VPNadmin Tool

Zürcher Hochschule Winterthur
Studiengang Informationstechnologie

Diplomarbeit 2001

Zentrale Security Policy für Linux IPSec

Marco Bertossa & Thomas Wendel

Partnerfirma:
open systems AG
Räffelstrasse 29
8045 Zürich

Inhaltsverzeichnis

1 Zusammenfassung

Bei der Erstellung eines firmeninternen Netzwerkes lohnt sich die Überlegung, die einzelnen Filialen nicht über teure Mietleitungen, sondern über das kostengünstigere Internet miteinander zu verbinden. Wie wir Medienberichten entnehmen können, ist aufgrund von Angriffen über das Internet die Sicherheit solcher Verbindungen jedoch gefährdet. Die Verwendung von Virtual Private Networks (VPN) garantiert einen sehr hohen Schutz, ohne dass für den Benutzer des firmeninternen Netzwerkes ein Mehraufwand durch die benötigten zusätzlichen Sicherheitsmechanismen entsteht.

Die Firma Open Systems AG in Zürich bietet ihren Kunden flexible, preiswerte VPN Lösungen an, welche sie rund um die Uhr überwachen. Ein Problem, das sich bei komplexeren Netzwerken mit vielen Security Gateways stellt, ist die konsistente Verwaltung aller Zertifikate und Konfigurationsdateien der beteiligten Gateways.

Im Auftrag der Firma Open Systems AG haben wir ein Administrations Tool entwickelt, mit welchem alle für ein VPN benötigten Angaben und Parameter in einer Datenbank zentral gespeichert und verwaltet werden können. Des weiteren können anhand der Daten die Konfigurationsfiles aller Security Gateways generiert und über eine kryptografisch sichere Verbindung im ganzen Netz verteilt werden. Zudem ist es möglich, eine eigene Zertifizierungsstelle zu unterhalten, welche X.509 Zertifikate generiert und verwaltet. Änderungen der Netzwerktopologie können so einfach und effizient vorgenommen werden, ohne dass eine längere Downtime einzelner Verbindungen in Kauf genommen werden muss.

Für ihre VPN's verwendet Open Systems AG IPSec Produkte von verschiedenen Anbietern. Aus zeitlichen Gründen beschränkten wir uns darauf, ein Filter zum Generieren der Konfigurationsfiles zu implementieren, welches für das IPSec von Linux FreeS/WAN ausgelegt ist. Um weitere und vor allem auch zukünftige Produkte unterstützen zu können, legten wir bei der Spezifikation unseres Tools grossen Wert darauf, alle Komponenten möglichst modular aufzubauen. Dies erlaubt, auch eine Datenbank eines anderen Herstellers zu verwenden. Die Daten, welche man zum Generieren der Konfigurationsfiles benötigt, werden in xml-Files gespeichert, die eine universelle Schnittstelle zu den Generierungsfilter bilden. Im Ausblick auf andere IPSec Lösungen haben wir beim Design des relationalen Datenbankschemas verschiedene zusätzliche Entitäten und Attribute eingeführt.

In Rahmen unserer Diplomarbeit haben wir ein Werkzeug entwickelt, mit welchem auch komplexere Virtual Private Networks entworfen und ohne grossen Aufwand sauber konfiguriert werden können. Während dieser Arbeit haben wir wichtige Erfahrungen in der Projektplanung und deren Durchführung gesammelt.

2 Abstract

By creating an internal network system it is worthwhile to consider whether to connect its individual branches by expensive rented lines or the more reasonable Internet. However, this raises the question about the security of such connections, since the media informs us about regular abuses by means of the Internet. The usage of Virtual Private Networks (VPN) guarantees outstanding security without causing additional expenditure for the user of the company's internal network.

Open Systems AG in Zurich offers their clients flexible and reasonable VPN solutions, which they keep permanently under surveillance. One problem which arises when using such complex networks consisting of many security gateways, is to keep the administration of all certificates and configuration files of the involved gateways consistent.

On behalf of the company Open Systems AG we developed an administration tool, with which the specifications and parameters needed in order to describe a VPN can be stored and administered centrally in a data base. Moreover, the configuration files of the security gateways can be generated on the basis of the stored data and distributed over an encrypted connection within the whole network. A Certificate Authority (CA) can be maintained with which X.509 certificates can be generated and administered. Modifications of the network topology can be made simply and efficiently without any longer downtime of individual connections.

Open Systems AG uses for their VPN's IPSec products of different providers. Within this work we implemened a filter, which is able to generate the configuration files used by Linux IPSec solution FreeS/WAN. In order to be able to support further and above all future products, we attached great importance to the specification of our tool and to the structure of modular components. Consequently, it is possible to use data bases of other manufacturers. Data which are needed to generate the configuration files are stored as xml-files which in their turn form a universal interface with the generation filter. When designing the relational data base some entities and attributes were introduced considering the use of other IPSec solutions.

Within the scope of this diploma thesis we created a tool which allows to sketch and configure neatly and without large expenditure complex Virtual Private Networks. During this process we have gained a lot of knowledge of project planning and its execution.

3 Aufgabenstellung

Kommunikationssysteme (KSy)

Praktische Diplomarbeiten 2001 - Sna01/4

Zentrale Security Policy für Linux IPSec

Studierende:

- Marco Bertossa, IT3a
- Thomas Wendel, IT3a

Partnerfirma:
open systems AG, Räffelstrasse 29, 8045 Zürich
Termine:

- Ausgabe: Donnerstag, 6.09.2001 09:00 bei open systems
- Abgabe: Montag, 29.10.2001 12:00

Beschreibung:
Auf der Basis von Linux und dem FreeS/WAN IPSec Stack lassen sich zuverlässige, durchsatzstarke und kryptografisch sicherere Security Gateways zu einem günstigen Preis realisieren. Mit zunehmender Anzahl der eingesetzten Gateways und damit auch der IPSec Verbindungen zwischen den Gateways, steigt jedoch der mit der Aktualisierung der lokalen Konfigurationsdaten verbundene Administrierungssaufwand und die Konsistenz der Einstellungen kann nicht mehr gewährleistet werden.

Wünschenswert wäre deshalb eine netzwerkweite Formulierung einer Security Policy und deren Umsetzung mittels einer zentralen Verwaltung der Konfigurationsdaten, verbunden mit einer kryptografisch sicheren Verteilung dieser Daten auf die einzelnen Security Gateways auf der Basis von "scp" oder "sftp".

Konfigurationsdaten, wie z.Bsp. die zu verwendenden Verschlüsselungs-, Authentisierungs- und Komprimierungsalgorithmen, sowie eine Liste der zugelassenen IPSec Verbindungen zwischen den Security Gateways, inklusive der benötigten Schlüssel und Zertifikate sollen in einer relationalen Datenbank abgelegt und via ein Web-Interface administriert werden können. Die Anbindung der Datenbank soll mit PHP realisiert werden.

Ziele:

- Systemkonzept
- Software und Datenbank-Spezifikation
- Implementation und Test
- Installations- und Betriebsanleitung
- Dokumentation der Arbeit

Infrastruktur / Tools:

- Raum: E523 / open systems AG, Räffelstrasse 29, 8045 Zürich
- Rechner: 2 PCs mit Dual-Boot: SuSE Linux / Windows NT 4.0
- SW-Tools: Oracle DB, PHP

Literatur / Links:

- Linux - FreeS/WAN Projekt
  http://www.freeswan.org
- X.509 Patch für Linux - FreeS/WAN
  http://www.strongsec.com/freeswan
- OpenSSH Projekt
  http://www.openssh.org
- PHP Homepage
  http://www.php.net

Winterthur, 6. September 2001

Dr. Andreas Steffen

4 Grundlagen

4.1 VPN Theorie

Für eine sichere Verbindung zwischen lokalen Netzen (Intranets) kann ein virtueller privater Tunnel durch das Internet gelegt werden. Dieser virtuelle private Tunnel (VPN) gewährleistet eine abhörsichere und verschlüsselte Verbindung in den unsicheren Netzen.
Zwischen den lokalen Netzwerken und dem unsicheren Netz werden sogenannte Security Gateways eingebunden, welche die Pakete vor dem Eintritt in das unsichere Netz verschlüsseln und beim Verlassen wieder entschlüsseln und somit einen sicheren Tunnel zwischen den eigenen Netzwerken aufbauen.

4.2 IPSec-FreeS/WAN

Linux FreeS/WAN [8] ist eine Implementation von IPSec und IKE (Internet Key Exchange) [17] für Linux.
IPSec (Internet Protocol Security) ist ein Protokoll, welches Services zur Authentifizierung sowie Verschlüsselung der Daten auf der Vermittlungsschicht (IP) mittels starker Kryptographie zur Verfügung stellt. Es können sichere Tunnels durch das unsichere Internet gelegt werden, welche die Daten transportieren und vor unbefugtem Zugriff oder Veräderungen schützen.
Um Verbindungen zwischen den Security Gateways zu erstellen, müssen diese authentifiziert werden, um sicherzustellen, dass es sich auch um den gewünschten Kommunikationspartner handelt.

4.2.1 Konfiguration

Einige Linux Distributionen (zum Beispiel SuseLinux) liefern eine Version von FreeS/WAN bereits mit. Falls dies nicht der Fall ist, oder eine neuere Version eingesetzt werden soll, kann die neuste Version von der Homepage [8] heruntergeladen werden. Um die volle Unterstützung für die X.509 Zertifikate zu erhalten, sollte noch der dazugehörige, neueste Patch [9] installiert werden. Auf die Installation des Softwarepaketes wird hier nicht weiter eingegangen, sondern auf die Dokumentation Dritter verwiesen (siehe Quellenverzeichnis).
Für die einzelnen Verbindungen wichtige Parameter sind die verwenden Authentisierungsalgorithmen, Identifikation der Verbindungspartner, sowie der verwendete IPSec-Mode. In den beiden Konfigurationsfiles, `ipsec.conf` und `ipsec.secrets`, stehen die entsprechenden Anagaben zu den Verbindungen (`conf`), beziehungsweise die zur Authentisierung verwendeten Schlüssel (`secrets`). Je nach verwendeter Verbindungsart müssen diese dementsprechend angepasst werden.

4.2.2 Authentisierung

Bei der Athentifizierung geht es darum, sich gegenüber seinem Verbindungspartner zu identifizieren. Dies kann auf zwei verschiedene Arten durchgeführt werden:

- Preshared-Secrets
- Zertifikate

Preshared-Secrets

Für eine Preshared-Secrets Authentisierung besitzen beide Verbindungspartner ein zuvor vereinbarter Geheimschlüssel, bevor die Verbindung zwischen diesen beiden erstellt wird. Die Identität des Gegenübers ist dadurch gegeben, dass für jede Verbindung genau ein gemeinsames Geheimnis vorhanden ist. Dieses wird im `ipsec.secrets` gespeichert, zusammen mit der Angabe zur Verbindung:

```
Host-IP-Adresse Remote-IP-Adresse: PSK "Preshared Secret"
```

Im `ipsec.conf` wird für jede Verbindung angegeben, welche Authentifizierung angewendet wird. Bei einer Preshared-Verbindung genügt der Eintrag der IP-Adresse zur Identifizierung des Gegenübers.

```
authby=secret
left=host-ip-address
right= remote-ip-address
```

Zertifikate

Bei der Authentisierung mit Zertifikaten [4.3] wird ein öffentlicher RSA Schlüssel einer Identität zugewiesen. Beim Verbindungsaufbau (IKE) wird ein X.509 Zertifikat übermittelt, welches verifiziert [PA 01] wird wimit die Authentizität des Gegenübers gewährleistet ist. Der private Schlüssel zu dem gesendeten Zertifikat wird ebenfalls im `ipsec.secrets` gespeichert.

Wenn verschiedene Zertifikate für verschiedene Verbindungen verwendet werden, wird der Subject-Alt-Name des dazugehörigen Zertifikates angegeben, um dieses dem dazugehörigen Private-Keys zuzuordnen.

```
Subject-Alt-Name:RSA        {
      ….
}
```

Im ipsec.conf müssen neben der IP-Adresse auch noch der Subject-Alt Name für die Identifizierung angegeben werden. Wenn für alle Verbindungen das selbe Zertifikat verwendet wird, wird dieses im DER-Format (binär) im `/etc/x509cert.der` abgespeichert. Werden verschiedene Zertifikate verwendet, muss der Name des zu verwendenden Zertifikates für jede einzelne Verbindung angegeben werden [Patch]. Diese müssen im PEM-Format (Base 64) im Verzeichnis `/etc/ipsec.d` gespeichert werden.

```
authby=rsasigkey
leftid=host-subject-alt-name
leftcert=certificate.pem
right=remote-subject-alt-name
```

4.2.3 Verbindungen

IPSec unterstützt zwei Modi [17], welche sich darin unterscheiden, wie die IP Pakete transportiert, beziehungsweise verarbeitet werden.

- Transport-Mode:
  Hier handelt es sich um eine Verbindung innerhalb eines lokalen Netzwerkes. Dementsprechend ist der Sicherheitsgrad geringer als beim Tunnel-Mode. Es wird nur auf die Payload entweder das AH oder das ESP Protokoll angewendet. Der IP-Header bleibt unverändert.

- Tunnel-Mode:
  Hier handelt es sich um Verbindungen zwischen zwei Netzwerken über ein öffentliches Netz. Hier wird das ursprüngliche Tunneling zusammen mit ESP angewendet. Dies wird erreicht indem das originale Datenpaket in ein neues IP-Datenpaket "eingepackt" und somit die Identität verbogen wird. Die Payload des neuen Paketes (das gesamte ursprüngliche Paket) wird noch mit ESP verschlüsselt um die Sicherheit zusätzlich zu erhöhen.

Ein weiterer Pluspunkt des Tunnel-Modus ist, dass in den einzelnen Netzwerken private IP-Adressen benutzt werden können. Durch das Verpacken dieser IP-Pakete in solche mit registrierten und routingfähigen IP-Adressen, können diese durch das Internet übertragen werden.

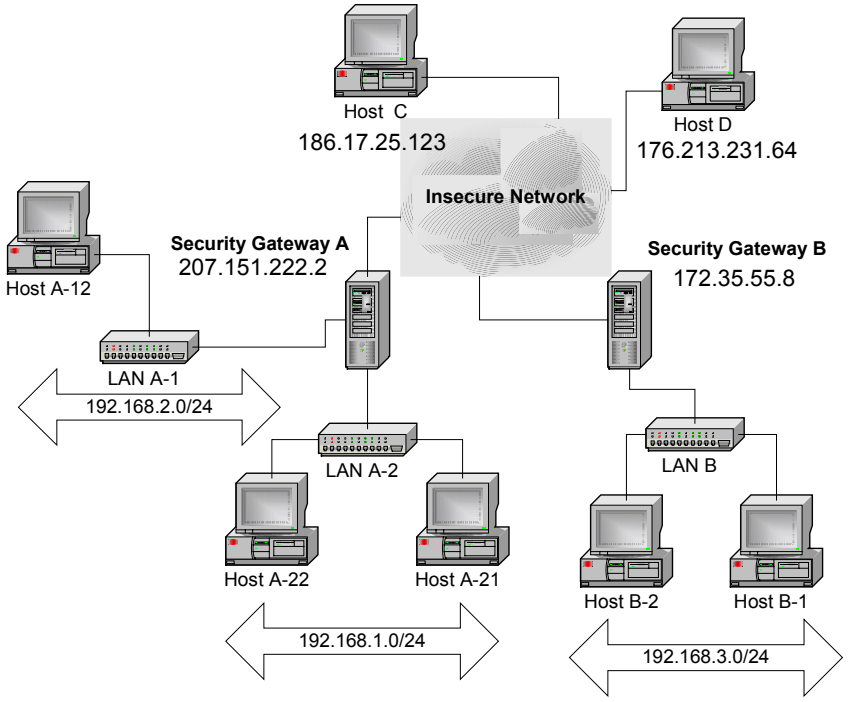


Abbildung 4.1: Netzwerk Beispiel IPSec Verbindungen

Bei den folgenden Beispielen werden die für diese Arbeit wichtigsten Verbindungstypen beschrieben. Es werden jeweils nur die wichtigsten Einträge in der Datei ipsec.conf angegeben. Vollständige Konfigurationsfiles, ipsec.conf und ipsec.secrets, sind im Anhang aufgelistet.

Einfache Host-Host Verbindung

Für eine einfache Host zu Host Verbindung wird wie oben beschrieben der Transport Modus verwendet. Im Netzwerk in Abbildung 4.2 würde eine Verbindung von Host C zu Host D mit preshared-secrets folgendermassen aussehen:

```
ipsec.conf
# a preshared secret connection
conn CD
     type=transport
     left=186.17.25.123
     right=176.213.231.64
ipsec.secrets
186.17.25.123 176.213.231.64: PSK "preshared secret"
```

Subnet - Subnet Verbindung 1 Zertifikat
Eine Verbindung von LAN A-1 zu LAN B, Authentisierung mit Zertifikaten, würde mit dem Tunnel-Mode realisiert werden:

ipsec.conf

```
# first subnet LAN A-1
conn lanA1-lanB-1
     type=tunnel
     leftrsasigkey=%cert
     left=207.151.222.2
     leftid=left@mail.com
     leftsubnet=192.168.1.0/24
     rigtrsasigkey=%cert
     right=173.35.55.8
     righted=right@mail.com
     rightsubnet=192.168.3.0/24

# second subnet LAN A-2
conn lanA1-lanB-1
     type=tunnel
     leftrsasigkey=%cert
     left=207.151.222.2
     leftid=left@mail.com
     leftsubnet=192.168.2.0/24
     rigtrsasigkey=%cert
     right=173.35.55.8
     righted=right@mail.com
     rightsubnet=192.168.3.0/24
```

ipsec.secrets

```
:RSA    {
          Modulus:         0xCCF100…
          PublicExponent:  0x010001
          PrivateExponent  0xB86EC…
          Prime1:          0xDD75B…
          Prime2:          0xEF98B…
          Exponent1        0xAB12D…
          Exponent2        0xCD56A…
          Coefficient:     0x1AD34…
}
```

Hier wurde für beide Verbindungen das Host-Zertifikat `/etc/x509cert`.der verwendet, somit musste kein `leftcert` Eintrag gemacht werden. Da sich hinter dem Security-Gateway-A zwei LAN (Protected Network) befinden, muss für jedes eine eigene Verbindung Eingetragen werden.

Subnet - Subnet Verbindung verschiedene Zertifikate
Die selbe Verbindung (LAN A-1 zu LAN B) kann mit verschiedenen Zertifikaten für verschiedene Verbindungen realisiert werden:

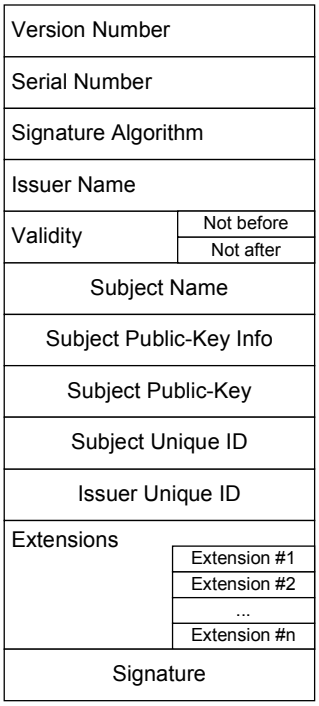
```
ipsec.conf
# first subnet LAN A-1
conn lanA1-lanB-1
     type=tunnel
     leftrsasigkey=%cert
     left=207.151.222.2
     leftcert=A-1.pem
     leftsubnet=192.168.1.0/24
     rigtrsasigkey=%cert
     right=173.35.55.8
     rightcert=B.pem
     rightsubnet=192.168.3.0/24

# second subnet LAN A-2
conn lanA1-lanB-1
     type=tunnel
     leftrsasigkey=%cert
     left=207.151.222.2
     leftcert=A-2.pem
     leftsubnet=192.168.2.0/24
     rigtrsasigkey=%cert
     right=173.35.55.8
     rightcert=B.pem
     rightsubnet=192.168.3.0/24


ipsec.secrets
# Private Key from certificate A-1.pem
a1@email.com:RSA    {
          Modulus:         0xCCF100…
          PublicExponent: 0x010001
          PrivateExponent 0xB86EC…
          Prime1:          0xDD75B…
          Prime2:          0xEF98B…
          Exponent1        0xAB12D…
          Exponent2        0xCD56A…
          Coefficient:     0x1AD34…
}

# Private Key from certificate A-2.pem
a2@email.com:RSA    {
          Modulus:         0xCDE34…
          PublicExponent: 0x010001
          PrivateExponent 0xDE89A…
          Prime1:          0cBC237…
          Prime2:          0xAFC3B…
          Exponent1        0xBDC34…
          Exponent2        0x98BD1…
          Coefficient:     0xEF65A…
}
```

Wenn lokal gespeicherte Zertifikate verwendet werden (`/etc/ipsec.d`), muss der Subject-Alt-Name als ID nicht mehr angegeben werden. Diese Variante wurde in dieser Arbeit gewählt, da es vorkommen kann, dass ein Security Gateway verschiedene Interfaces mit verschiedenen Zertifikaten benutzt. Wenn dies der Fall ist, müssen im `ipsec.conf` die Interfaces explizit angegeben werden:

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

4.3 Zertifikate

Zertifikate stellen die Verbindung her zwischen einer Identität (einer Person oder einer Instanz) und einem öffentlichen Schlüssel. Indirekt wird damit aufgrund des Zusammenhangs zwischen öffentlichem und geheimen Schlüssel auch ein Bezug zwischen Identität und Secret-Key hergestellt. Zertifikate helfen dabei, auf unsicheren Transportwegen öffentliche Schlüssel authentisch auszutauschen. Derartige Zertifikate sind eine wichtige Voraussetzung/ein wichtiges Hilfsmittel beim Einsatz von Public-Key-Kryptographie.
Zertifikate enthalten mindestens einen öffentlichen Schlüssel, den Namen (die Identität) der betreffenden Person oder Instanz und eine digitale Signatur, die den Urheber dieser Bestätigung nachprüfbar macht und zugleich unbemerkte Modifikationen des Zertifikates verhindert (Authentizitäts- und Integritätsschutz). Darüber hinaus können Zertifikate noch diverse andere zusätzliche Bestandteile umfassen.



