/WAN-Security-Gateway ein gültiges X.509-Zertifikat schickt, kann eine Verbindung aufbauen. Ist dies nicht erwünscht, müsste noch eine Liste mit Distinguished Names erstellt werden, zu denen eine Verbindung aufgebaut werden darf.

11 ZEITPLAN


Mai
Juni
Juli
20
21
22
23
24
25
26
27
28
29
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Informationsbeschaffung
Einlesen in das Thema
Softwaredownload und Installation
Installation FreeS/WAN-Patch
Verbindung zweier PGP-Clients
Verbindung PGPnet-to-FreeS/WAN mit Preshared Secrets
Verbindung PGPnet-to-FreeS/WAN mit PGP-Keys
Analyse des Quelltextes zur Erweiterung auf X.509 inklusive X.509 Verbindung
Subjectextraktor
Projektdokumentation
Projektsitzung
Auffahrt
Pfingstmontag

12 LITERATUR & LINKS

12.1 AUFGABENSTELLUNG

[1] http://fbi.zhwin.ch/ksy/pa/PA_2000_Sna03.htm

12.2 SOFTWARE

[2] PGP und PGPnet Software
http://www.man.torun.pl/pub/pgp/

[3] PGPnet 6.5.3 - US Freeware Version (transport mode only, no X.509 certificates)
http://www.pgpi.org/

[4] PGPnet 6.5.1i - International Professional Version (tunnel mode and X.509 certificates)
http://www.pgpinternational.com/

[5] FreeS/WAN inkl. Onlinedokumentation
http://www.freeswan.org/

[6] PGP Patch
http://fbi.zhwin.ch/ksy/pa/freeswan-pgpnet.tar.gz

[7] Peter Gutman's Crypto Library
ftp://ftp.easynet.de/pub/pc/security/crypl200.zip

12.3 DOKUMENTATION

[8] ZHW Diplomarbeit zum Thema IPsec
http://www.twi.ch/~sna/research/VPN/DA99/DA99_gaertner_uenal.pdf

[9] Virtual Private Network Consortium
http://www.vpnc.org/

[10] Peter Gutman's Crypto Library
http://www.cs.auckland.ac.nz/~pgut001/cryptlib/

[11] Ipsec practical configurations for Linux Freeswan 1.3.
http://jixen.tripod.com/

[12] IP Security Protocol
http://www.ietf.org/html.charters/ipsec-charter.html

[13] Diverse Links zum Thema Sicherheit
http://amor.rz.hu-berlin.de/~h0271cbj/#resources

[14] A Survey of Public Key Infrastructures
http://home.xcert.com/~marcnarc//PKI/thesis/

[15] Das OpenSSL Handbuch
http://www.pca.dfn.de/dfnpca/certify/ssl/handbuch/ossl095/

[16] OpenSSL Project
http://www.openssl.org/docs/apps/openssl.html

[17] Deutsche Anleitung zu PGP
http://home.kamp.net/home/kai.raven/pgp/pgpindex.html

[18] IPSec Integration in einer Linux-Umgebung
http://www.informatik.uni-muenchen.de/~_fackelma/fopra/ip18/ip18.html

[19] Informationen zum Thema IPsec
http://www.twi.ch/~sna/research/VPN/ipsec.htm

[20] ITU-T RECOMMENDATION X.509
http://fbi.zhwin.ch/ksy/Block04/ITU/x509_ww7.zip

[21] Secure Network Communication Part IV - Secure Network Applications
http://www.strongsec.com/zhw/KSy_SecApp_1.pdf

[22] Swan and Penguin
http://www.heise.de/ct/english/99/16/180/

[23] Nachschlagewerk für RFCs
http://www.faqs.org/rfcs/