Figur 4.2 Struktur eines X 509 Zertifikats

In Abbildung 4.2 (die Struktur eines X.509 Zertifikates) sind die möglichen zusätzlichen Einträge ersichtlich, welche ein Zertifikat beinhalten kann wie

- Seriennummer des Zertifikates (Serial Number)
- Algorithmus, der zum Signieren des Zertifikates benutzt wurde (SignatureAlgorithm)
- Gültigkeitsdauer bzw. frühestes Gültigkeitsdatum sowie Verfallsdatum des Zertifikates (Validity)
- Zusätzliche Angaben über den Zertifikatnehmer (Distinguished Name)

Um einem Zertifikat die gewünschte Glaubwürdigkeit zu verschaffen, werden diese von einer vertrauenswürdigen 3. Partei, einer sogenannten Certificate Authority (CA) signiert. Mit dem öffentlichen Schlüssel der CA kann die Richtigkeit des Zertifikates überprüft werden.
Muss ein Zertifikat innerhalb der Gültigkeitsdauer gesperrt werden, ist dieses in der Certificate Revokation List (CRL) einzutragen. Eine CRL ist eine Zertifikat ähnliche Datenstruktur, welche von der CA signiert wird und alle ungültigen Zertifikate in einer Liste speichert.

4.4 OpenSSL

OpenSSL [10] ist eine frei verfügbare Implementierung des SSL/TLS-Protokolls und bietet zusätzlich Funktionen zur Zertifikat-Verwaltung sowie verschiedene kryptographische Funktionen.. Das Paket umfaßt mehrere Applikationen, z.B. zur Erzeugung von Zertifikaten, von Zertifizierungsanträgen und zur Verschlüsselung für das in dieser Arbeit verwendeten X.509 Formats.

4.4.1 OpenSSL Konfigurations Datei

Um mit OpenSSL eine eigene Zertifizierungsstelle einrichten zu können, muss die Konfigurationsdatei `openssl.cnf` der eigenen CA angepasst werden. Im Anhang 15.2 befindet sich ein komplettes Beispiel eines solchen Konfigurations-Files.
Die Datei gliedert sich grob in vier Abschnitte:

- [ca] Erzeugung eines Zertifikats (signieren).
- [policy] Auflagen an das zu signierende Zertifikat
- [req] Erzeugung eines "Zertifizierungswunsches" (Request).
- [extensions] Erweiterungen, welche einem Zertifikat beim Signieren hinzugefügt werden

Auf die Abschnitte [ca] und [req] wird zugegriffen, wenn openssl mit dem Parameter ca bzw. req aufgerufen wird.

Sowohl `ca` als auch `req` bieten die Möglichkeit, über die Option -config die zu verwendende Konfigurationsdatei explizit anzugeben:

```
openssl ca/req –config <openssl.cnf>...
```

Am Anfang dieses Konfiguration-Files können verschiedene Parameter, beziehungsweise Variablen gesetzt werden, welche von allen Abschnitten verwendet werden.

```
path = $ENV::VPNDIR
RANDFILE = $path/src/utils/.rand
```

Certificate Authority (CA) Abschnitt

Es können mehrere solcher Abschnitte für verschiedene CAs definiert werden. Auf diese können beim Signieren von Zertifikaten (siehe unten) mit dem Parameter –name <Abschnitt> je nach CA zugegriffen werden.

Innerhalb von [ca] -Abschnitten werden drei Schlüsselwörter erkannt, die auf weitere Abschnitte in einer Konfigurationsdatei verweisen. Die Schlüsselwörter lauten:

- **x509_extensions**: Verweist auf einen Abschnitt, in dem Zertifikat-erweiterungen (Extensions) für neue Zertifikate festgelegt sind. In diesem Abschnitt könnten die Extensions für ein Benutzer- oder ein CA-Zertifikat festgelegt werden.
- **crl_extensions**: Verweist auf einen entsprechenden Abschnitt für Extensions die in eine Zertifikat-Widerrufliste (Certificate revocation list, CRL)
- **policy**: Weist auf einen Abschnitt in dem festgelegt wird, inwieweit der Distinguished-Name in einem zu signierenden Request mit dem des CA-Zertifikats übereinstimmen muß.

Zusätzlich werden verschiedene Verzeichnisse angegebne und einzelne Files spezifiziert, auf welche bei entsprechenden Aktionen zugegriffen , beziehungsweise erstellt werden.

```
[ Root_CA ]       # Abschnitt fuer eine Root CA

dir               = $path/tmp/ca      # Where everything is kept
certs             = $dir/certs        # Where the issued certs are
kept
crl_dir           = $dir/crl          # Where the issued crl are kept
databas           = $path/src/utils/index.txt    # database index
file.
new_certs_dir     = $dir/certs        # default place for new certs.

certificate       = $dir/RootCA.pem   # The CA certificate
serial            = $path/src/utils/serial   # The serial number
crl               = $dir/crl.pem      # The current CRL
private_key       = $dir/private/CAkey.pem       # The private key

x509_extensions   = ca_ext             # The extentions to add to the
cert

crl_extensions    = crl_ext           # Extensions to add to CRL

default_days      = 730               # how long to certify for
default_crl_days  = 30                # how long before next CRL
default_md        = md5               # which hash-alg. to use.
preserve          = no                # keep passed DN ordering
policy            = policy_match
```

Certificate Request Policy Abschnitt
In diesem Abschnitt werden verschiedene Certificate Request Policies definiert, welche beim Signieren eines Requests gelesen werden. Es werden ausgewählte Felder des Distinguished-Names eines Request eingeschränkt. Es gibt drei mögliche Einträge:

- match: Der Wert des Requests und der CA müssen übereinstimmen.
- supplied: Der Eintrag muss im Request vorhanden sein.
- optional: Kann im Request vorhanden sein oder auch nicht.

```
[ policy_match ]
countryName             = match
stateOrProvinceName             = optional
localityName            = optional
organizationName        = supplied
organizationalUnitName          = optional
commonName              = supplied
emailAddress            = optional
```

Certificate Request Abschnitt
Hier werden für die Distinguished-Name-Einträge welche beim Erstellen eines Requests benötigt werden, die Default-Werte und Längenbegrenzungen der Eingaben festgelegt. Einige dieser Felder, wie zum Beispiel der "common name", sind für alle Requests verschieden, während andere ("country name") für die meisten gleich sein werden.

Zusätzlich können Einträge zum Generieren des Private-Keys angegeben werden, falls dieser zusammen mit dem Request generiert werden soll.

```
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes

# The extentions to add to the self signed cert
x509_extensions         = ca_ext
countryName             = Country Name (2 letter code)
countryName_default     = CH
countryName_min         = 2
countryName_max         = 2

localityName            = Locality Name (eg, city)
localityName_default    = City

0.organizationName      = Organization Name (eg, company)
0.organizationName_default  = Organisation
organizationalUnitName  = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Unit
commonName              = Common Name (eg, YOUR name)
commonName_max          = 64

emailAddress            = Email Address
emailAddress_max        = 60
```

Extension Abschnitt
Mittels Zertifikaterweiterungen [11] wird der Verwendungszweck von Zertifikaten gesteuert. Üblicherweise werden diese Erweiterungen durch eine CA beim Signieren eines Request in das dann erstellte Zertifikat eingebunden.

Key Usage steuert den Verwendungszweck des zu einem Zertifikat gehörenden Schlüssels: z.B. darf ein Schlüssel nur zum Signieren von CRL´s, nur zur Daten-Verschlüsselung oder nur zum Unterschreiben verwendet werden.

Mittels Basic Constraints kann eine Anwendung erkennen, ob es sich um ein CA-Zertifikat handelt oder nicht. Diese Extension sollte in jedem CA-Zertifikat verwendet und als `critical` markiert werden.

Es können mehrere Extension Abschnitte definiert werden, auf welche in den verschiedenen Abschnitten verwiesen werden kann.

```
[ ca_ext ]
basicConstraints       = critical, CA:TRUE
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer:always
keyUsage               = CRLSign, keyCertSign
nsCertType             = sslCA, emailCA, objCA
subjectAltName         = email:copy
issuerAltName          = issuer:copy
crlDistributionPoints  = URI:http://www…./This.crl
nsBaseUrl              = https://www…./ca/
nsCaPolicyUrl          = http://www…./policy.html
nsComment              = This certificate is a Root CA Certificate
```

4.4.2 Zertifizieren mit OpenSSL

Nachdem das Konfigurations-File openssl.cnf und eine entsprechende CA erstellt wurden, kann mit dem Ausstellen von Zertifikaten sowie deren Verwaltung begonnen werden.

- Erzeugen eines Requests, zusammen mit dem dazugehörigen Private-Key
- Signieren eines Requests durch eine CA
- Unterhalten der Certificate Revocation List (CRL)

Diese Schritte können mit OpenSSL sehr einfach und effizient durchgeführt werden. Wie im vorherigen Abschnitt beschrieben, greift OpenSSL bei all den benötigten Befehlen auf das `openssl.cnf` zu, um für diese CA die gewünschten Parameter einzulesen. Diese kann mit dem Parameter `-config path/filename.cnf` angegeben werden, sodass für verschiedenen CAs auch verschiedene Konfigurationsdateien verwendet werden können.

Erstellen einer CA

Normalerweise werden die oben beschriebenen Zertifikaterweiterungen beim Signieren des Requests in das zu generierende Zertifikat eingefügt. Im Abschnitt [req] der Konfigurationsdatei muss somit ein Eintrag vorhanden sein, welcher die Erweiterungen angibt, welche eingefügt werden sollen, wenn eine selbstsignierte CA generiert wird.

```
openssl req -config <file> -new -x509 -days <duration>
            -newkey rsa:<length> -keyout CAkey.pem -out RootCA.pem
```

Mit den Parametern `-new -x509` wird angegeben, dass eine selbstsignierte X.509-CA erstellt werden soll. Zusätzlich wird der dazugehörige RSA-Private-Key mit der angegebenen Länge generiert.

Erstellen eines Certifikate Requests
Um ein Zertifikat zu erstellen muss zuerst ein Certificate Request erstellt werden. Zusätzlich müssen noch Angaben zum Private-Key gemacht werden, welcher zusammen mit dem Request generiert wird.

```
openssl req –config <file> -newkey rsa:<length> -keyout User_Key.pem
           -out UserReq.pem
```

Für die Schlüsselerzeugung werden Zufallszahlen, beziehungsweise eine Reihe zufälliger Bytes benötigt um die Entropie[nnn] klein zu halten, damit ein möglichst sicherer Schlüssel (`-newkey`) generiert werden kann. Diese Zufallsdaten werden aus einem File gelesen, welches im `openssl.cnf` angeben werden muss.

Nach der Eingabe dieses Befehls müssen die Angaben, welche im Abschnitt [req_distinguished_name] definiert sind und mit der entsprechenden Policy übereinstimmen müssen, eingegeben werden. Falls nichts eingegeben wird, werden die Default-Werte eingesetzt.

Ein Certificate Request und der Private-Key bilden ein Paar, welches immer wieder benutzt werden kann und an keine bestimmte CA gebunden ist. Es kann somit von verschiedenen Zertifizierungsstellen signiert werden. Wenn ein Zertifikat abgelaufen ist, kann der dazugehörige Request wieder verwendet werden, um dieses zu erneuern.

Signieren von Certificate Requests
Beim Signieren eines Requests werden durch die CA möglicherweise einige Extensions in das Zertifikat eingefügt. Diese müssen im Konfigurations-File unter dem richtigen Abschnitt gesetzt werden. Dieser wird mit dem Parameter `–name Abschnitt` angegeben.

```
openssl ca –config <file> -name <paragraph> –days <duration>
          -md <hash-alg>-in UserReq.pem -out UserCert.pem
```

Den zu verwendenden Hash-Algorithmus für die Signatur kann mit dem Parameter `-md` angegeben werden. `–days` spezifiziert die Gültigkeitsdauer des zu erstellenden Zertifikates.
Nach der Eingabe dieses Befehls wird der Subject-Distinguished-Name ausgegeben und gefragt ob dieses Zertifikat wirklich signiert werden soll. Nach bestätigender Eingabe (yes) wird die "Datenbank" der CA (`index.txt`) aktualisiert und der Zähler der Seriennummern im File `serial` inkrementiert. Das erstellte Zertifikat wird unter dem Namen, welcher nach dem Parameter `–out` angegeben wurde, im PEM-Format (Base 64) gespeichert.

Certificate Revokation List (CRL)
Eine CRL wird aufgrund der Einträge im Index-File `index.txt` generiert. Ein Zertifikat wird nach erfolgreichem Signieren wie folgt eingetragen:

1)	2)	3)	4)	5)	6)
V	020130161602Z		01	unknown	/C=CH/L=ort/O=org/OU=unit/ CN=name/Email=email

1) Status des Zertifikates (V=valid, E=expired, R=revoked)
2) Ablaufdatum
3) Wiederrufdatum, wird eingetragen, wenn das Zertifikat gesperrt wird
4) Hexadezimale Seienummer
5) Wo das Zertifikat zu finden ist (immer auf "unknown" gesetzt).
6) Name des Zertikatinhabers (Distinguished-Name)

Soll nun ein Zertifikat gesperrt werden, wird mit dem folgenden Befehl, dieser Eintrag geändert.

```
openssl ca -config <file> -revoke UserCert.pem
```

Dies hat zur Folge, dass im Index-File das Zertifikat als gesperrt gekennzeichnet wird und dieses beim Generieren der CRL in die Liste eingetragen wird.

R	020130161602Z	011027224518Z	01	unknown	/C=CH/L=ort/O=org/OU=unit/ CN=name/Email=email

Wird nun eine CRL generiert, werden alle Einträge des Index-Files, welche mit einem "R" markiert sind in diese eingetragen und so die gesperrten Zertifikate einer bestimmten CA publiziert.

```
openssl ca -config <file> -gencrl -crldays <duration> -out crl.der
```

Der Parameter `-gencrl` gibt an. dass eine CRL generiert werden soll, welche eine gewisse Gültigkeit hat (`-crldays`) und in das File `crl.der (-out)` geschrieben werden soll.

4.5 SecureShell (ssh)

Remote-Dienste, wiersh, rlogin, rexec oder rcp sind im allgemeinen recht praktisch, aber unsicher in der Anwendung, da sie nicht nur Daten sondern auch das Benutzerkennwort und das Passwort unverschlüsselt übertragen. Heutzutage ist es schon fast unverantwortlich, diese Dienste zur Kommunikation über das Internet zu verwenden, denn sie sind anfällig auf sogenannt "man-in-the-middle"-Attacken. Einem Angreifer, der das Netz abhorcht und Login plus Kennwort abfangen kann, ist es zu einem späteren Zeitpunkt möglich sich auf den Server einzuloggen. Andererseits kann er, da diese Dienste keine Authentifizierung von Server unterstützen, sich dem Opfer als Server präsentieren und dessen gesendeten Daten empfangen.

Die SecureShell ist ein vollwertiger Ersatz für die remote-Dienste. Grundsätzlich ist die Verwendung der ssh-Utilities anderen Protokollen vorzuziehen, da bei der ssh bereits das Passwort nur verschlüsselt übertragen wird.Die Benutzung von ssh hat folgende Vorteile:

- Verschlüsselung des Datenverkehrs. Dadurch werden man-in-the-middle Attacken werden so gut wie unmöglich.
- Schutz gegen DNS- oder IP-Spoofing.
- Kompression des Datenverkehrs.
- Bietet die Möglichkeit, Dienste wie ftp, pop, PPP, usw. zu "tunneln".

Bei SecureShell [15] handelt es sich um ein Client/Server System, das RSA Schlüsselpaare zur Authentifizierung und zum Erstellen eines Session-Keys verwendet. Der eigentliche Datenaustausch wird dann mit einem Streamcypher, wie zum Beispiel IDEA, 3DES, Blowfish, etc. vershlüssert, welcher den Session-Key benutzt.
Ein Verbindungsaufbau läuft dabei wie folgt ab:

1. Der lokale ssh-Client auf dem Arbeitsplatzrechner wendet sich an den Server (sshd) um eine Verbindung aufzunehmen.
2. Der Server schickt daraufhin seinen öffentlichen Serverschlüssel und seinen öffentlichen Hostschlüssel dem Client.
3. Der Client prüft, ob der geschickte Hostschlüssel mit demjenigen übereinstimmt, den er bereits schon in einer Liste von Hostschlüsseln vorliegen hat.
4. Falls dies der Fall ist, verschlüsselt der Client eine von ihm erzeugte Zufallszahl mit den beiden übergebenen öffentlichen Schlüsseln (Hostkey und Serverkey). Diese Zufallszahl dient im weiteren Verlauf als aktueller Session-Key.
5. Der Server entschlüsselt mit seinen geheimen Schlüsseln diese Zahl und verwendet diese Zahl als Session-Key für die weitere Kommunikation. Dieser Session-Key wird jede Stunde erneuert.
6. Nun erst übergibt der Client, verschlüsselt mit dem Session-Key, seine Userid und sein Passwort.

4.5.1 SecureCopy (scp)

Um Dateien über das Netz von einem Rechner zum anderen zu übertragen, wird traditionell der Dienst ftp benutzt. Auch dieser hat den Nachteil, dass Passwort und Daten unverschlüsselt über die Leitung gehen. Hier bietet das Programm scp, das Bestandteil des ssh-Paketes ist, eine gute Alternative. scp kann einzelne Dateien aber auch genze Verzeichnissstrukturen von und zu einem entfernten Rechner über das Netz kopieren.

4.5.2 Tunneling

Eine bestehende ssh-Verbindung ist auch in der Lage, andere Protokolle über den verschlüsselten Kanal umzuleiten. Besonders einfach wird das Umleiten von Programmen, die unter dem X-Window-System laufen. Wo man normalerweise nach dem Einloggen per telnet umständlich mit `xhost` oder besser `xauth` und der Environmentvariable `DISPLAY` hantieren muss, ehe man ein grafisches Programm auf dem entfernten Rechner aufgerufen und auf dem lokalen Bildschirm dargestellt werden kann, setzt ssh alles automatisch und verschlüsselt die Übertragung obendrein.

4.5.3 Generieren von RSA Schlüsselpaaren

Um ssh benutzen zu können, muss man zu erst ein RAS Schlüsselpaar generieren. Dies macht man mit dem Tool `ssh-keygen`, welches sich im selben Verzeichnis wie ssh selber befindet. Das genaue Vorgehen wird hier kurz beschrieben:

1. Das Tool mit `ssh-keygen` starten.
2. Das Programm generiert ein Schlüsselpaar und testet es auch zugleich auf Fehler. Danach wird der User gefragt, wo die Schlüssel abgelegt werden sollen. Das vorgeschlagene Default-Verzeichnis ist sicher keine schlechte Wahl, da andere Literatur ebenfalls auf dieses Verzeichnis verweisen.
3. Der User wird aufgefordert einen Passphrase einzugeben und dieser nochmals zu bestätigen. Der Passphrase muss mindestens 12 Zeichen lang sein. Dieses Passwort muss immer angegeben werden, wenn ssh benutzt wird.

Es werden zwei Files in dem unter Schritt 2 angegebenen Verzeichnis angelegt: `identity`, enthält den private Key, und `identity.pub`, enthält den pulbic Key.

4. Der public Key aus der Datei `identity.pub` muss nun auf allen Rechnern, auf welche man von der lokalen Maschine aus zugreifen will, an die Datei `authorized_keys` angehängt werden.
5. Will man nicht nur vom lokalen Rechner auf die anderen Machinen zugreifen können, kann man das File `identity`, welches den private Key enthält, auch auf andere Rechner ins Verszeichnis `.ssh` kopieren.

Achtung: Die Verteilung der Schlüssel in den Schritten 4 und 5 ist nicht ganz unproblematisch. Da zu diesem Zeitpunkt noch keine sichere Verbindung besteht, kann jemand, der den Public Key abfängt dem Benutzer vorgaukeln, einer der vertrauenswürdigen Hosts zu sein, da der Benutzer ja korrekt einloggen kann. Noch schlimmer ist es, wenn es einem Angreifer gelingt, den private Key zu ergattern. Falls er zudem noch im Besitz des Passwortes ist, kann er sich bei den Hosts einloggen. Die beste Methode diese Schlüssel zu verteilen ist weiterhin die gute, alte Diskette.

4.5.4 Verbindung ohne Passphrase

Wird ssh oder scp aus einem Skript heraus benutzt, kann kein Passphrase eingetippt werden. Da es jedoch vorkommen kann, das man zum Beispiel einen Kopiervorgang in einem Skript per `cron` oder `at` automatisieren will, besteht die Möglichkeit, auf dem Zielrechner einen Eintrag in die Datei `~/.shosts` zu schreiben. Ein solcher Eintrag besteht aus einer Zeile, welche zuerst den Rechner, von welchem die Verbindung aufgebaut werden soll, und den Loginnamen des Benutzers, spezifiziert:

```
remotehost remoteuser
```

Bekommt ein Host nun eine Anfrage von einem Rechner, vergleicht er die Einträge dieses Files mit Adresse und dem Usernamen der anfragenden Maschine. Findet er eine Übereinstimmung, lässt er eine Verbindung ohne Eingabe des Passwortes zu.

Sollte eine passwortlose Verbindung aus irgend einem Grund nicht zu stande kommen, kann ssh zusätzlich mit der Option `-v` gestartet werden. Mit dieser Option gibt ssh zusätzlich genauere Debuginformationen aus.

4.5.5 Versionen

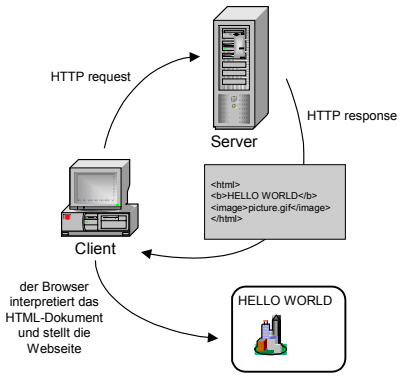
Nachdem lange Jahre das SSH-Protokoll in der Version 1 gute Dienste leistete, steht seit neuerem die Protokollversion 2 zur Verfügung. Ein SSH2-Client kann in der Regel auch Verbindungen mit einem SSH1-Server aufnehmen, und ein SSH2-Server startet bei Kontakt mit einem SSH1-Client den erforderlichen SSH1-Server automatisch, so dass wechselseitige Abwärtskompatibilität gewährleistet ist. Mit der Protokollversion 2 sind einige neue Features realisiert worden wie SecureFTP und einige interne Änderungen zum Protokollieren und Abwehren von feindlichen Angriffen.

4.6 PHP

In den Anfängen des World Wide Webs wurde in der Forschung HTML dazu benutzt, Resultate und Abhandlungen möglichst einfach und schnell unter Wissenschaftlern zu verbreiten. Einer der grössten Vorteile war, dass über sogenannte Links auf andere Seiten im Web verweisen kann. Da mathematische Fromeln, physikalische Tabellen und anderer wissenschaftlicher Kontext sich nicht ändert, ist es kein Problem, Seiten mit HTML brauchbar zu gestalten. Doch wenn sich der Inhalt auf einer Seite regelmässig ändert (zum Beispiel ein Newsticker) oder wenn der Besucher der Seite interaktiv den Inhalt der Seite bestimmen können soll, ist HTML nicht die richtige Methode um eine Web-Page zu gestallten, da HTML-Seiten statisch sind.
Im Gegensatz zu HTML eignet sich PHP [13] hervorragen um Webseiten mit dynamischem Inhalt zu kreieren. PHP (PHP Hypertext Preprozessor) ist eine server-seitige Skriptsprache und kann in normale HTML-Dokumente eingebettet werden. So wird eine Website zur Applikation. Die beiden folgenden Abbildungen sollen den Unterschied zwischen HTML und PHP[1] erklären.

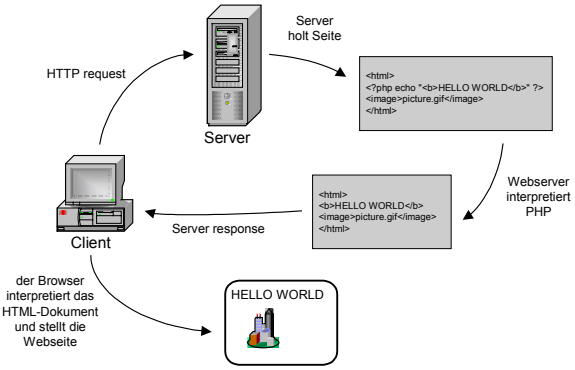
Was geschieht mit HTML-Seiten?
Der Server bekommt eine Seitenanfrage. Drei Schritte werden ausgeführt:

- Server liest die Anfrage vom Browser.
- Server holt die Seite.
- Server schickt das Dokument über das Internet zum Browser zurück.



Was geschieht mit PHP-Seiten?
Mit PHP wird nun ein zusätzlicher Schritt eingefügt. Anstatt einfach eine statische Seite zum Client zu schicken, handelt der Server den PHP-Code ab, je nach Resultat führt er Änderungen an der HTML-Seite durch und schickt diese an den Browser.

- Server liest die Anfrage vom Browser.
- Server holt die Seite.
- Server arbeitet PHP-Code ab und kreiert eine HTML-Seite.
- Server schickt das HTML-Dokument über das Internet zum Browser.



4.6.1 PHP versus CGI

CGI (Common Gateway Interface) ist die weitaus am meisten verbreitete serverseitige Skriptsprche im Web. Obwohl ein CGI-Skript in den verschiedensten Sprachen geschrieben werden kann, ist Perl die am häufigsten angewendete um CGI-Skripts zu programmieren. Ein Vorteil von CGI gegenüber PHP ist die Geschwindigkeit: Bei CGI wird für jede Anfrage aus dem Web ein neuer Prozess gestartet. Diese Tatsache kann aber auch schnell zum Nachteil von CGI werden, denn bei einer Vielzahl von Anfragen kann der Server an den Anschlag kommen.

4.6.2 Strukturierungsmöglichkeiten in PHP

Die Sprache von PHP ist sehr einfach gehalten. Für kleinere und mittlere Projekte sind ihre Möglichkeiten zur strukturierten Programmierung ausreichend. Will man jedoch größere Anwendungen implementieren, erreicht man ihre Grenzen. Zwar existieren Funktionen zum Verwalten von Namespaces, und mit Includes lässt sich der Code organisieren, Unterstützung für objektorientierte Programmierung oder erweiterte Möglichkeiten der Fehlerbehandlung sucht man aber vergebens.

4.6.3 Sessions in PHP

Wer in PHP mehrere Skripte für eine Webseite programmiert, kennt sicher das Problem: Geht der Anwender von einer Seite zur nächsten, sind alle Variablen für das nächste Skript verloren, es sei denn, sie wurden über ein Formular oder mit der URL zur nächsten Seite übertragen. Diese Übertragung ist jedoch nicht nur lästig, sondern auch fehleranfällig und unsicher. URL-Daten können leicht manipuliert werden, und auch Hidden-Felder in Formularen sind vor Manipulationen nicht sicher.
Der Grund für diesen Zustand ist, dass HTTP ein Protokoll ohne Status ist. Während ein Programm auf dem eigenen PC immer "weiss", welcher Benutzer das Programm gerade bedient, kennt der Webserver den Benutzer nicht. Daher muss sich der Entwickler darum bemühen, den Anwender für das Skript erkennbar zu machen.
Um nun Daten über mehrere Seiten hinweg zu speichern, ist eine Session in PHP ideal geeignet. Das Grundprinzip ist ebenso einfach wie genial: Alle Daten, die über mehrere Seiten gespeichert werden sollen, werden auf der Festplatte des Webservers zwischengespeichert. Für jeden Anwender wird dabei eine eigene Datei verwendet. Damit der Webserver die Daten dem richtigen Anwender wieder zuordnen kann, wird eine Session-ID erzeugt, mit der sich die entsprechende Datei wiederfinden lässt. Nun muss nur noch diese eine Session-ID von Seite zu Seite übergeben werden.
Damit hat sich im Prinzip nicht sehr viel geändert, die Anforderungen sind jedoch einfacher geworden. Statt einer Reihe von Daten muss nur noch eine ID von Seite zu Seite gegeben werden. Diese ID ist so gestaltet, dass sie nur schwer manipulierbar ist. Damit ist sichergestellt, dass niemand auf die Daten eines anderen zugreifen kann, der zur gleichen Zeit online ist. Da auch keine einzelnen Daten mehr übertragen werden müssen, können diese auch nicht mehr zwischen den Seitenabrufen manipuliert werden.

Standardmässig wird die Session-ID in einem Cookie gespeichert. Ist dieses Cookie noch nicht vorhanden (das heisst, die Session wird auf der Seite zum ersten Mal verwendet), wird es automatisch angelegt.
Bei Formularen wird per GET oder POST automatisch die Session-ID mit übergeben. PHP verändert den HTML-Code entsprechend, so dass dies gewährleistet ist. Ausserdem ändert PHP jeden HTML-Link so ab, dass PHPSESSID als Parameter mit der passenden Session-ID übergeben wird. Diese Fallback-Strategie wird jedoch nur dann verwendet, wenn kein Cookie zur Verfügung steht.

5 System Konzept

5.1 Zielsetzung

5.1.1 Allgemeine Beschreibung

In dieser Diplomarbeit soll ein Administrations-Tool realisiert werden, welches Konfigurations Daten von Security Gateways und deren Verbindungen konsistent verwaltet.
Die Funktionalität soll das Einfügen, Ändern und Löschen von Knoten (Security Gateways), zusammen mit den dazugehörigen Daten erlauben. Des weiteren können mit den in der Datenbank vorhandenen Knoten, sowie neu hinzugefügten, ein Netz von IPSec-Tunnels definiert werden, für welche die entsprechenden Konfigurations-Dateien erstellt werden. Die koordinierte Aktualisierung der Verbindungen durch diese Skripts erfolgt nachdem diese auf die einzelnen Knoten kryptografisch sicher verteilt worden sind.
Es sollen verschiedene Datenbanken an diese Applikation gebunden werden können, ohne den gesamten Code ändern zu müssen. Im weiteren sollten verschiedene IpSec-Implementationen unterstützt werden, welche als Filter eine Schnittstelle implementieren und so die gewünschten Skripts generieren.

5.1.2 Ziele und Wünsche

Die Verwaltung der Verbindungen lässt drei verschiedene Typen zu:

- Volle Vernetzung (full-mesh): jeder Knoten ist mit jedem Knoten mit einem IPSec Tunnel verbunden.
- Zentrale Vernetzung (hub): Alle Verbindungen gehen von einem zentralen Knoten aus. Keine Verbindungen zwischen den teilnehmenden Knoten.
- Punkt-zu-Punkt Verbindung (p-t-p): Ein Tunnel zwischen zwei Knoten.

Es soll in einem ersten Schritt die Linux IPSec Implementation (FreeS/WAN) unterstützt werden. Die Daten werden auf der Basis von XML unabhängig dargestellt, wodurch andere Module für verschiedene IPSec Implementiertungen einfach hinzugefügt werden können.
Das Web-Interface soll eine effiziente Eingabe ermöglichen, welche zusammen mit der dahinterliegenden Datenbank die Konsistenz der Daten gewährleistet.
Das Datenbank-Interface soll generisch sein, d.h. die momentane Datenbank kann durch eine Datenbank anderen Herstellers ersetzt werden, ohne Anpassungen an der Software vornehmen zu müssen.
Die Verteilung der Konfigurationsdateien soll auf der Basis von ssh/scp durch eine kryptografisch sichere Verbindungen erfolgen.
Das Bearbeiten von vorhandenen Netzen, sowie Änderungen in der Datenbank sollten geloggt werden.

Wünsche
Implementation weiterer Module zur Generierung verschiedener IPSec-Konfigurations Dateien. (Windows 2000, Check-Point)
Unterstützung von virtuellen Interfaces. Konfigurationsdateien auf die Server verteilen, diese aber für das virtuelle Interface konfigurieren.

Abgrenzungen
Diese Arbeit beschränkt sich auf die Implementation der Logik und das Generieren der Konfigurations-Files. In einem weiteren Schritt könnte dieses Tool um eine grafische Darstellung der verschiedenen Netze erweitert werden. Dies ist nicht Teil dieser Version.

5.2 Funktionsumfang

Aufgaben / funktionale Anforderungen
Folgende Aufgaben müssen realisiert werden:

- Verwaltung der einzelnen Knoten (INSERT, UPDATE, DELETE)
- Erzeugung der gewünschten Konfigurations-Dateien (FULL-MESH, HUB, P-T-P)
- Erzeugung eines X509-Zertifikates für einen neuen Knoten (openSSL)
- Logging der verrichteten Tätigkeiten mit diesem Tool (wer hat wann, was gemacht)
- Verteilen der Konfigurations-Files auf kryprografisch sicheren Verbindungen (ssh/scp)

Um die Trennung zwischen der Datenbank, der Logik und der Darstellung der Konfigurations-Files zu garantieren, werden die relevanten Daten in XML-Form dargestellt. Dies ermöglicht eine einheitliche Schnittstelle zwischen den Daten und deren Darstellung, beziehungsweise weiteren Verwendung.

Benutzungsschnittstellen
Das Interface zum Benutzer zur Administration dieser Knoten wird als Web-Interface implementiert.

Die Schnittstelle für den Zugriff auf Daten wird mittels XML definiert. Um die gewünschten Daten, je nach IPSec-Typ, zu lesen, muss das XML-File geparsed werden.

Grenzen und Einschränkungen
Um Zertifikate generieren zu können, wird openSSL , auf den Platformen UNIX und Linux verwendet. Dies beschränkt die gesamte Anwendung auf diese Betriebssysteme.

Konfiguration, Aufbaustufen, Varianten
Es ist die volle Funktionalität zu implementieren. Die als Wunschanforderungen beschriebenen Punkte können in der ersten Version fehlen. Falls die Zeit reicht, kann noch ein weiteres Modul, zum Beispiel für Windows 2000 implementiert werden.

In einer weiteren Version könnte eine Darstellung der konfigurierten Netze implementiert werden.

Kompatibilität, Portabilität
Dadurch, dass diese Applikation in PHP implementiert wird, kann diese auf verschiedenen Web-Servern eingesetzt werden. Voraussetzung ist die Unterstützung von PHP.

Da die Datenbankanbindung generisch implementiert wird, kann diese mit verschiedenen Datenbanken eingesetzt werden. Da verschiedene Datenbanken, verschiedene Datentypen definieren, kann es sein, dass einzelne Anpassungen an den SQL-Abfragen vorgenommen werden müssen.

5.3 Lösungsvarianten

5.3.1 Grafisches User Interface (GUI)

Aus der Aufgabenstellung der Arbeit geht hervor, dass mit der Datenbank per Webinterface kommuniziert werden und dessen Anbindung über PHP erfolgen soll. Zudem soll das GUI möglichst übersichtlich gestaltet und intuitiv zu bedienen sein. Aus der Analyse der gewünschten Funktionalitäten geht hervor, dass es Sinn macht die Navigation durch das GUI zweistufig zu gestallten und die Funktionalitäten in verschiedene Bereiche zu unterteilen, welche sich farblich voneinander unterscheiden. In den Bereichen wiederum werden dem Benutzer einzelne Masken, welche mit HTML Form Elementen aufgebaut sind präsentiert. Es muss ohne grosse Umwege möglich sein, in alle Bereiche zu gelangen und die verschiedenen Masken auszuwählen. Die Aufteilung in die verschiedenen Bereiche ist die folgende:

- insert: Daten in die Datenbank einfügen oder Daten generieren.
- update: Bestehende Datensätze abändern.
- delete: Datensätze löschen.
- show: Anzeigen der Daten.

Bei allen Manipulationen der Datenbank muss die Konsistenz der Daten gewährleistet sein. Dies wird zum einen bereits bei der Eingabe der Daten in die HTML Form Elemente und zum anderen beim Bestätigen der Eingaben erreicht. Sollte eine Manipulation der Daten zu einer Inkonsistenz führen, wird der Vorgang abgebrochen und dem Benutzer mitgeteilt, was genau zu einer Verletzung der Datenkonsistenz führen würde.

Zu Beginn wurde die Bedingung gestellt, ein GUI zu kreieren, welches ohne Frames auskommt und mit Tabellen aufgebaut ist. Es zeigte sich jedoch schnell, dass es relativ kompliziert ist eine zweistufige Navigation ohne Frames zu implementieren. Zudem stellen Frames für die Webbrowser der neueren Generation kein Problem mehr dar. Darum wurde das GUI trotzdem mit Frames erstellt.

5.3.2 XML-Schnittstelle

Um eine möglichst grosse Flexibilität zu erreichen wurde ein XML-File als Schnittstelle zwischen der Logik, das heisst zwischen dem korrekten Zusammenstellen der gewünschten Daten, und dem Filter, welches diese Daten in Skript-Files (Linux FreeS/WAN) oder Anleitungen für den Administrator (Check Point) entsprechend zusammenfasst. Dies soll die verschiedenen Filter möglichst einfach halten und deren Implementation vereinfachen.

Eine weitere Möglichkeit wäre, die gesamte Logik in die einzelnen Filter zu legen, um so nur die gewünschten Daten, welche für die einzelnen IPSec Implementationen nötig sind, zusammenzustellen und direkt von der Datenbank in die entsprechenden Files zu schreiben. Dies würde den Aufwand ein Filter für weitere IPSec Anwendungen erhöhen, und die Flexibilität, zum Beispiel welche Datenbank verwendet wird, eingeschränkt.

Es wurde die Variante mit einem XML-File als Schnittstelle gewählt. Der Vorteil dieses Modells liegt darin, dass die Logik gekapselt wird und so eine definierte Schnittstelle für die Filter vorhanden ist. Ausserdem wird mit dem Database Abstract Layer (DAL) die Unabhängigkeit der benutzten Datenbank gewährleistet. Zusätzlich sind die aktuellen Informationen eines bestimmten Knoten einfach aus dem XML-File ersichtlich und auswertbar.

Ein Filter für eine weitere IPSec Implementation kann ebenfalls ohne grösseren Aufwand in einer anderen Programmiersprache geschrieben werden da für die gängigsten Programmiersprachen XML-Parser-Funktionen schon vorhanden sind und nicht seperat noch implementiert werden müssen.

Die in der Datenbank gespeicherten Daten werden zusammen mit dem XML-File nochmals in der Datenbank abgeelegt und sind so in gewisser Weise redundant. Da aber genügend Ressourcen zur Verfügung stehen und die Speicherkapazitäten heutzutage immer grösser und günstiger werden, kann dieser Punkt ausser Acht gelassen werden.

5.3.3 Datenbank Unabhängigkeit

Die Datenbankzugriffe werden mit PHP vorgenommen. PHP bietet zu verschiedenen Datenbanken spezifische Funktionen um effizient auf diese zuzugreifen und Abfragen auszuführen. Um die gewünschte Datenbankunabhängigkeit zu erreichen, wurde zwischen der Datenbank und der darauf zugreifenden Komponenten eine einheitliche Schnittstelle, einen sogenannten Database Abstraction Layer (DAL) eingeführt. Dieser verfügt über verschiedene Treiber, welche die Hersteller spezifischen PHP-Funktionen implementieren.

PHP bietet auch Funktionen um auf ODBC (Open Database Connectivity)-Treiber zuzugreifen. Mit den ODBC-Treibern können beliebige Datenbanken benützt werden, welche eine ODBC Bridge zur Verfügung stellen.

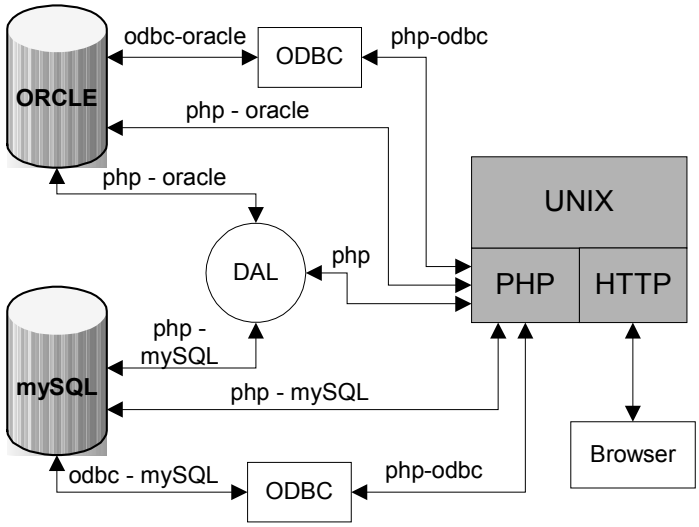


Abbildung 5.1 Mögliche Datenbank Anbindungen

Abbildung 5.1 zeigt die verschiedenen Möglichkeiten, um in diesem Fall auf eine Datenbank zuzugreifen, auf:

Direkt mit Datenbank spezifischen PHP-Funktionen

- Mit PHP-ODBC-Funktionen auf, vom Datenbank Hersteller zur Verfügung gestellte, ODBC-Bridge.
- Mit DAL-Funktionen, welcher die Datenbank spezifischen PHP-Funktionen, als Treiber implementiert.

Der Nachteil der ODBC-Variante ist der, dass es sich um eine proprietäre (Microsoft) API handelt. Dies ist in einer Unix-Umgebung, in welcher diese Applikation eingesetzt wird, nicht gern gesehen und bei der Datenbank müsste die dazugehörige ODBC-Bridge installiert werden. Letzteres wurde vom Datenbankadministrator kategorisch als sehr schlecht eingestuft und würde nur im äussersten Notfall durchgeführt.
Der eingesetzte DAL, welcher unter einer OpenSource BSD Lizenz frei verfügbar ist, stellte nur drei Treiber, für freie Datenbanken, wie mySQL oder Postgres, zur Verfügung. Es musste zuerst der Treiber für die Oracel Datenbank geschrieben werden, bevor dieser eingesetzt werden konnte.

5.3.4 Linux IPSec Filter

Da die Schnittstelle zwischen den Filtern und der Logik durch ein XML-File dargestellt wird, wäre es möglich gewesen, das mit der Applikation ausgelieferte Linux Free/SWAN Filter, in einer beliebigen Programmiersprache zu implementieren, welche XML-Parser Funktionen beinhaltet

5.3.5 Erstellen eines XML-Files (Meshing)

Es wurden zwei Möglichkeiten in Betracht gezogen:

1) Ein XML-File pro Meshing
2) Ein XML-File pro Knoten

Diese zwei Varianten haben nicht nur einen anderen Bezugspunkt, das Meshing selbst oder ein Knoten, es gibt auch Unterschiede in der Implementation der Filter. Die Anforderung an einen Filter soll möglichst klein sein.

Wenn ein XML-File für jedes Meshing erstellt wird, würde das Filter die Konfigurations Files für dieses Meshing erstellen, ohne andere Meshings, in welchen dieser Knoten beteiligt ist, zu berücksichtigen. Dies würde bedeuten, dass das Filter auf die Datenbank zugreifen müsste und die fehlenden Informationen herausfiltern müsste. Dies wiederum würde eine gewisse Logik zusätzlich zum Erstellen der einzelnen Files erfordern, was die Anforderungen an ein Filter erhöhen würde.
Ein XML-File für jeden Knoten bedeutet, dass alle Informationen, die benötigt werden, um für einen bestimmten Security Gateway die Konfigurations-Files zu generieren, vorhanden sind. Das Filter muss nur dieses interpretieren und ist von der Datenbank unabhängig.

Der erste Fall, ein XML-File pro Meshing würde dem gesamten Design wiedersprechen, mit einer gemeinsamen Schnittstelle, die Filter möglicht schlank zu halten und so das Erweitern dieser Applikation über diese Arbeit hinaus, zu vereinfachen. So wurde die zweite Variante gewählt, was eigentlich nicht mehr viel mit dem Erstellen eines Meshings zu tun hat. Von der ersten Überlegung ist deshalb nur noch der Name Meshing für die Funktion, welche das Zusammenstellen der Daten erledigt, übrig geblieben.

6 System Design

In diesem Kapitel wird das Design vorgestellt, sowie grobe Funktionsweise der einzelnen Komponente, beziehungsweiseKlassen und deren Schnittstellen nach Aussen beschrieben.

6.1 Modell

Wie in Abbildung 6.1. gezeigt wird, greift der Benutzer via WWW (World Wide Web) auf den Server zu, auf welchem die Applikation läuft. Diese wird in HTML mit PHP implementiert um auf die Datenbank zuzugreifen und ein XML File zu generieren, aus welchem die einzelnen Filter für die verschiedenen IPSec Implementationen die notwendigen Daten lesen können.

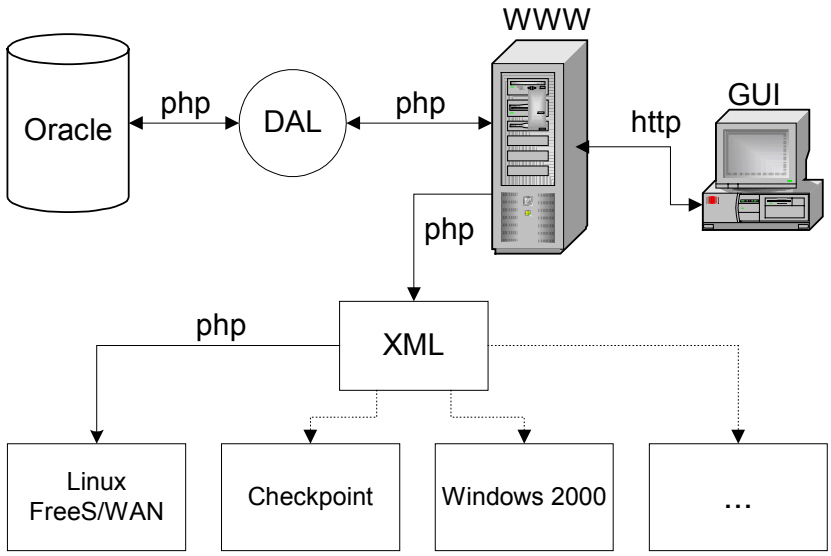


Abbildung 6.1 Komponenten Modell

Die gesamte Applikation wurde in 3 Hauptteile aufgeteilt:

- Relationale Datenbank, welche die gesamten Daten beinhaltet und durch ihr Design den Aufbau der Applikation weitgehend bestimmt.
- GUI, Interaktivität mit dem Benutzer
- Logik, Funktionen um Meshings zusammenzustellen

6.2 Datenbank

Das zentrale Element des VPNadmin-Tools ist die Datenbank, welche alle Informationen über ein Netzwerk und dessen Komponenten speichert. Im Rahmen dieser Dilpomarbeit wurde auf der Oracle8i Datenbank, welche die Firma Open Systems AG betreibt, ein Tablespace für diese Arbeit eingerichtet. Das Aufsetzen und das Verwalten der Datenbank, sowie das Einfügen des Datenbankschemas wurde vom Datenbankadministrator der Firma übernommen. Anpassungen der Datenstruktur konnten dennoch vorgenommen werden.

6.2.1 Datenbank Design

Im Folgenden werden die einzelnen Entitäten der Datenbank beschrieben. Das ER-Diagramm ist im Anhang ersichtlich. Alle Entitäten besitzen eine vortlaufende Sequenznummer als Primary-Key, Relationentabellen haben jeweils das Foreign-Key-Paar als Primary-Key. Nicht alle Attribute haben beim Erstellen der IPSec Konfigurationsfiles eine entscheidende Rolle, sondern dienen lediglich als Notizfelder. Die vollständige Liste aller Attribute ist ebenfalls dem Anhang zu entnehmen. Bei den wichtigeren Entitäten werden die Attribute und deren Eigenschaften trotzdem in einer Tabelle aufgeführt.

node

Attirbute	Type	NULL ?	Default	Foreign-Key
ID	nummer (11,0)	NOT NULL	0	
hostname	varchar2 (20)	NOT NULL	''	
location	varchar2 (50)	NULL	''	
Mgmt_IP1	varchar2 (20)	NOT NULL	''	
Mgmt_IP2	varchar2 (20)	NULL	''	
node_typeID	nummer (11,0)	NOT NULL	NULL	x

In dieser Entität werden die einzelnen Security Gateways gespeichert. Sie zählt zu den wichtigen Entitäten, da die Gateways in einem VPN die zentralen Geräte sind, um dessen Konfiguration es in diesem Tool geht. Zur genaueren Bestimmung, um was für ein Gerät es sich handelt, muss jedem Knoten ein Node-Type zugewiesen werden. Die management IP's 1 und 2 (`Mgmt_IP1` & `2`)werden für den Betrieb im VPN nicht benötigt. Sie dienen der Wartung des Gateways. Mindestens die management IP 1 muss in bei einem Eintrag angegeben werden. Diese Adressen werden auch in die xml-Files geschrieben, denn an sie werden die generierten IPSec Konfigurationfiles geschickt.

node_type

Mit Node-Type wird die Art des Konten spezifiziert (zum Beispiel Linux FreeS/WAN, CheckPoint, Windows2000, usw.). Diese Tabelle wird benötigt, wenn in einem näschten Schritt neue Filter implementiert werden und zwischen einzelnen IPSec Produkten unterschieden werden muss. In dieser Version werden nur Gateways, auf welchen Linux FreeS/WAN installiert ist, behandelt, die Einträge in diese Tabelle werden aber aus Konsistenzgründen trotzdem nachgeführt.

IPSec_algorithm

Die von den Node-Types unterstützten IPSec Algorithmen werden in dieser Tabelle aufgelistet und über die Relationstabelle `node_t_algorithm` den einzelnen Node-Types zugewiesen.

node_t_algorithm

Weist den verschiedenen Node-Typen die IPSec Algorithmen zu, welche von diesen unterstützt werden. Da ein Node-Type mehrere Algorithmen unterstützen kann und dieser Algorithmus auch in ganz anderen Node-Types implementiert sein kann, ist dies eine m:n-Relation.

protected_network

Die LAN's, welche durch die Knoten "geschützt" werden, sind in dieser Tabelle spezifiziert. Das Attribut `IP` enthält die Netzwerkadresse des LAN's, `netmask`, wie es der Name verrät, die Netzmaske.

protected_net_node

Diese Relationstabelle weist den verschiedenen Knoten ein oder mehrere LAN's zu. Es ist jedoch auch möglich, dass ein LAN mehrere Security Gateways hat. Darum ist die Beziehung zwischen den Protected Networks und den Gateways m:n.

config

Die beim Erstellen von Meshings generierten xml-Files werden in dieser Entität abgelegt. Es existiert pro Knoten, welcher in an einem Meshing beteiligt ist, genau ein aktuelles xml-File. Da anhand dieses Files Änderungen an den Meshings nachvollzogen werden können, werden die alten xml-Files nicht aus der Datenbank gelöscht. Damit man auch sieht wer die Änderungen wann vollzogen hat, wird der Name des Users zusammen mit dem Datum ebenfalls in der Tabelle abgelegt.

interface				
Attirbute	Type	NULL ?	Default	Foreign-Key
ID	nummer (11,0)	NOT NULL	0	
IP_address	varchar2 (15)	NOT NULL	"	
name	varchar2 (20)	NULL	"	
nodeID	nummer (11,0)	NOT NULL	NULL	x

Jeder Gateway kann mehrere Interfaces haben, welche hier spezifiziert werden. In `IP_address` wird die Adresse gespeichert, welche für die VPN-Verbindungen verwendet werden.

preshared_key

In einem Meshing mit Preshared-Keys müssen diese Schlüssel irgendwo gespeichert werden. Eine Möglichkeit währe gewesen diese in die Tabelle `key_set` einzutragen. Dies wurde aus folgenden Gründen nicht so gemacht: Pro Verbindung teilen sich immer nur zwei Interfaces den gleichen Schlüssel. Über den Foreign-Key von `key_set` könnte man die Preshared-Keys zwar den Interfaces zuweisen und man kann sogar mit einer Abfrage über die ganze Tabelle die beiden korrespondierenden Interfaces finden. Da ein Eintrag in die Tabelle `key_set` jedoch einen Foreign-Key auf `cert_request` erfordert, und Preshared-Keys mit Zertifikaten nichts zu tun haben, macht dies keinen Sinn. Aus diesem Grund wurde die Entität `preshared_key` in die Datenbank eingeführt. Die beiden Foreign-Keys auf die Interfaces dienen dieser Entität als Primary-Key.

CA				
Attirbute	Type	NULL ?	Default	Foreign-Key
ID	nummer (11,0)	NOT NULL	0	
name	varchar2 (20)	NOT NULL	''	
certificate	clob	NULL	NULL	
certificate_policy	varchar2 (200)	NULL	''	
description	varchar2 (200)	NULL	''	
last_serial	varchar2 (20)	NOT NULL	NULL	
index_file	clob	NULL	NULL	
openSSL_config	clob	NULL	NULL	
private_key	clob	NULL	NULL	
passphrase	varchar2 (20)	NOT NULL	''	

In der Entität `CA` werden alle Daten festgehalten, welche erstens die CA spezifizieren und zweitens zum signieren von Zertifikaten benötigt werden. Das Attribut `certificate` enthält das CA-Zertifikat. In `index_file` ist die eigentliche CRL der CA. In diesem File stehen alle ungültigen Zertifikate, welche von dieser CA signiert sind. Der Inhalt der Attribute `certificate_policy`, `last_serial`, `openSSL_config`, `private_key` und `passphrase` werden benötigt um Zertifikate signieren zu können. Wie dies genau gemacht wird, ist im Kapitel 4.4.2 beschrieben. Es stellt sich die Frage, warum das CA Zertifikat nicht in der Tabelle `certificate` und der Private-Key nicht in `key_set` gespeichert sind. Währe das CA Zerfifikat in `certificate` eingetragen, müsste der CA ein Interface und ein Key-Set zugewiesen werden und diesem wiederum ein Certificate Request. Eine CA hat aber nichts mit den Interfaces zu tun. Ein Certificate Request muss noch von einer CA unterzeichnet werden, was nicht geht, wenn die CA noch nicht in der Datenbank ist. Zudem sind CA Zertifikate selfsigned. Es ist also die praktikabelste Lösung den Private-Key und das CA Zertifikat direkt in `CA` einzutragen.

CRL

Dies sind nicht die eigentlichen Certificate Revocation Lists der CA's gespeichert, den die ungültigen Zertifikate werden im Index-File der jeweiligen CA gespeichert (siehe `CA`). In dieser Tabelle stehen aber ebenfalls alle ungültigen oder abgelaufenen Zertifikate.

certificate				
Attirbute	Type	NULL ?	Default	Foreign-Key
ID	nummer (11,0)	NOT NULL	0	
certificate	clob	NULL	NULL	
creation_date	date	NULL	'01-01-2001 00:00:00'	
expiration_date	date	NULL	'01-01-2001 00:00:00'	
subject_alt_name	varchar2(20)	NULL	NULL	
CAID	nummer (11,0)	NOT NULL	0	x
key_setID	nummer (11,0)	NOT NULL	0	x
interfaceID	nummer (11,0)	NOT NULL	0	x

cert_request

Der erste Schritt beim Generieren eines Zertifikates ist das Erzeugen eines Requests. Das Feld `cert_info` wird mit einem String gefüllt, in welchem die einzelnen Informationen durch zwei `##` getrennt sind. Diese Infos werden vom User beim Generieren angegeben.

key_set

Wird ein Zertifikate generiert, muss unter anderem ein Schlüsselpaar erzeugt und in dieser Tabelle gespeichert werden. Jedes Paar ist von einem bestimmten Typ und gehört zu einen Certificate Request. Zudem wird auch die Schlüssellänge und der bei der Generierung des korrespondierenden Zertifikats benötigte Passphrase abgelegt.

key_type

Jedes Schlüsselpaar liegt einem bestimmten Algorithmus zu Grunde. Die Namen der verschiedenen Algorithmen sind hier abgelegt.

meshing				
Attirbute	Type	NULL ?	Default	Foreign-Key
ID	nummer (11,0)	NOT NULL	0	
name	varchar2 (20)	NOT NULL	''	
IKE_lifetime	nummer (7,0)	NULL	0	
IPSec_mode	varchar2 (20)	NOT NULL	''	
IPSec_type	varchar2 (20)	NOT NULL	''	
pfs	varchar2 (10)	NULL	'yes'	
meshing_typeID	nummer (11,0)	NOT NULL	0	x
CAID	nummer (11,0)	NULL	0	x

Dies ist die Entität, in welcher die Meshings gespeichert werden. Die IKE-lifetime wird in Sekunden angegeben. Eine Besonderheit stellt das Attribut `CAID` dar: Obwohl es ein Foreign-Key ist, kann es `NULL` sein. Dies ist definitiv unschön, kommt aber aus der Überlegung heraus, dass dieses Attribut nur gesetzt wird, wenn der Benutzer will, dass alle im Meshing verwendeten Zeritfikate von der gleichen CA signiert worden sind. Hat dieses Feld den Wert `NULL`, dürfen Zertifikate von verschiedenen CA's benutzt werden. Aus dem ER-Diagramm ist ersichtlich, dass die Relationen zwischen den Entitäten `meshing`, `interface` und `certificate` einen Kreis bilden, der auf den ersten Blick als unnötig erscheint. Schliesslich lässt sich über die Relationstabelle `interface_meshing` auf die am Meshing beteiligten Interfaces schliessen und von diesen wiederum auf deren Zertifikate. Da nun aber ein Interface mehrere Zerifikate besitzen kann, kann nicht mehr bestimmt werden, welche Zertifikate an diesem Meshing beteiligt sind. Dies ist der Grund warum die Relationstabelle `certificate_meshing` geführt wird.

interface_meshing

Diese Tabelle ordnet den Meshings die verschiedenen Interfaces zu. Handelt es sich um ein Hub-Meshing, wird mit dem Attribut `central_interface` das zentrale Interface bestimmt:

`0` = normales Interface
`1` = Central-Interface dieses Meshings

meshing_type

Spezifiziert die drei möglichen Meshingarten: Full, Hub oder Point-to-Point.

6.3 Design GUI

Das Benutzerinterface ist in folgene Struktur aufgeteilt:

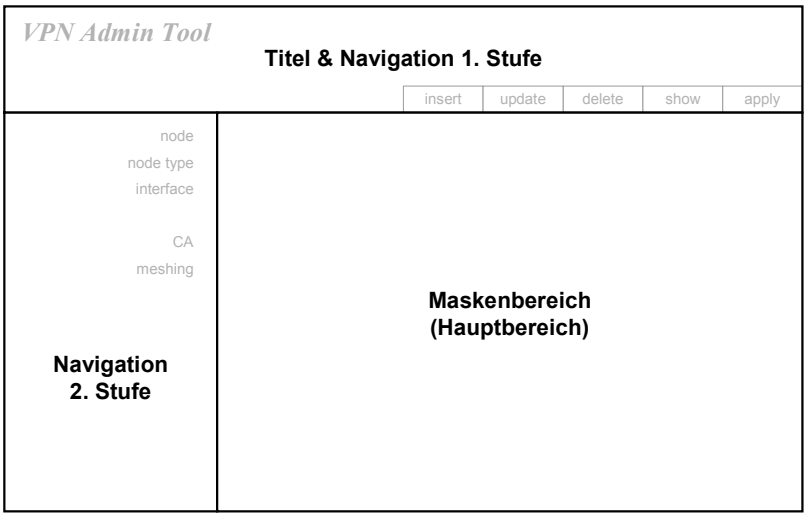


Abbildung 6.2 Layout Rahmen

Wird das GUI frisch geladen, so sieht der Benutzer am oberen Rand die Titelleiste, der Frame für die zweite Stufe der Navigation ist vorerst noch grau, der Hauptbereich schwarz. Zu Beginn der Arbeit war auch noch eine Login-Maske vorgesehen. Es stellte sich jedoch heraus, dass die Benutzer des Web-Servers von der Firma Open Systems sich speziell auf diesen einloggen müssen. Die Authentifizierung findet also nicht im VPNadmin Tool statt, der Username kann jedoch vom Web-Server übernommen und verwendet werden.
In der Titelleiste ist die erste Navigationsstufe untergebracht. Dieses Frame ändert sein Aussehen während der Bedienung nicht. Klickt man auf einen der fünf Navigationslinks, so wird im linken Frame die entsprechende Navigationsseite geladen, welche wiederum für das Laden der einzelnen Masken im Hauptteil verantwortlich sind.

Aufbau
1. Stufe
Auf dieser Stufe wird entschieden, was für eine Art von Tätigkeit durchgeführt werden soll. Aus Gründen der besseren Übersichtlichkeit werden die verschiedenen Links auch farblich voneinander unterschieden.

- insert
- update
- delete
- show

2. Stufe
Grob gesagt wird mit den Punkten, welche in dieser Stufe aufgeführt sind, auf die einzelnen Entitäten der Datenbank zugegriffen. Dies stimmt jedoch nur bedingt, denn wer die aufgeführten Menüpunkte mit dem ER-Diagramm aus dem Anhang vergleicht, stellt fest, dass es mehr als doppelt so viele Entitäten in der Datenbank gibt wie Menüpunkte. Dies kommt jedoch daher, dass gewisse Tabellen sich nicht oder äusserst selten ändern (siehe Kapitel Datenbank Design 6.2.1) oder dessen Manipulation von den implementierten Funktionen übernommen wird.

	insert	update	delete	show
node	x	x	x	x
node_type	x	x	x	x
interface	x	x	x	x
logging			x	x
protected_network	x	x	x	x
CA	x			x
certificate	x		x	x
revocation_list			x	x
cert_request				x
meshing	x	x	x	x

Mit diesen Masken lassen sich zwar alle Daten, welche für ein VPN nötig sind in die Datenbank eintragen, verändern oder löschen. Doch die Generierung der IPSec-Konfigurationsfiles aus den erstellten XML-Files kann nicht ausgelöst werden. Zu diesem Zweck gibt es in der ersten Stufe der Navigation den Link "apply", über welchen man direkt, das heisst ohne über die zweite Stufe der Navigation gehen zu müssen, in eine abschliessende Maske gelangt. Diese Maske präsentiert dem User einen Button welcher die Generierung der Konfigurationsfiles und deren Versand auslöst. Wie dies genau vor sich geht, ist im Kapitel 7.8 beschrieben.

6.4 Logik

Um die verschiedenen Aufgaben, wie das generieren eines Meshings, das Erstellen von Zertifikaten und das Parsen eines XML-Files, ausführen zu können, wurden verschiedenen Komponenten, beziehungsweise Klassen implementiert, welche diese Aufgaben übernehmen.

6.5 Database Abstract Layer (DAL)

Wie bereits im Kapitel 5 Analyse erwähnt, stellt der DAL die Schnittstelle zu verschiedenen Datenbanken dar. Diese Komponente wird wo immer auf die Datenbank zugegriffen werden muss, initialisiert. Dadurch wird ein einheitlicher Zugriff auf die Datenbank gewährleistet und für allfällige Änderungen der Datenbank, zum Beispiel Upgrade auf eine neuere Version oder Installation einer neuen Datenbank eines anderen Herstellers, muss lediglich der Treiber angepasst, oder eventuell ein neuer geschrieben werden.

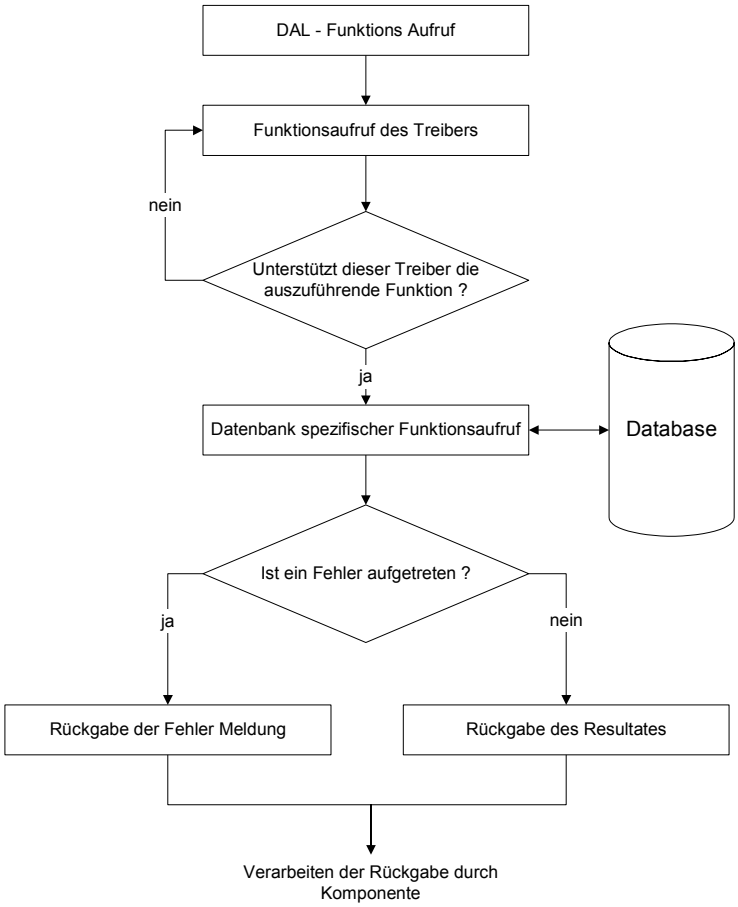


Abbildung 6.3: Ablauf eines DAL-Funktionsaufrufes

Nachdem die Komponente den DAL initialisiert hat, und den für den Zugriff auf die Datenbank richtigen Treiber ausgewählt hat, können die abstrakten Datenbank Funktionen aufgerufen werden.

Beim Aufruf einer Funktion prüft der DAL, ob der von der Komponente ausgewählte Treiber diese Funktion unterstützt und gibt entsprechend eine Fehlermeldung zurück oder ruft die Funktion des Treibers auf (siehe Abbildung 6.3). Diese übernimmt den Datenbank spezifische Aufruf und gibt je nach Typ, Ausführen einer Abfrage, Einlesen von Resultaten einer Abfrage, usw., nichts, eine Fehlermeldung oder das Resultat zurück.

6.6 XML Schnittstelle

Um die Trennung zwischen der Datenbank, der Logik und der Darstellung der Konfigurations-Files zu garantieren, werden die relevanten Daten in XML-Form dargestellt. Dies ermöglicht eine einheitliche Schnittstelle zwischen den Daten und deren Darstellung.
Für jeden einzelnen Knoten wird ein
XML-File generiert, welches in die drei Hauptteile HEAD, MESHING und CONNECTIONS aufgeteilt werden kann und alle Informationen über diesen beinhaltet:

- HEAD: Allgemeine Daten des Knoten
- MESHING: Alle Meshings in welchen dieser Knoten vorkommt
    - CONNCETIONS:
        - Alle Verbindungen der einzelnen Meshings dieses Knoten
        - Alle Interfaces mit welchen dieser in den Meshings beteiligt ist
        - Allfällige Zertifikate welche zur Authentisierung einzelner Verbindungen dienen.

Für jedes Meshing wird das Interface des Knotens, mit all den nötigen Parametern, sowie die Daten der einzelnen Verbindungspartnern in das XML-File geschrieben. Es werden nur die für die Erstellung der Konfigurations Skripts notwendigen Daten in diesem File dargestellt.

Falls für eine andere Anwendung (z.B. einer graphischen Darstellung der vorhandenen Netzen der IPSec Gateways) mehr Informationen nötig sind, kann ein eigenes XML-File generiert werden.

Auf der folgenden Seite ist der Aufbau eines XML-Files aufgezeigt. Die Parameter und die möglichen Werte sind in der Tabelle XML-Parameter im Anhang 15.2 spezifiziert.

```
<?xml version="1.0"?>
<!-- Created DATE -->
<node name="NAME">
<HEAD>
<mgmt_IP_1>IP-address-1</mgmt_IP_1>
<mgmt_IP_2>IP-address-2 | n/a</mgmt_IP_2>
<node_type name="NAME">
<os>os</os>
<version>version</version>
<ipsec_alg>algorithmus</ipsec_alg>
</node_type>
<protected_network>
<ip>ip-address</ip>
<netmask>netmask</netmask>
</protected_network>
</HEAD>
<MESHING name="NAME" type="TYPE">
<ipsec_type>certificate | preshared</ipsec_type>
<central_node>central-node| n/a</central_node>
<ipsec_mode>tunnel</ipsec_mode>
<ike_lifetime>231</ike_lifetime>
<pfs>no</pfs>
<interface ip="IP-ADRESSE" name="NAME">
<subject_alt_name>n/a</subject_alt_name>
<ca><![CDATA[BASE64 certificate | n/a]]></ca>
<crl><![CDATA[BASE64 certificate | n/a]]></crl>
<certificate><![CDATA[BASE64 certificate | n/a]]></certificate>
<passphrase>passphrase des private key</passphrase>
<private_key><![CDATA[BASE64 private-key | n/a]]></private_key>
<public_key>HEX public-key</public_key>
</interface>
<CONNECTIONS number="NUMBER">
<connection name="NAME">
<remote_ip>remote IP-address</remote_ip>
<remote_name>remote name</remote_name>
<remote_subject>remote netmask</remote_subject>
<protected_network>
<ip>remote protected network IP-address</ip>
<netmask> remote protected network netmask</netmask>
</protected_network>
<protected_network>
                                                    …
</protected_network>
<preshared_key>BASE64 preshared key | n/a</preshared_key>
</connection>
<connection name="NAME ">
                                                    …
</connection>
</CONNECTIONS>
</meshing>
<meshing name="NAME" type="TYPE">
                                                    …
</MESHING>
                    …
</node>
```

Abbildung 6.4: Struktur eins XML-Files

Um ein XML-File zu erstellen werden verschiedene Funktionen zur Verfügung gestellt, mit welchen die einzelnen Tags, Attribute und Werte gesetzt werden können. Da ein XML-File streng hierarchisch gegliedert ist, muss für jedes Element die Hierarchiestufe und das darüber liegende Element angegeben werden, beginnend mit einem sogenannten "Root" Element, welches vor dem Erstellen definiert werden muss.

Einem Element können beliebige Attribute, das heisst Name/Wert-Paare zugeordnet werden. Ein Element kann wiederum ein oder mehrere Elemente oder ein Wert enthalten (siehe Abbildung 6.4). Es ist möglich einem bestimmten Element auch das XML-interne Tag CDATA zuzuordnen, welches einem XML-Parser mitteilt, dass der Wert nicht interpretiert, sondern direkt an die Applikation weitergegeben werden soll.

Ein XML-Dokument kann mit sogenannten Stylesheets formatiert werden. Ein solches Dokument, wird von einem Parser gelesen und für bestimmte Tags werden weitere Informationen (Schrift, Schriftgrösse, usw.)aus einem Stylesheet gelesen, um die Daten zu formatieren. Dies wird hier nicht unterstützt, da ein Filter nur den Wert die Daten selbst benötigt.

6.7 Erstellen eines Meshing

Ein Meshing wird anhand der in der Datenbank gespeicherten Angaben erstellt. Es wird zwischen drei verschiedenen Arten eines zu erstellenden Meshings unterschieden:

- FULL: Jeder Knoten hat eine sichere IPSec Verbindung zu jedem anderen Knoten in diesem Netzwerk der Security Gateways.
- HUB: Alle Security Gateways haben genau eine Verbindung in deisem Netzwerk zu einem bestimmten Knoten, dem Central Interface.
- POINT: Eine Punkt zu Punkt Verbindung zweier Security Gateways

Zusätzlich wird innerhalb eines Meshings eine der beiden Authentifizierungs Methoden, Preshared-Key oder Zertifikate mit RSA Public-Key[11], gewählt. Es ist nicht möglich, verschiedene Authentifizierungsmechanismen innerhalb eines Meshings zu haben. Aufgrund dieser beiden Angaben und den in der Datenbank vorhandenen Daten, wird das XML-File generiert. (siehe Abbildung 6.4).

Wie in dieser Abbildung ersichtlich, gehören die Zertifikate jeweils zu einem Meshing und die Preshared Keys zu einer Verbindung. Somit müssen diese beiden Varianten der Authentifizierung in den jeweiligen Funktionen abgefragt und die entsprechenden Daten geladen und in das XML-File geschrieben werden.

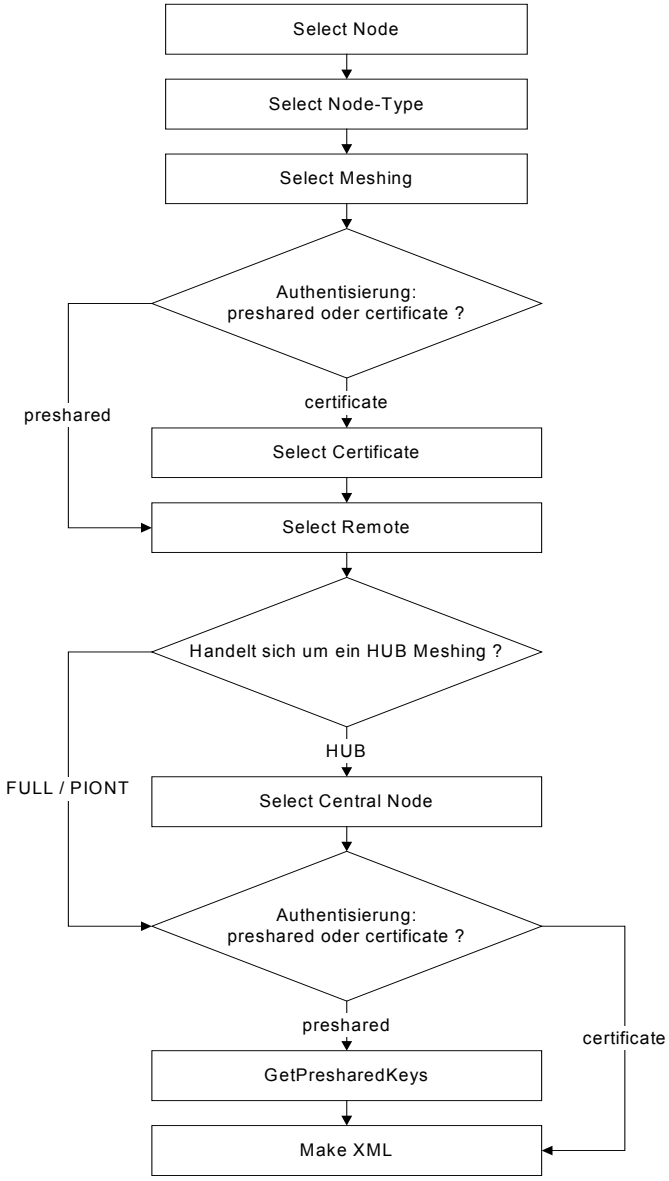


Abbildung 6.5: Zusammenstellen der Daten eines Meshings

Welche Daten in den einzelnen Funktionen aus der Datenbank gelesen werden:

- Select Node/Select Node Type: Daten des Knoten für welchen die Meshings erstellt werden.
- Select Meshing: Daten der zu erstellenden Meshings.
- Select Certificate: Das Zertifikat des jeweiligen Meshings.
- Select Remote: Daten der Verbindungspartner dieses Knoten in den jeweiligen Meshings.
- GetPresharedKey: Preshared Secret der jeweiligen Verbindung.

6.8 Linux IPSec-Filter

Das Filter für die Linux Implementation (FreeS/WAN) generiert die benötigten Konfigurations Dateien ipsec.conf und ipsec.secrets und schreibt die im XML-File vorhandenen Zertifikate und die dazugehörigen CAs in die entsprechenden Files.

Das Filter verwendet die Funktionen des XML-Parsers von PHP um die benötigten Daten zu parsen. Dieser stellt Funktionen zur Verfügung um die einzelnen Tags, deren Attribute und Werte aus dem XML-File zu extrahieren.Es werden zuerst alle Daten geparsed und zusammengetragen, bevor diese in die beiden Files `ipsec.conf` und `ipsec.secters` geschrieben werden.

Zusätzlich wird ein temporäres Verzeichnis erstellt, in welchem alle Files zwischengespeichert werden, bevor diese verschickt werden. Falls ein Meshing mit Zertifikaten authentifiziert, werden diese, sowie die dazugehörigen CAs in Files geschrieben. Die dazugehörigen CRLs werden ebenfalls generiert und zum Senden bereitgestellt.

6.9 Erstellung von Zertifikaten

Damit die Umgebung, in der diese Applikation eingesetzt wird, eine eigene Zertifikat Vertrauens Kette unterhalten kann, müssen Zertifikate generiert werden können und allenfalls auch gesperrt werden.

Voraussetzung für die Erstellung eines Zertifikates ist das Vorhandensein einer CA. Es können verschiedene CAs erstellt werden und so verschiedene Zertifikat Vertrauens Ketten für verschiedene Zwecke verwendet werden.

Das Erstellen von CAs, Certificate Requests und deren Siginierung wird mit openSSL[10] durchgeführt. Zuerst muss ein Zertifikat Request erstellt werden, welcher vom Root CA signiert ein funktionelles Zertifikat ergibt.

Die einzelnen CAs werden zusammen mit all den wichtigen Komponenten, dem Private Key, der aktuellen Seriennummer des zuletzt unterschriebenen Zertifikates, dem Index-File und dem für diese CA gültigen OpenSSL Konfigurationsfile, gespeichert.

6.10 Logging und Error Handling

Alle Aktivitäten, welche Änderungen der Daten in der Datenbank zur Folge haben müssen geloggt werden, damit nachvollzogen werden kann, wer und vor allem was geändert wurde.
Es werden alle Aktivitäten in ein Log-File geschrieben. Sollten Fehler auftreten werden diese in einem separaten Error-Log gespeichert und können so zusammen mit dem error_log des Apache Webservers Aufschluss über diesen geben.
Es werden zwei Klassen zur Verfügung gestellt, welche von allen Komponenten initialisiert werden. So kann an entsprechenden Stellen ein Eintrag in das Log-File geschrieben und Fehler können abgefangen und in das Error-Log-File geschrieben werden.

6.11 Verteilen der erstellten Files

Das ausgewählte Filter wird auf die gewünschten Knoten angewendet bevor die generierten Files an diesen per `scp` gesendet werden. Zuerst an die Management IP-Adresse 1 des Knotens mittels `scp` eine `ssh` Verbinung aufgebaut und die Files somit sicher kopiert. Danach wird überprüft, ob die Files kopiert wurden, indem auf dem Security Gateway und auf dem Server, für alle Files ein MD5-Hash berechnet wird und diese miteinander verglichen werden. Stimmen die Hash-Werte überein, wurden die Files korrekt kopiert.

Die SSH-Verbindung muss nicht überprüft werden, da Open Systems diese Verbindungen zu den Management IP-Adressen der Security Gateways ständig überprüft.

7 Implementation

In diesem Kapitel werden die Schnittstellen und die Funktionsweise der einzelnen Komponenten beschrieben. Es wird ein kurzer Überblick über die Komponente gegeben, die Schnittstellen, welche nach aussen sichtbar sind bezeichnet, sowie die interne Funktionsweise erläutert.

7.1 GUI

Das webbasierte Benutzerinterface wurde bewusst so simpel wie möglich gehalten. Alle Dateien des GUI sind mit dem Programm Macromedia Dreamweaver 4 unter Windows 2000 als reine HTML-Files entworfen worden. Die meisten Files wurden jedoch mit PHP-Code erweitert. Zu diesem Zweck wurde Quanta 2 unter SuSE Linux 7.2 verwendet. Getestet wurden das GUI hauptsächlich mit dem KDE-Browser Konqueror. Auf die Problematik der uneinheitlichen Interpretation von HTML-Code in den verschiedenen Browsern wurde nicht eingegangen. Zwar wurde die Funktionalität des GUI's mit verschiedenden Browsern getestet, die völlig unterschiedliche Darstellung wurde bewusst ausser Acht gelassen, da dies den Zeitrahmen der Diplomarbeit bei weitem gesprengt hätte.

Das Benutzerinterface wird aus drei verschiedenen Frames zusammen gesetzt (siehe Design GUI), welche in das Frameset der Datei `index.php` geladen werden. Die Frames `title` und `navigation` dienen beide zur Navigation durch das GUI. Die Masken, welche in das Frame `main` geladen werden, bestehen aus zwei Teilen: Einem darstellenden Teil und einem ausführenden Teil. Jeder Teil ist ein eigenes PHP-File, wobei der Dateiname jeweils Aufschluss gibt, ob es sich um den darstellenden oder den ausführenden Teil einer Maske handelt. Zusätzlich kann dem Namen noch entnommen werden, in welchen Bereich die Maske gehört. Ein Beispiel: Die Datei `insert_node.php` beinhaltet den darstellenden Teil der Maske mit welcher einzelne Security Gateways in die Datenbank eingefügt werden können, die Datei `insert_node_action.php` ist der korespondierende Ausführungsteil. Folgende Abbildung stellt diese Trennung grafisch dar.

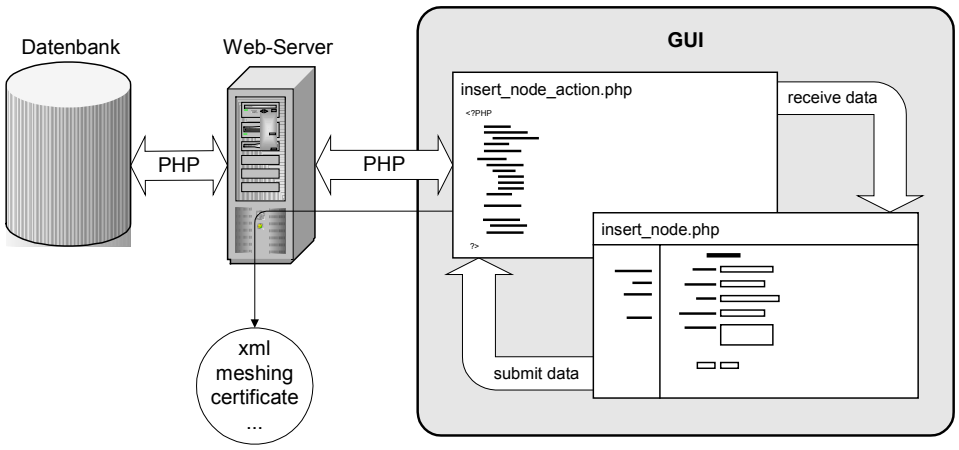


Abbildung 7.1 Aktionsablauf des GUI

Obwohl die Trennung von Darstellung und Ausführung weitgehend eingehalten wurde, enthalten die darstellenden Files neben dem HTML- auch PHP-Code, welcher aber nur dazu dient, Rückgabewerte vom Aktionsteil darzustellen. Dies kann entweder eine einfache Ausgabe eines Variableninhalts oder aber eine Datenbankabfrage zum Füllen einer Auswahlliste sein.

Der Abbildung ist zusätzlich zu entnehmen, dass weitere Funktionen aus dem Ausführungsteil heraus aufgerufen werden.

Daten, welche vom Benutzer in ein Formular eingefüllt werden, werden mittels post-Methode, an den Server geschickt, wie dies bei normalen HTML-Formularen der Fall ist. Daten, an das aufrufende PHP-File werden in normalen PHP-Variablen zurückgegeben. Dies funktioniert, da das aufrufende File mit include eingebunden wird. Der Scope der Variablen bleibt so erhalten. Müssen Daten an ein File übergeben werden, welches nicht per include-Anweisung eingebunden, sondern mit JavaScript geladen wird, funktioniert dies nicht so einfach. Die Variablen verlieren ihren Gültigkeitsbereich im aufgerufenen File. Dieses Problem kann in PHP mittels Sessions einfach umgangen werden (siehe PHP-Sessions 4.6.3).

Zu beachten ist auch, dass in jedem File, in welchem eine Klasse benötigt wird, diese instanziert und vor dem verlassen des Files wieder geschlossen wird. Dies wurde so gelöst, da in PHP instanzierter Speicher wieder frei gegeben wird, wenn ein anderes File geladen wird. So ist es nicht möglich, schon in `index.php` eine Klasse DAL für den Zugriff auf die Datenbank zu instanzieren und dann aus allen, später geladenen PHP-Dateien den selben DAL zu benützen, wie dies zum Beispiel in einem C++ Programm möglich ist.

insert

Mit diesen Masken werden Daten in die Datenbank geschrieben. Der Benutzer füllt die Eingabefelder oder wählt verschiedene Punkte aus den Select-Feldern aus. Diese Menüpunkte werden über einfache Queries aus der Datenbank ausgelesen. Bei den Select-Feldern, welche als pull-down Auswahlfelder realisiert sind, ist es jeweils nur ein Wert, bei den anderen sind mehrfach Selektionen möglich. Mehrfachselektionen werden in Arrays an PHP übergeben. Sollte der User ein Feld nicht oder falsch gefüllt haben, welches für die Konsistenz der Daten jedoch erforderlich ist, wird ihm dies mit einer Fehlermeldung mitgeteilt.

insert certificate
Bei der Maske insert certificate werden die Angaben des Benutzers nicht in die Datenbank gespeichert, sondern an die Funktion `makeUserCA()` übergeben. Diese generiert ein Zertifikat und legt dies in der Datenbank ab. Die genaue Beschreibung der Funktion ist unter 7.6 entnehmen.

insert CA
Ähnlich wie bei insert certificate wird bei insert CA eine Funktion, `makeRootCA()`, aufgerufen um die CA zu generieren.

insert meshing
Mit insert meshing können Meshings aus bestehenden Gateways zusammengestellt werden. Zu jedem Gateway muss noch ein Interface angegeben werden und, falls es sich nicht um preshared Verbindungen handelt, ein Zertifikat des gewählten Interfaces. Diese komplexe Abfrage setzt sich aus mehreren Stufen von Datenbankabfragen zusammen. In jeder Auswahl von diesen Stufen wird das PHP-File `insert_meshing_action.php` über die JavaScript Funktion "onClick" aufgerufen, welches die Auswahlmöglichkeiten der nächsten Stufe zusammenstellt. Ein Beispiel: Für jeden Security Gateway, welcher aus der Mehrfachauswahlliste angewählt wird, muss eine Liste mit all seinen Interfaces angezeigt werden und von diesen wiederum eine Auswahl von sämtlichen Zertifikaten. Benötigt werden aber auch Angaben über den Type und den Mode der zu verwendenden IPSec Verbindungen und wie lange die IKE-lifetime sein soll. Bei der Art von Meshing gibt es wieder ein paar spezielle Punkte die beachtet werden müssen: Bei einem Fullmesh werden Knoten, Interfaces und Zertifikate wie oben beschrieben angezeigt und ausgewählt. Falls ein Point-to-Point Meshing generiert werden soll, muss die Anzahl der ausgewählten Knoten überprüft werden, da diese nicht mehr als zwei sein kann. Bei einem Hub-Meshing hingegen muss noch eine Auswahl aller schon ausgewählten Interfaces gemacht werden, da der Benutzer ein Central-Interface auswählen muss. Nachdem alle Angaben korrekt gemacht wurden, wird die Funktion `makeMesh()` aufgerufen, welche ein xml-file für jeden einzelnen Knoten generiert und in die Datenbank schreibt.

update

Bestehende Einträge in der Datenbank können mit den Masken aus dem Bereich update modifiziert werden. Aus Zeitgründen wurden nur die update-Funktionen update node und update meshing implementiert. Die Realisierung weiterer Masken sollte allerdings kein grosses Problem darstellen. Die HTML-Files aller Masken sind bereits vorhanden und die Logik der darunter liegenden Aktionfiles ist aus den beiden bestehenden ersichtlich. Grob gesagt sind die Masken leicht modifizierte Versionen der jeweiligen insert Maske. Die Logik zum Anzeigen der Werte des zu modifizierenden Eintrages stammt von den show Aktionfiles und schlussendlich können noch Bestandteile aus den insert Aktionfiles zum aufdatieren der geänderten Daten übernommen werden. Ein Update läuft wie folgt ab:

1. Die Werte des zu modifizierenden Eintrages werden eingelesen und in HTML-Formelementen dargestellt.
2. Die vom User gemachten Änderungen werden auf ihre Richtigkeit und Konsistenz geprüft.
3. Ein neuer Eintrag in die Datenbank wird vorgenommen. Der alte bleibt vorerst bestehen.
4. Alle m:n-Relationen werden ebenfalls neu geschrieben.
5. Der alte Eintrag und die alten m:n-Verbindungen werden gelöscht.
6. Falls die neuen Daten noch einer Funktion übergeben werden sollen (zum Beispiel die xml-Files neu generieren), wird dies als letzter Schritt des Updateprozesses getan.

delete

Mit delete werden Einträge aus der Datenbank gelöscht, sofern dies aus Sicht der Datenkonsistenz erlaubt ist. Würde eine Löschaktion eine Relation in der Datenbank verletzen, bzw. der User versucht einen Eintrag zu löschen, auf welchen noch ein Foreign-Key zeigt, würde dies dem User mitgeteilt.

Wird ein Zertifikat gelöscht, erfolgt ein Eintrag in die Revokationsliste der entsprechenden CA.

show

Diese Masken sind dazu gedacht, Einträge aus der Datenbank anzuschauen. Es werden keine Änderungen an den Daten vorgenommen.

show meshing
Wie schon bei insert meshing und auch update meshing ist dies die komplizierteste Maske, da die komplizierten Angaben über Knoten, Interfaces und Zertifikate, welche verwendet werden, nicht direkt in der Datenbank gespeichert sind, sondern mit zwei verschachtelten Queries abgefragt und als Tabelle dargestellt werden müssen.

show logging
Mit dieser Maske werden Einträge aus der Entität `config` angezeigt. In ihr sind die xml-Files aller Gateways zusammen mit dem Erstellungsdatum und dem Benutzernamen abgelegt. Änderungen an den Meshings können den xml-Files nachvollzogen werden. Um ein solches File betrachten zu können kann der User xml-Files auf die Harddisk des Web-Servers kopieren.

Navigation

Um die Benutzerfreundlichkeit zu erhöhen ist es möglich, von den show-Masken direkt zu den delete- oder update-Masken zu springen ohne dort den betrachteten Eintrag nochmals anwählen zu müssen. Es ist auch möglich von der delete in die update-Maske zu gelangen. Dies wird erreicht, indem die ID des entsprechenden Eintrags in eine Session-Variable abgelegt wird und in der angesprungenen Maske mit diesem Wert eine Fetchquery durchgeführt wird. Dieses Feature ist einfach zu implementieren, da alle drei Maskenarten ähnlich aufgebaut sind.

7.2 Database Abstraction Layer (DAL)

package vpnadmin/src/dal/

Es wird ein Open Source DAL [14] eingesetzt, welcher unter einer BSD Lizenz erstellt wurde. Dieser beinhaltet den Abstraktions Layer, verschiedene "Treiber" für verschiedene Datenbanken, welche die PHP Funktionen dieser implementiert sowie ein Konfigurationsfile.

7.2.1 Schnittstellen

Methoden:

Name	Beschreibung
Initialize()	Wird vom Konstruktor aufgerufen und initialisiert das System
Destroy()	Destruktor, falls etwas aufzuräumen ist, bevor die Verbindung abgebaut wird.
int selectDriver($drivername)	Auswahl des zu benützenden Treibers
int driverExist($drivername)	Überprüft, ob der gewählte Treiber existiert
array enumDrivers()	Gibt vorhandene Treiber zurück
int executeQuery($query)	**Ausführen einer Abfrage**
Int insertObject($query, $object)	**Einfügen eines Objektes in die Datenbank**
int selectDatabase($database)	Auswahl der zu benützenden Datenbank
object fetchLastInsert()	**Gibt letzten Insert Record zurück**
int fetchInsertID()	**Gibt letzte Insert ID zurück**
object fetchObject()	**Gibt eine Spalte als Objekt zurück**
array fetchArray()	**Gibt eine Spalte als assoziatives Array zurück**
array fetchRow()	**Gibt eine Spalte als numerischs Array zurück**
onError($errorMessage)	Error Handling
errorHandler()	Wrapper für den Error Handler
verifyCapability($capability)	Überprüft, ob der Treiber eine gewisse Funktion unterstützt
setErrorHandler($errorHandler)	Wrapper um eigenen Error Handler einzufügen
startTransaction()	**Startet Transaktion**
commitTransaction()	**Bestätigt Transaktion**
abortTransaction()	**Unterbricht Transaktion**
nextRecord()	**Setzt Cursor auf den nächsten Eintrag**
prevRecord()	**Setzt Cursor auf den vorherigen Eintrag**
seekRecord()	**Setzt Cursor auf einen bestimmten Eintrag, falls möglich**

Die in der Tabelle fett gedruckten Methoden werden zur Interaktion mit der Datenbank benötigt. Falls ein Treiber eine Funktion nicht implementiert hat, wird eine Fehlermeldung zurückgegeben.

7.2.2 Klasse DAL

Diese Klasse implementiert den Abstraktions Layer. Nach der Instanzierung können die verschiedenen Datenbank Funktionen gebraucht werden um auf die Datenbank zuzugreifen.

Felder:

Name	Beschreibung
$driver	Benützter Treiber
$selectedDB	Zu welcher DB verbunden ist
$connected	Steht eine Verbindung
$lastError	Letzter aufgetretener Fehler
$lastQuery	Letzter Abfragestring
$rowcount	Anzahl Spalten
$errorjHandler	Aktueller Errorhandler
$queryType	Typ der Abfrage
$transactionRunning	Wird eine Transaktion ausgeführt

In diesen Variablen ist der aktuelle Zustand der Interaktion zwischen der Applikation und der Datenbank gespeichert.
Die Treiber Klassen implementieren die unter Schnittstellen aufgeführten Funktionen sofern deren Funktionalität von der Datenbank unterstützt wird. Falls nicht, wird diese nicht implementiert und bei einem allfälligen Aufruf durch den Abstracion Layer, wird dem DAL mitgeteilt, dass diese nicht zur Verfügung steht.
Es kann auch vorkommen, dass eine solche Funktion mit mehreren Datenbank spezifischen Funktionen implementiert werden muss, um die gewünschte Funktionalität zu erreichen (siehe 7.2.3 Klasse oci-driver).

Konfigurations File
Die wichtigsten Einträge des Konfigurationsfiles für den DAL wurden in das `vpn_admin` Konfigurationsfile übernommen und müssen dort gesetzt werden:

- $driver: Ein Array mit allen zur Verfügung stehenden Treibern
- $dbuser: Benutzername der Datenbank
- $dbpassword: Passwort
- $dbconnectionString: Die benötigte Zeichenkette, um auf die Datenbank zugreifen zu können

7.2.3 Klasse class_oci_driver

Diese Klasse stellt den Treiber, um auf eine Oracle Datenbank zugreifen zu könen, dar. Es werden folgende DAL-Funktionen unterstützt:

- `connectDB()`
- `object fetchLastInsert()`
- `array fetchRow()`
- `array fetchArray()`
- `object fetchObject()`
- `int fetchInsertID()`
- `int executeQuery($query)`
- `startTransaction()`
- `commitTransaction()`
- `abortTransaction()`
- `nextRecord()`
- `prevRecord()`
- `seekRecord()`

PHP unterstützt zwei verschiedene Oracle Funktions Sets, Oracle und OCI. Letzteres unterstützt die neueren Versionen von Oracle (ab 8i) und weist mehr Funktionalität auf. Da die Ziel-Datenbank Oracle 8i ist, werden die OCI-Funktionen [13] implementiert.

Mit der Funktion `connectDB()`, welche von der DAL-Funktion `selectDatabase()` aufgerufen wird, wird eine Verbindung zur Datenbank aufgebaut, die erst mit dem Schliessen der Klasse DAL wieder unterbrochen wird. Um eine Abfrage, innerhalb einer Schleife von Resultaten einer anderen Abfrage, zu starten, muss ein neue DAL Klasse instanziert werden und mit dieser eine neue Verbindung zur Datenbank hergestellt werden. Somit erhält man einen zweiten Datenbank Cursor, mit welchem man sich unabhängig in der Datenbank bewegen kann.

Oracle verwendet für grosse Datenfelder eigene Datentypen, welche mit PHP und den in dieser PHP-Version (4.06) verfügbaren OCI Funktionen schwierig zu handhaben sind.

Typ	Max. Length	Beschreibung
LONG	bis 2GBytes	Character Daten
RAW	bis 2000 Bytes	Binäre Daten welche von Oracle nicht convertiert werden
LONG RAW	bis 2GBytes	Wie RAW
BLOB	bis 4GBytes	Wie RAW, werden nicht convertiert beim Verschieben zwischen Systemen mit verschiedenen Character Sets
CLOB	bis 4GBytes	Character Daten
BFILE	bis 4GBytes	Stellt zugriff auf ein binär File zur Verfügung, welches sich ausserhalb der Datenbank befindet

Quelle Oracle SQL [6]

Funktion insertObject($query,$object)
Alle grösseren Attribute in einer Tabelle der Datenbank sind vom Typ CLOB (siehe Datenbank .6.2) Um mit diesen Daten arbeiten zu können musste eine Funktion dem DAL hinzugefügt werden, um einen solchen CLOB in die Datenbank zu schreiben. Die Funktion unterstützt INSERT sowie UPDATE SQL-Abfragen mit sogenannten LOBs (Large Object).

```
function insertObject($query,$object){
if(eregi("^update",$query)){
if($handle = OCIParse($this->connection,$query)){
if(OCIBindByName($handle,':clob',&$clob, strlen($clob))){
OCIExecute($handle);
OCICommit($this->connection);
          }
else{
error handling …
else if(eregi("^insert",$query)){
$clob = OCINewDescriptor($this->connection,OCI_D_LOB);
if($handle = OCIParse($this->connection,$query)){
if(OCIBindByName($handle,':clob',&$clob, -1, OCI_B_CLOB)){
OCIExecute($handle,OCI_DEFAULT);
if(strlen($object)){
if($clob->save($object)){
$this->lastInsert($query);
OCIFreeStatement($handle);
OCICommit($this->connection);
return true;
             }
else{
error handling …
       }
     }
   }
```

Es wird eine PHP-Variable an einen Oracle Platzhalter (`':clob'`) gebunden, um den benötigten Speicherplatz für die Ein/Ausgabe zur Laufzeit festzulegen. Die folgende Abfrage zeigt eine solche INSERT Anweisung:

```
INSERT INTO certificate (id, certificate ,key_setid, caid,
interfaceid)
VALUES (certificate_sequence.nextval,EMPTY_CLOB(),keyID, caID,
interface,)
returning certificate into :clob
```

Anstelle in dieser Abfrage den Wert des `certificate` direkt zu übergeben, wird nur ein leeres Element (`EMPTY_CLOB()`) eingefügt und mit der `returning` Anweisung angegeben, welcher Platzhalter dafür verwendet werden soll.

Im INSERT Teil des obigen Codes wird die PHP-Variable `$clob` an diesen Platzhalter (`:clob`) gebunden. Der eigentliche Wert wird danach in den definierten Platzhalter geschrieben.

Der UPDATE Teil funktioniert im wesentlichen gleich, ausser dass für eine UPDATE Abfrage kein LOB (large object) an einen Platzhalter gebunden werden kann. Dies beschränkt die Grösse des übergebenen Strings (auf PHP Verrsion 4.06 Seite) auf 2000 Byte. Auf einigen PHP-Webseiten, welche das aktuelle Manual unterhalten, gibt es neuere OCI-Funktionen mit denen man solche LOBs einfachener handhaben können sollte.
In der in dieser Arbeit verwendeten Version von PHP (Version 4.06) beschränkt sich die Unterstützung auf das Einfügen eines large obects. Dies hatte zur Folge, dass grosse Attribute in der Datenbank, wie zum Beispiel das OpenSSL Konfigurations File (siehe 15.2), welche mit einer UPDATE Abfrage in die Datenbank eingefügt werden sollten, auf der Festplatte gespeichert werden müssen.

7.3 XML – File

package vpnadmin/src/xml/

Um ein XML-File erstellen zu können, muss die Struktur angegeben werden können. Dies wird mittels Ebenen erreicht, indem jedes Element zu einer bestimmter Ebene gehört und einem Tag der höheren Ebene zugeordnet wird. Das erste Element wird Root-Element genannt und muss gesetzt werden. Jedes Element kann Attribute, einen Wert oder weitere Elemente beinhalten.

7.3.1 Schnittstellen

Mit den folgenden Funktionen können Elemente definiert und mit Werten gefüllt werden. Bevor ein XML-File generiert werden kann, muss die Klasse xmlFile instanziert werden.

Funktionen:

Konstruktor

Beschreibung	Initialisiert die xmlFile Klasse und setzt die ID des Knotens, für welchen das XML-File generiert werden soll. Zusätzlich wird die XML Version aus dem vpn_config-File gelesen und die Logger-Klassen instanziert.		
Pre-Kondition	n/a		
Post-Kondition	Eine neue Klasse ist initialisiert		
Parameter	Name	Typ	Beschreibung
	$nodeID	Integer	Node ID des Nodes für welche das XML-File generiert werden soll
Return Wert	n/a		

newElement

Beschreibung	Initialisiert ein neues Element (Tag im XML–File). Instanziert eine Klasse Element und übergibt dieser den Namen, die Ebene auf welcher sich das neue Element befindet und zu welchem Element dieses in der XML-Filestruktur gehört.			
Pre-Kondition	n/a			
Post-Kondition	Ein neues Element wurde dem Array `elements` hinzugefügt.			
Parameter	Name	Typ	Value	Beschreibung
	$id	Integer		Eindeutige ID
	$name	String		
	$level	Integer		Gibt das Level des Tags an
	$super	Integer	Super Element	ID des Element zu welchem dieses gehört
Return Wert		Boolean	True/0	Error handling

setRootElement

Beschreibung	Setzt ein gewünschtes Element als Root Element			
Pre-Kondition	n/a			
Post-Kondition	Ein Element ist das Root Element auf der Ebene 0			
Parameter	Name	Typ	Value	Beschreibung
	$id	Integer		Eindeutige ID
Return Wert		Boolean	True/0	Error Handling

addElementValue

Beschreibung	Fügt ein Wert einem bestimmten Element zu			
Pre-Kondition	Element muss vorhanden sein			
Post-Kondition	Element besitzt einen Wert			
Parameter	Name	Typ	Value	Beschreibung
	$element	Integer		Element ID
	$name	String		
Return Wert		Boolean	True/0	Error Handling

addAttributeValue

Beschreibung	Fügt ein Attribut (Name/Wert –Paar) einem bestimmten Element zu. Ein Element kann beliebige Attribute haben.			
Pre-Kondition	Element muss vorhanden sein			
Post-Kondition	Element hat ein oder mehrere Attribute			
Parameter	Name	Typ	Value	Beschreibung
	$element	Integer		Element ID
	$name	String	Name	
	$value	String	Wert	
Return Wert		Boolean	True/0	Error Handling

setCDATA

Beschreibung	Fügt einem Element das XML-Attribut CDATA hinzu. Der Wert dieses Elementes wird von einem XML-Parser nicht interpretiert und einfach der Applikation zurückgegeben			
Pre-Kondition	Element muss vorhanden sein			
Post-Kondition	Element hat CDATA Tag			
Parameter	Name	Typ	Value	Beschreibung
	$element	Integer		Element ID
Return Wert		Boolean	True/0	Error Handling

7.3.2 Klasse xml_file

Diese Klasse stellt die Funktionen zur Verfügung, um ein XML-File zu generieren. Neben den im Abschnitt 7.3.2 beschriebenen Schnittstellen nach aussen werden zusätzliche Funktionen zur Zusammenstellung der einzelnen Elemente und der Formatierung und Darstellung benötigt.

Felder:

Name	Beschreibung
$version	XML-Version
$elements	Array aller Elemente
$head	Kopf des XML-Files
$rootElement	ID des gesetzten Root-Elemntes
$nodeID	ID des `nodes` zu welchem das generierte XML-File gehört

Für jedes Element wird eine neue Klasse `Elements` (siehe 7.3.3) erstellt, welche dieses repräsentiert und alle benötigten Werte speichert. Diese werden in einem Array `elements` gespeichert, wobei die eindeutige ID des Elements als Index des Arrays dient. So können die einzelnen Elemente im Array unterschieden und gespeichert werden, ohne dass ein Element von einem anderen überschrieben werden kann.

Da das gesamte XML-File verschiedene Hierarchiestufen hat (beim Formatieren durch das Einrücken dargestellt), müssen nur alle Elemente der 1. Ebene explizit aufgerufen werden. Alle diesem Element untergeordneten Elemente werden rekursiv zusammengestellt. Die Funktion toString() wird aufgerufen um alle Elemente zu formatieren und in einen String zu Schreiben. Es werden alle Elemente der 1. Ebene der Funktion nextElement() übergeben, welche die Hierarchie dieses Elementes erstellt.

```
function elementsToString($e){
// set array pointer to the beginning
reset($e);
// get root element
$root=$this->findElement($this->rootElement);
// start tag root element
$this->startElement($root);
do{
// after level 1 elements are written, all the sub-levels
// are written too
$cur=current($e);
if($cur->level == 1)
$this->nextElement($cur);
}while(next($e));
// end tag root element
$this->endElement($root);
return $this->head.$this->comment.$this->data;
}
```

Die Funktion `nextElement()` formatiert die einzelnen Elemente, indem das Start-Tag des Elementes der 1.Ebene geschrieben wird und dann nach Elementen gesucht wird, welche dieses als "Super-Element" eingetragen haben. Wird ein solches gefunden, ruft sich die Funktion selbst wieder auf (Rekursion), um für das "Sub-Element" weitere "Sub-Elemente" zu suchen, usw. Wird keines gefunden wird der Wert dieses Elementes anstelle von "Sub-Elementen" geschrieben. Danach muss noch das "End-Tag" gesetzt werden.

```
function nextElement($current){
// a level 1 element is passed
$e=$this->elements;
$this->startElement($current);
$e=$this->elements;
// search elements that have this element as super-element
while(next($e)){
$cur = current($e);
if($cur->super == $current->id){
// recursion to write all the depending sub-elements
$this->nextElement($cur);
    }
 }
// write the value, if there are no sub-elements
$this->writeValue($current);
// set the end tag
$this->endElement($current);
}
```

Nachdem alle Elemente der 1. Ebene formatiert wurden, sind alle Elemente in den String geschrieben worden, welcher das XML-File repräsentiert.
Die Funktionen `endElement()` und `startElement()` formatieren, respektive rücken die Elemente anhand der angegebenen Hierarchiestufe ein. So erhält man die bekannte Formatierung eines XML-Files (siehe Abbildung 6.4, Struktur eines XML-Files in Kapitel 6.6)

7.3.3 Klasse Element

Diese Klasse wird ausschliesslich von der Klasse `xml_file` benutzt und stellt eigentlich eine Innere-Klasse dar. PHP unterstützt zwar Klassen aber nicht die gesamte Funktionalität einer objektorientierten Programmiersprache (siehe PHP Theorie 4.6), deshalb wurde eine separate Klasse implementiert. Es wird auf einen Schnittstellen Beschreib verzichtet, da es sich eigentlich um die gleichen Funktionen wie in der Klasse xml_file handelt, und nur auf diese von aussen zugegriffen wird.
Diese Klasse repräsentiert ein Element (Tag) und dessen Zustand. Bei der Instanzierung eines Elements müssen die wichtigsten Werte gesetzt werden:

Felder:

Name	Beschreibung
$name	Tag Name im XML-File
$attributes	Array mit allen Name/Wert-Paaren des Elementes
$value	Wert des Elements, kann auch NULL sein (Element hat keinen Wert).
$level	Hierarchiestufe
$number	Anzahl der Attribute
$ super	ID des "Suber-Elements"
$id	Eindeutige ID
$cdata	Flag, ob Element ein CDATA-Attribut besitzt

Die in der obigen Tabelle beschriebenen Felder geben den Zustand, beziehungsweise die Art des Elementes an und wo es sich im XML-File befindet. Beim instanzieren müssen folgende Werte dem Konstruktor übergeben werden:

- $id
- $name
- $level
- $super

Dies setzt voraus, dass das "Super-Element" schon gesetzt und somit bekannt ist. So muss beim Erstellen eines XML-Files darauf geachtet werden, dass die Elemente in der richtigen Reihenfolge gesetzt werden und die Hierarchie bekannt ist.

Mögliche Anwendung

Folgender Ausschnitt zeigt eine mögliche Anwendung dieser Funktionen:

```
/* incrementing the ids */
$id=0;
$xmlFile = new XMLFile($this->nodes["$nodeID"]["id"]);
/* set Head comment */
$xmlFile->setHeadComment("Created date");
/* node: root Element */
$xmlFile->newElement($id,"node",LEVEL_0,0);
$xmlFile->addElementAttribute($id,"name",$this-
>nodes["$nodeID"]["name"]);
$xmlFile->setRootElement($id++);
/* head: level 1 */
$xmlFile->newElement($id++,"head",LEVEL_1,0);
$level_1 = $id - 1;
/* mgmt_IP_1: level 2 */
$xmlFile->newElement($id,"mgmt_IP_1",LEVEL_2,$level_1);
$xmlFile->setElementValue($id++,xxx);
/* mgmt_IP_2: level 2 */
$xmlFile->newElement($id,"mgmt_IP_2",LEVEL_2,$level_1);
$xmlFile->setElementValue($id++, xxx )
/* os: level 3 */
$xmlFile->newElement($id,"os",LEVEL_3,$level_2);
$xmlFile->setElementValue($id++,xxx);

...

for ($j = 0; $j< xxx ; $j++){
$xmlFile->newElement($id++,"protected ",LEVEL_2,$level_1);
$level_2 = $id - 1;
/* ip: level 3 */
$xmlFile->newElement($id,"ip",LEVEL_3,$level_2);
$xmlFile->setElementValue($id++, xxx);
/* subnet: level 3 */
$xmlFile->newElement($id,"netmask",LEVEL_3,$level_2);
$xmlFile->setElementValue($id++, xxx);
  }

...
```

Um die Übersicht der zu vergebenden IDs zu behalten, wird ein Zähler eingeführt, welcher immer bei der letzten Funktion eines Elementes inkrementiert wird. So können Elemente an beliebiger Stelle eingefügt werden, ohne die IDs ständig ändern zu müssen. Auch werden die "Super-IDs" in Variablen gespeichert, damit man diese nicht von Hand ändern muss, wenn ein Element in der Mitte eingeführt wird. Um die Ebene anzugeben, wurden der übersichtshalber Konstanten eingeführt.
Wie im zweiten Abschnitt gezeigt wird, kann ohne weiteres auch eine Schleife gebraucht werden, um allfällige Wiederholungen von bestimmten Elementen zu erreichen.

7.4 Meshing

package vpnadmin/src/meshing/

Hier werden alle Daten zusammengestellt, ausgehend von einem oder mehreren Knoten, welche benötigt werden, um die entsprechenden XML-Files zu generieren. Aufgrund verschiedener Kriterien werden die einzelnen Meshings pro Knoten mit den dazugehörigen Remotes (Verbindungspartner) erstellt.

7.4.1 Schnittstellen

Um ein Meshing erstellen zu können, müssen dem Konstruktor ein Array mit den Knoten, für welche ein XML-File generiert werden soll und in einem zweiten Array die Meshings, in welchen ein bestimmter Knoten vorkommt, übergeben werden.

Methoden

Konstruktor

Beschreibung	Übergibt der Klasse ein Array mit allen Node-IDs, für welche ein XML-File erstellt werden muss, sowie ein Array mit allen Meshing-IDs für die entsprechenden Knoten. Zusätzlich werden Daten aus dem Konfigurationsfile `vpn_admin` herausgelesen, die Loggerklassen und die DAL Klasse instanziert.			
Pre-Kondition	n/a			
Post-Kondition	Eine neue Klasse ist instanziert.			
Parameter	Name	Typ	Value	Beschreibung
	$nodes	Array		Primärschlüssel aller Knoten, für welche ein XML-File generiert werden soll
	$meshings	Array		Pro übergebener knoten, alle Meshings, in welchen dieseer beteiligt ist.
Return Wert	n/a			

MakeMesh

Beschreibung	Ruft alle Methoden auf, um die entsprechenden Werte aus der Datenbank zu lesen und das XML-File zu generieren.			
Pre-Kondition	Klasse muss instanziert und die benötigten IDs übergeben worden sein			
Post-Kondition	XML-Files der gewünschten Knoten wurden erstellt und in der Datenbank gespeichert			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert		Boolean	True/0	Error Handling

GetPresharedKeys

Beschreibung	Erstellt die Keys (Base 64) zur Verwendung der Preshared Konfiguration, und speichert diesen in die Tabelle Preshared_Key für jedes Interface.			
Pre-Kondition	n/a			
Post-Kondition	Ein Preshared Key für eine bestimmte Verbindung wurde generiert und in die Tabelle eingetragen.			
Parameter	Name	Typ	Value	Beschreibung
	$interfaces	Integer		Interface IDs
Return Wert		Boolean	True/0	Error Handling

destroy (Destruktor)

Beschreibung	Schliesst alle geöffneten Ressourcen, sowie die Datenbank-verbindung			
Pre-Kondition	n/a			
Post-Kondition	Alle Ressourcen sind freigegeben.			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert				

7.4.2 Klasse Meshing

Diese Klasse ist verantwortlich, dass das XML-File für einen oder mehrere Knoten korrekt erstellt wird. Es werden der Reihe nach die Daten für den aktuellen Knoten, für die Meshings in welchen dieser vorkommt und die Verbindungen innerhalb der einzelnen Meshings aus der Datenbank gelesen und in die dafür zuständigen Arrays gespeichert. Am Schluss wird das XML-File erstellt.

Felder:

Name	Typ	Beschreibung
nodesIDs	Array	Primary Keys
meshings	Array	Für jeden Knoten (ID als Index) alle Meshings in denen dieser vorkommt
nodes	Array	Alle Daten für die Knoten
selectedMeshing	Array	Alle notwendigen Daten für das Meshing der jeweiligen Knoten
remotes	Array	Alle daten der Verbindungspartner pro Meshing
os	Array	Betriebssystem des aktuellen Knoten
version	Array	Version der IPSec Implementation des aktuellen Knoten
nodeType	Array	IPSec Implementation des aktuellen Knoten
db	Integer	Instanzierte Klasse DAL, Datenbank Verbindung

Die Daten, welche aus der Datenbank gelesen werden, müssen so zwischengespeichert werden, dass auf sie beim Erstellen des XML-Files für jeden Knoten und für alle Meshings dieses Knotens einfach zuzugreifen sind. Die Arrays `nodes`, `selectedMeshing` und `remote` enthalten für die 3 Hauptabschnitte der XML-Struktur(siehe Kapitel 6.6 XML-File) alle Daten, welche in das File geschrieben werden.

`nodes["ID"]["Element"][Element #]["Sub-Element"]`

- "ID": ID des zu verarbeitenden Nodes
- "Element": Tag im zu erstellenden XML-File

Element #: Nummer des aktuellen Elements, falls von einem Tag mehrere Einträge gemacht werden.

- "Sub-Element": Tag innerhalb dieses Tags.

`selectedMeshing[#]["Element"][Element #]["Sub-Element"][Sub-Element#]`

- #: Nummer des aktuellen Meshings für einen bestimmten

Sub-Element #: Nummer des Sub-Tags, falls mehrere Einträge vorhanden sind Knoten

`remotes[Meshing #][Connection #]["Element"][Element #]["Sub-Element"]`

- Meshing #: Nummer des Meshings für einen bestimmten Knoten
- Connection #: Innerhalb dieses Meshings, die Nummer der aktuellen Verbindung

Mit der oben gezeigten Struktur der Arrays kann auf bestimmte Elemente der verschiedenen Abschnitte im XML-File einfach zugegriffen werden. Die Veschachtelung von Arrays bleibt dank den verschiedenen Möglichkeiten der Indexierung (assoziativ, String, oder numerisch) übersichtlich.

```
function makeMesh(){
for($i=0; $i<count($this->nodeIDs); $i++){
$this->selectNode($this->nodeIDs[$i]);
$this->selectNodeType($this->nodeIDs[$i]);
$this->selectMeshing($this->nodeIDs[$i]);
$this->selectRemote($this->nodeIDs[$i]);
$this->makeXML($this->nodeIDs[$i]);
      }
}
```

Die aufgerufenen Funktionen in makeMesh() greifen mittels SQL auf die Datenbank zu, um die gewünschten Daten zu erhalten. Die Funktion selectRemote() bestimmt zusätzlich, ob es sich um ein HUB-Meshing handelt. Falls dies der Fall ist, muss noch geprüft werden, ob der aktuelle Node der Central-Node dieses Meshing ist. Wenn nicht, gibt es nur eine Verbindung, die zum Central-Node. Sonst kann gleich wie bei einem FULL-Mesh weitergefahren werden:

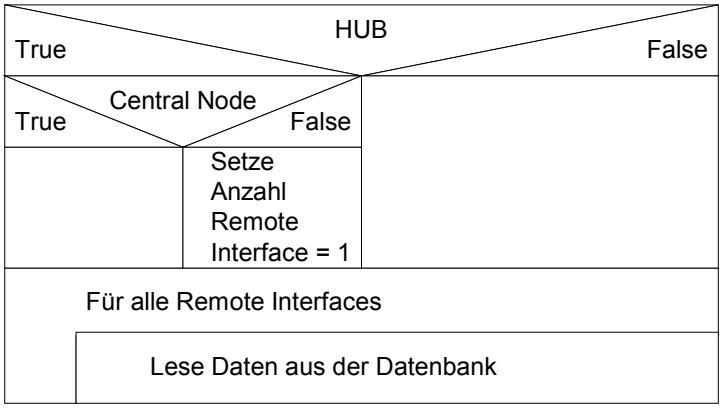


Abbildung XXX: Struktogramm HUB

Nachdem alle Daten zusammengetragen wurden, wird das XML-File generiert. Die Funktion makeXML() legt die Struktur des XML-Files fest, indem die Klasse xml_file instanziert wird und deren Funktionen dementsprechend aufgerufen werden (siehe 7.3.2

7.5 Linux Filter

package vpnadmin/src/filter/

Im Filter werden die FreeS/WAN Konfigurations-Files ipsec.conf und ipsec.secrets generiert, indem das XML-File für den ausgewählten Knoten geparsed und je nach gewählten Authentisierungs-Verfahren zusammengestellt werden.

7.5.1 Schnittstellen

Um das XML-File zu parsen und die Konfigurations-Files zu erstellen, muss lediglich die Klasse `linux_filter` instanziert und die Funktion `createFiles()` aufgerufen werden, welcher die ID des Knotens, für den die Files erstellt werden sollen, übergeben wird.

Methoden:

Konstruktor

Beschreibung	Initialisiert die Klasse linux_filter und instanziert die Logger-klassen.			
Pre-Kondition	n/a			
Post-Kondition	Eine neue Klasse ist instanziert			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert	n/a			

createFiles

Beschreibung	Parsed das XML-File und ruft die Methoden auf, um die Konfigurations-Files zu erstellen.			
Pre-Kondition	n/a			
Post-Kondition	Konfigurations-Files sind bereit um an den Knoten zu senden.			
Parameter	Name	Typ	Value	Beschreibung
	$id	Integer		ID des Knotens, für welchen die Files generiert werden sollen
Return Wert		Boolean	True/0	Error Handling

destroy (Destruktor)

Beschreibung	Schliesst alle geöffneten Ressourcen, sowie die Datenbank-verbindung			
Pre-Kondition	n/a			
Post-Kondition	Alle Ressourcen sind freigegeben.			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert				

7.5.2 Klasse linux_filter

Die zum Parsen benötigten Funktionen werden von PHP [13] zur Verfügung gestellt. Es wird der eigentliche Parser, die Handler-Funktionen der Start- und End-Tags, sowie dem Handler für die Daten benötigt:

- `xml_parser_create()`
- `xml_set_element_handler(parser,`*`startHandler,endHandler`*`)`
- `xml_set_character_data_handler(parser,`*`dataHandler`*`)`
- `xml_parse(parser,data,length)`

Die Funktionen startHandler, endHandler und dataHandler, welche den PHP-Funktionen übergeben werden, müssen implementiert werden um mit den Daten zu Arbeiten. `xml_parser_create()` erzeugt eine Referenz auf den Parser, welche an alle xml_xxx-Funktionen übergeben werden muss. Die eigentliche Parse-Funktion, welcher das XML-File übergeben werden muss, ist `xml_parse()`. Diese muss aufgerufen werden, um den Parsevorgang zu starten.

Um diese Klasse zu implementieren musste etwas gemogelt werden: Den beiden Funktionen `xml_set_element_handler()` und `xml_set_character_data_handler()` müssen die Namen ("startHandler", "endHandler", beziehungsweise "dataHandler") als Strings übergeben werden. Um diese nun in einer Klasse implementieren zu können, indem zum Beispiel eine `init_xml()` Funktion die PHP Funktionen initialisieren würde, hätten aber Referenzen übergeben werden müssen.

Aus diesem Grund wurde der gesamte XML-Parse Teil in einer eigenen Funktion implementiert. Die zweite Schwierigkeit lag nun darin, dass die Funktionen innerhalb der `parseFile()`-Funktion nicht auf Klassenvariablen zugreifen können! Dies wurde auf eine unschöne Art, mit globalen Variablen, gelöst. PHP hat den Vorteil, dass man nur auf globale Variablen zugreifen kann, wenn diese mit dem Schlüsselwort `global` sichtbar gemacht werden. Somit kann sichergestellt werden, dass innerhalb des gesamten `linux.phps` File nicht zufällig eine solche überschrieben werden kann.

```
function parseFile(){
global $protected, $c, $n, $tag, $attr, $ip, $remote_ip,
$name, $pNetwork, $certID;
static $line = 0;
$parser = xml_parser_create("ISO-8859-1");
  …
  }

function startHandler($parser, $name, $attribs){
global $protected,$c,$n,$tag,$attr,$linux_ipsecPath;
  …
  }

function dataHandler($parser, $data){
global $protected, $n, $tag, $certFile, $type, $mode,
$ike, $central, $conn, $m,$line;
static $pn = 0;
static $i = 0;
  …
  }

function endHandler(){
global $n,$certFile,$m;
  …
  }

/* set handler functions */
xml_set_element_handler($parser, "startHandler","endHandler");
xml_set_character_data_handler($parser, "dataHandler");
/* parse the file */
if(!xml_parse($parser,$this->data){
  …
  }

$this->tag = $tag;
$this->attr = $attr;
/* freeing the parser */
xml_parser_free($parser);
}
```

Am Ende der Funktion parseXML() wurden die globalen Variablen den Klassenvariablen übergeben, um damit innerhalb der Klasse sauber (ohne globale Variablen) arbeiten zu können.

Um die Werte der Elemente in die vorgesehene Datenstruktur, den unten beschriebenen Arrays zu schreiben, muss der `dataHandler()` Funktion der Namen des Elements via globale Variable übergeben werden:

```
function parseFile(){
…
if(ereg("SUBJECT_ALT_NAME",$name)
$n = $name;
…
}

function dataHandler($parser, $data){
global $n, … ;
stataic $i=0;
…
if(ereg("SUBJECT_ALT_NAME",$n)){
global $tag,$m;
$i = 0;
$tag["MESHING"]["$m"]["INTERFACE"]["SUBJECT"] = $data;
      }
…
}
```

Codeausschnitt 7.X: Übergabe der geparsden Elementnamen

Felder:

Name	Beschreibung
$psecPath	Pfad, wo die Files temporär gespeichert werden.
$setup	Setup String des ipsec.conf
$default	Default String des ipsec.conf
$connection	Alle Connection-Abschnitte im ipsec.conf
$head	Kommentar am Anfang des ipsec.conf
$secret	Gesamtes ipsec.secrets
$tag	Array mit den geparsden Daten, geordnet nach Bedarf
$attr	Array mit allen Name/Wert Paare, wobei der Index der Name des Elements ist.

Ausser den Certifikaten und dazugehörign CAs, welche in Files geschrieben werden, werden alle Daten in das Array `$tag` geschrieben. Den Zertifikaten wird der Name des Meshings, den CAs deren Name, gegeben, damit diese unterschieden werden können. Das Array `$tag`, ist in drei Teile unterteilt worden:

1) `$tag["MESHING"]["Meshing-Name"]["Element"]`
2) `$tag["Meshing-Name"]["Connection-Name"]["Element"][Number]`
3) `$tag["PROTECTED_NETWORK"][Number]`

Im 1. Teil werden für jedes Meshing die organisatorischen Daten gespeichert, welche benötigt werden, um die Konfigurations-Files korrekt zusammenzustellen. Der 2.und 3. Teil beinhaltet alle für den Abschnitt `conn` im `ipsec.conf`.File wichtigen Daten, wobei `$tag["PROTECTED_NETWORK"]` die von diesem Knoten beinhaltet. Für das `ipsec.secrets` File sind die Private Keys, falls mit Zertifikaten authentisiert wird, oder die Preshared Keys, im Falle von Preshared-Secrets Authentisierung gespeichert. Beim ipsec.conf muss nur unterschieden werden, welche Authentisierung für die einzelnen Meshings gewählt wurden, ob es sich beim IPSec Mode um "tunnel" oder "transport" handelt. Bei "tunnel" muss für jedes "Protected-Network", ob beim Host- oder beim Remote Security Gateway, eine Verbindung eingetragen werden, um diese erreichen zu können (4.2.3).

Im ipsec.secrets werden alle Private-Keys, beziehungsweise alle Preshared-Secrets eingetragen. Um ein Private-Key einem Zertifikat und somit einem Meshing zuordnen zu können, muss der "Subject-Alt-Name" des Zertifikates vor den Schlüssel eingetragen werden:

```
subject-alt-name:RSA    {
          Modulus:         0xCDE34…
          PublicExponent: 0x010001
          PrivateExponent 0xDE89A…
          Prime1:          0cBC237…
          Prime2:          0xAFC3B…
          Exponent1        0xBDC34…
          Exponent2        0x98BD1…
          Coefficient:     0xEF65A…
}
```

Bei den Preshared-Keys müssen die Host-IP und die Remote-IP angegeben werden, um die Verbindung eindeutig zu identifizieren:

```
Host-IP Remote-IP: Preshared-Secret
```

Alle Zertifikate (Zertifikat, CA, CRL) sowie der Private-Key im XML-File sind im PEM-Format (Base64) gespeichert. Somit müssen allfällige Formatänderungen im jeweiligen Filter vorgenommen werden. Hier werden die CAs und die CRLs ins DER-Format und der Private-Key in das FreeS/WAN kompatible Format für das ipsec.secrets gewandelt werden [9].

Damit für verschiedene Knoten die Konfigurations-Files erzeugt und temporär auf der Festplatte gespeichert werden können, wird für jeden Knoten ein eigenes Verzeichnis erstellt:

```
vpnadmin/tmp/config/<node-name>
```

7.6 Zertifikat Erzeugung

package vpnadmin/src/certificates/

Damit eine eigene Zertifizierungsstelle unterhalten werden kann, muss ausser einer eigenen CA auch die Möglichkeit vorhanden sein, Certificate Requests zu erstellen und diese nach Bedarf mit der CA zu signieren. Das Package Certificates stellt alle Funktionen zur Verfügung um mittels OpenSSL eine oder mehrere CAs zu unterhalten.

7.6.1 Schnittstellen

Um die einzelnen Funktionen aufzurufen, muss zuerst die entsprechende Klasse, certificate oder crl, instanziert werden. Danach können CAs, Zertifikate und CRLs erstellt werden. Ausser beim Erstellen einer neuen CA, muss eine CA zuerst geladen werden, um für die entsprechende Zertifizierungskette ein Zertifikat zu erstellen oder zu sperren, beziehungsweise eine CRL dieser CA zu erzeugen. Nachdem alle Arbeiten ausgeführt wurden, muss die geladene CA wieder geschlossen werden, das heisst, sie wird von der Festplatte gelöscht und das `index.txt` File wird in das entsprechende Verzeichnis kopiert.

Wie im Abschnitt 7.2, DAL, beschrieben wurde, kann dieses File, sowie das `openssl.cnf`, in dieser Version nicht in der Datenbank abgespeichert werden. Deshalb wird beim Erstellen einer neuen CA ein Verzeichnis erstellt, in welchem die korrespondierenden Files permanent gespeichert werden (siehe 9.1 Konfiguration.).

Die internen Funktionen, um diese Files in der Datenbank zu speichern sind implementiert und können, sobald PHP die dafür benötigten Oracle 8 OCI-Funktionen zur Verfügung stellt, gebraucht werden. Im folgenden Abschnitt 7.6.2 Klasse `certificate` wird näher darauf eingegangen.

Methoden:

Konstruktor certificate	
Beschreibung	Es werden verschiedene Werte aus dem Konfigurationsfile vpn_config gelesen und initialisiert, sowie eine Verbindung zur Datenbank hergestellt. Zusätzlich wird ein File mit Zufallswerten aus /dev/urandom gefüllt, auf welches zugegriffen wird, wenn ein Private-Key erstellt wird (siehe OpenSSL 4.4.)
Pre-Kondition	n/a
Post-Kondition	Verbindung zur Datenbank ist hergestellt, Variablen sind initialisiert.
Parameter	Name Typ Value Beschreibung
	n/a
Return Wert	n/a

Konstruktor crl	
Beschreibung	Es werden verschiedene Werte aus dem Konfigurationsfile vpn_config gelesen und initialisiert, sowie eine Verbindung zur Datenbank hergestellt.
Pre-Kondition	n/a
Post-Kondition	Verbindung zur Datenbank ist hergestellt, Variablen sind initialisiert.
Parameter	Name Typ Value Beschreibung
	n/a
Return Wert	n/a

makeRootCA

Beschreibung	Mit dieser Funktion wird eine neue CA generiert und in der Datenbank gespeichert, sowie die benötigten Verzeichnisse erstellt in welche das `openssl.cnf` und das `index.txt` kopiert werden.			
Pre-Kondition	n/a			
Post-Kondition	Neue CA kann zur weiteren Arbeit benützt werden.			
Parameter	Name	Typ	Value	Beschreibung
	$keyType	Integer	Fremd-schlüssel	ID der zu verwendenden Hash-Funktion
	$dur	Integer		Die Gültigkeitsdauer der CA
	$length	Integer	1024 2048 4096 ...	Die Schlüssellänge des Private Keys
	$country	String	XX	Abkürzung des Landes mit zwei Buchstaben
	$location	String		Angaben zur CA. Können weggelassen werden.
	$organisation	String		
	$unit	String		
	$name	String		
	$email	String		
	$caName	String		Der Name der CA
	$policy	String		Info
Return Wert		Boolean	True/0	Error Handling

makeUserCA

Beschreibung	Mit dieser Funktion wird ein neues Zertifikat erstellt. Folgende Schritte werden ausgeführt: • Erstellen eines Certificate Request • Einfügen des Requests und des Private keys in die Datenbank • Signieren des Certificate Requests • Einfügen des neuen Zertifikates in die Datenbank			
Pre-Kondition	Mit der zu signierenden CA muss geladen sein			
Post-Kondition	n/a			
Parameter	Name	Typ	Value	Beschreibung
	$caID	Integer	Fremd-schlüssel	ID der CA, mit welcher signiert werden soll
	$interface	Integer	Fremd-schlüssel	Interface ID, definiert, zu welchem Interface das Zertifikat gehört.
	$keyType	Integer	Fremd-schlüssel	ID der zu verwendenden Hash-Funktion
	$keyType	Integer	Fremd-schlüssel	ID der zu verwendenden Hash-Funktion
	$dur	Integer		Die Gültigkeitsdauer der CA
	$length	Integer	1024 2048 4096 ...	Die Schlüssellänge des Private Keys
	$country	String	XX	Abkürzung des Landes mit zwei Buchstaben
	$location $organisation $unit $name $email	String String String String String		Angaben zur CA. Können weggelassen werden.
	$caName	String		Der Name der CA
	$policy	String		Info
Return Wert		Boolean	True/0	Error Handling

getRootCA

Beschreibung	Laden einer CA, um Zertifikate zu signieren oder zu sperren, sowie die CRL erstellen zu können.			
Pre-Kondition	n/a			
Post-Kondition	`openssl.cnf` und `index.txt` werden ins Arbeitsverzeichnis dieser Applikation geladen.			
Parameter	Name	Typ	Value	Beschreibung
	$id	Integer	Prim$rschlüssel	ID der zu ladenden CA
Return Wert		Boolean	True/0	Error Handling

closeRootCA

Beschreibung	Löscht die temporär gespeicherte CA sowie das `openssl.cnf` File und verschiebt das `index.txt` zurück in das "Home"-Verzeichnis dieser CA.			
Pre-Kondition	n/a			
Post-Kondition	`Index.txt` ist gesichert.			
Parameter	Name	Typ	Value	Beschreibung
	$id	Integer	Primärschlüssel	ID der geladenden CA
Return Wert		Boolean	True/0	Error Handling

RevokeCertificate

Beschreibung	Sperrt ein Zertifikat einer bestimmten CA			
Pre-Kondition	n/a			
Post-Kondition	Zertifikat wurde im `index.txt` File der CA als gesperrt eingetragen.			
Parameter	Name	Typ	Value	Beschreibung
	$id	Integer	Primärschlüssel	ID der geladenden CA
	$certID	Integer	Primärschlüssel	ID des zu sperrenden Zertifikates
Return Wert		Boolean	True/0	Error Handling

makeOneCRL

Beschreibung	Erstellt ein CRL Zertifikat einer bestimmten CA			
Pre-Kondition	n/a			
Post-Kondition	CRL Zertifikat steht im angegebenen Verzeichnis.			
Parameter	Name	Typ	Value	Beschreibung
	$caid	Integer	Primärschlüssel	ID der geladenden CA
	$path	String		Verzeichnis, wo das CRL Zertifikat gespeichert werden soll
	$name	String		Name des zu generierenden Files
Return Wert		Boolean	true/0	Error Handling

destroy (Destruktor)

Beschreibung	Schliesst alle geöffneten Ressourcen, sowie die Datenbank-verbindung. Zusätzlich werden alle temporär auf der Festplatte zwischengespeicherten Files gelöscht			
Pre-Kondition	n/a			
Post-Kondition	Alle Ressourcen sind freigegeben.			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert				

7.6.2 Klasse certificate

Hier werden alle Funktionen zur Verfügung gestellt, um mit den verschiedenen Zertifikate zu Arbeiten. Es können zwei Gruppen unterschieden werden:

- Erstellen und manipulieren von Zertifikaten
- Verwalten der Zertifikate

Die Funktionen zur Erstellung sind zugleich die Schnittstellen nach aussen (siehe oben) und implementieren die verschiedenen OpenSSL Kommandos (siehe 4.4) OpenSSL). Um auf diese zugreifen zu können, wird jeweils eine Pipe zu `openssl` geöffnet um den gewünschten Befehl auszuführen. Normalerweise interagiert der Benutzer mit `openssl`, um die gewünschten Parameter oder eventuelle Passwörter anzugeben. Diese werden der Pipe übergeben, in der Reihenfolge, in welcher sie abgefragt werden.

```
/* open a pipe to create an selfsigned X509 Certificate */
$pipe = popen (
"openssl req -config $this->utilsPath/openssl.cnf ".
"-new -x509 -days $dur -newkey rsa:$length ".
"-keyout $this->SSLDIR/private/CAkey.pem ".
"-out $this->SSLDIR/RootCA.pem ","w"))
/* write all the needed parameters into the pipe */
fputs($pipe, $this->passphrase."\n");
fputs($pipe, $this->passphrase."\n");
/* if an value is empty, the defaults specified in the
openssl.cnf will be used send a line feed to get to the next
parameter */
if (ereg("^[ \n\t\f\r]",$country))
fputs($pipe, "\n");
else fputs($pipe, $country."\n");

     …

pclose($pipe);
```

Da es möglich ist, die default Werte aus dem openssl.cnf zu benützen, indem keine Eingabe gebacht wird, beziehungsweise einfach "Return" gedrückt wird, muss bei solchen Parameter abgefragt werden, ob ein Wert übergeben wurde oder nicht. Alle Eingaben müssen mit einem "\n" (new line) abgeschlossen werden, um diese zu bestätigen.

Bei den User Zertifikaten kann zwischen zwei `Subject-Alt-Names` unterschieden werden, welche dem Konfigurationsfile `openssl.cnf` mittels Umgebungsvariable übergeben wird. Es werden Email-Adresse sowie eine IPv4 Adresse unterstützt.

```
   /* if the common name is an IP address,
   the subject_alt_name will be set so*/
   if ($this->validate_IP($name)){
   /* set the environment variable used by the openssl.cnf file */
   putenv("SUBJECT_ALT_NAME=IP:$name");
   /* flag if email or ip is used */
   $ip = true;
   }
   /* otherwise use the email address as subject_alt_name */
   else{
   putenv("SUBJECT_ALT_NAME=email:copy");
   $ip = 0;
}
```

Im Konfigurationsfile wurde folgende Änderung vorgenommen, um auf die Umgebungsvariable (`$ENV`)zugreifen zu können:

- `subjectAltName = email:copy`
- `subjectAltName = $ENV::SUBJCT_ALT_NAME`

Die Passwörter, welche beim Erzeugen von Private-Keys erforderlich sind, werden zufällig, und mit einer im Konfigurationsfile `vpn_config` angegebenen Länge, erzeugt. Es wird ein Perl-Skript aufgerufen, welches aus `/dev/urandom` einen zufälligen Bytestrom ausliest, diesen in Base 64 wandelt und zurück gibt.

```
#! /usr/bin/perl
open (RAND, "/dev/urandom") or die
"Couldn't open /dev/urandom $!" ;
while (length $base < @ARGV[0]){
read RAND,$bytes,64;
$ascii = unpack "a*", $bytes;
use MIME::Base64;
$base .= encode_base64($ascii);
# no new line
$base =~ s/\n//;
}
close RAND;
print $base;
```

Das "caracter device file" `/dev/random` liest "Umgebungs Rauschen" (environmental noise) von verschiedenen Hardware-Treibern und anderen Quellen in einen Entropie-Pool. Es können nur so viele Bits gelesen werden wie vorhanden sind. Ein Prozess wird blockiert, wenn er aus einem leeren Pool lesen will. Aus diesem File sollte gelesen werden, wenn eine möglichst hohe Zufälligkeit erforderlich ist.

Eine weniger hohe Zufälligkeit, dafür werden so viele Bits zur Verfügung gestellt, wie angefordert werden, bietet `/dev/urandom`. Hier wird der Entropie-Pool, wenn keine Daten von der Umgebung mehr gesammelt werden können, werden die vorherigen wieder verwendet. Somit sind die zurückgegebenen Daten verwundbar gegenüber kryptographischen Attacken.

Da es für diese Anwendung wichtiger ist, nicht zu blockieren als eine absolut hohe Zufälligkeit zu erreichen, wird für das generieren der Passwörter aus `/dev/urandom` gelesen.

Die Funktionen zum Verwalten der Zertifikate lesen und schreiben diese und die dazugehörigen Daten in und von der Datenbank. Wie im Abschnitt 7.5.1 Schnittstellen bereits erwähnt wurde, können die beiden Files, `openssl.cnf` und `index.txt`, nicht in der Datenbank gespeichert werden (siehe auch 7.2 DAL). Die dazugehörigen Funktonen wurden implementiert:

- `putIndexFile($caID)`
- `getIndexFile($caID)`
- `putConfig($caID)`
- `getConfig($caID)`

Falls in einer späteren Version dieser Applikation eine Funktion in der Klasse `dal` implementiert wird, mit welcher Objekte mit einer UPDATE-Abfrage in die Datenbank geschrieben werden können, muss nur der Abfrage-String selbst und die aufzurufende Datenbankfunktion geändert werden. Auch die Aufrufe in diesen Funktionen in

- `putRootCA()`
- `openRootCA()`
- `closeRootCA()`

sind vorhanden und müssen nur einkommentiert werden und diejenigen Zeilen löschen, welche die beiden Files kopieren. Das Erstellen der Verzeichnisse, in welchen diese beiden Files permanent gespeichert werden, entfällt ebenfalls.

7.6.3 Klasse crl

In dieser Klasse werden die beiden OpenSSL-Befehle zum Sperren eines Zertifikates und zum Erstellen einer CRL für eine bestimmte CA implementiert. Der Aufruf der beiden in Abschnitt Schnittstellen (7.6.1) erwähnten Funktionen führt automatisch das Laden und wieder Schliessen der benötigten CA aus.

- `makeOneCRL($caid,$path,$name)`
- `revokeCertificate($caID,$certID)`

7.7 Logging und Error Handling

package vpnadmin/src/error/

Es werden die wichtigsten Tätigkeiten in ein Log-File geschrieben, um nachvollziehen zu können, welche Änderungen, von wem durchgeführt wurden. Falls ein Fehler auftritt, wird dieser separat ins Error-Log-File geschrieben.

7.7.1 Schnittstellen

Will man in ein Log-File schreiben, muss die entsprechende Klasse instanziert werden und dieser den Namen des Moduls, in welchem geloggt wird, übergeben. Danach kann an einer beliebigen Stelle ein Eintrag in das Log-File geschrieben werden.

Methoden:

Konstruktor

Beschreibung	Es werden der Pfad und der Name des Log-Files oder des Error-Files aus dem Konfigurationsfile `vpn-config` gelesen und das entsprechende File zum anhängen von Daten geöffent.			
Pre-Kondition	n/a			
Post-Kondition	Log-File wurde zum anhängn von Daten geöffnet.			
Parameter	Name	Typ	Value	Beschreibung
	$class	String		Name des Modules, in welchem geloggt wird.
Return Wert	n/a			

writeToLog

Beschreibung	Mit dieser Funktion schreibt man eine Zeile in das Log-File. Der zu schreibende Eintrag wird beim Aufruf übergeben.			
Pre-Kondition	n/a			
Post-Kondition	Eine Zeile wurde dem Log-File angehängt			
Parameter	Name	Typ	Value	Beschreibung
	$msg	String		Zu schreibender Eintrag
Return Wert	n/a			

writeToError

Beschreibung	Eine etwas abgeänderte Version der writeToLog-Funktion: Es können entweder vordefinierte Error Meldungen, zu diesen noch eine Ergänzung gemacht, oder eine eigene Meldung in das Error-File geschrieben werden.			
Pre-Kondition	n/a			
Post-Kondition	Eine Zeile wurde dem Log-File angehängt			
Parameter	Name	Typ	Value	Beschreibung
	$no	Integer		Whal der vordefinierten Error Meldung
	$msg	String		Zusätzlicher Eintrag
Return Wert	n/a			

destroy (Destruktor)

Beschreibung	Schliesst das geöffnete File und gibt den File-Handle wieder frei.			
Pre-Kondition	n/a			
Post-Kondition	Log-File wurde geschlossen.			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert				

7.7.2 Klasse Logger

Hier wird der Logging-String zusammengestellt und zusammen mit der übergebenen Meldung in das Log-File geschrieben. Es wird der Benutzer aus der Apache-Environment Variable `"REMOTE_USER"` gelesen und die aktuelle Zeit berechnet.
Logging-Eintrag:

```
2001.10.29:12:00:00;Modul-Name;User-Name;Logging-String
```

7.7.3 Klasse Error

Diese Klasse wurde von der Klass Logger abgeleitet. Zusätzlich wurden noch Funktionen implementiert, welche der Error Nummer den dazugehörigen Error-String zuordnet und in den Logging-String schreibt. Das Schreiben in das Error-File übernimmt die Funktion `writeToLog()` von der Klasse `Logger`, welche innerhalb der von `writeToError()` aufgerufen wird.

Vordefinierte Error-Meldungen:

Error #	Message
0	No Error Message available
1	Couldn't open file
2	Couldn't write to file
3	Couldn't read from file
4	Database Error
5	Couldn't read from pipe
6	Couldn't write to pipe

Der Funktion `writeToError()` kann neben der Error-Nummer auch noch ein zusätzlicher String übergeben werden, welcher an die vordefinierten Meldungen angehängt werden. Zum Beispiel wäre es interessant zu wissen, welche Datenbank Abfrage nicht ausgeführt werden konnte. So kann man neben der Error Nummer 4 noch den String der Abfrage übergeben.

Wenn eine Benutzerdefinierte Error Meldung geschrieben werden soll, kann die Error Nummer 0 übergeben werden, und zusätzlich die gewünschte Nachricht. Wird mit der Error Nummer 0 keine Message übergeben, wird der Default-Eintrag ins Error-File geschrienben.

7.8 Senden der Konfigurations Files

package vpnadmin/src/sending/

Nachdem Erstellen eines neuen Meshings müssen wie schon erwähnt, die verschiedenen Filter angewendet werden und die erstellten Konfigurations-Files an die entsprechenden Security Gateways gesendet werden. Dies wird auf der Basis von ssh [15] mittels einer sicheren Verbindung durchgeführt.

7.8.1 Schnittstellen

Es muss zuerst die Klasse sending instanziert werden, bevor die funktion `send()` aufgerufen werden kann, welche die für jeden angegebenen Knoten den benötigten Filter aufruft und danach die erstellten Files in die entsprechenden Verzeichnisse auf dem Security-Gateway kopiert.

Methoden:

Konstruktor

Beschreibung	Es werden verschiedene Informationen aus dem Konfigurations-File `vpn_config` gelesen.			
Pre-Kondition	n/a			
Post-Kondition	Variablen wurden initialisiert			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert	n/a			

Send

Beschreibung	Diese Funktion wird aufgerufen, wenn für bestimmte Knoten die Konfigurationsfiles (sicher) kopiert werden sollen. Es wird ein Array von Node-IDs erwartet, um für diese die Filter aufzurufen und danach die erstellten Files zu kopieren.			
Pre-Kondition	n/a			
Post-Kondition	Für alle angegebenen Knoten wurden die Konfigurations-Files erstellt und sicher mit scp (secure copy) zum Security gateway kopiert.			
Parameter	Name	Typ	Value	Beschreibung
	$nodes	Integer	Primärschlüssel	Node-Ids
Return Wert		Boolean	true/0	Error Handling

destroy (Destruktor)

Beschreibung	Schliesst alle geöffneten Ressourcen, sowie die Datenbank-verbindung. Zusätzlich werden alle erstellten Verzeichnisse in `vpnadmin/tmp/config` gelöscht.			
Pre-Kondition	n/a			
Post-Kondition	Alle Ressourcen sind freigegeben.			
Parameter	Name	Typ	Value	Beschreibung
	n/a			
Return Wert				

7.8.2 Klasse sending

Diese Klasse implementiert das Senden der Konfigurations-Files. Es werden für alle Knoten die richtigen Filter aufgerufen und danach werden diese in die entsprechenden Verzeichnisse kopiert.

Für das Kopieren und verifizieren, ob die Files auch kopiert wurden, wird ein Skript aufgerufen, welches für eine bestimmte IPSec-Implementation die veschiedenen Files an die richtigen Orte kopiert.

FreeS/WAN Verzeichnisse:

Quelle vpnamin/tmp/config/<node-name>	**Ziel** /etc/
ipsec.conf	ipsec.conf
ipsec.secrets	ipsec.secrets
<zertifikat>.pem	ipsec.d/<zertifikat>.pem
certs/<ca>.der	ipsec.d/certs/<ca>.der
crls/<crl>.der	ipsec.d/crls/<crl>.der

Im Shell-Skript werden zuerst die verschiedenen Files mit `scp` kopiert. Dies benötigt keine besondere Aufmerksamkeit, da die Firma Open Systems AG alle sichern ssh Verbindungen zu den von ihr verwalteten Securrity Gateways permanent überwacht. Es bestehen zudem zu allen Management-IP-Adressen eine "Trusted-Host" Verbindung (siehe 4.5 ssh), das heisst, es müssen keine Passwörter eingegeben werden um eine SSH-Verbindung zu erstellen. Dem Skript wird jedoch angegeben, welche Identität für die Verbindung benützt werden soll.

```
#! /bin/sh
# Send Konfiguration Files and Certificates
# to the Security-Gateway

# -i identity for ssh
IDENT=$1
# ip address where to send the files to
IP=$2
# the source directory
DIR=$3

cd $DIR
scp $IDENT ipsec.conf ipsec.secrets root@$IP:/etc
scp $IDENT *.pem root@$IP:/etc/ipsec.d
scp $IDENT cacerts/* root@$IP:/etc/ipsec.d/cacerts
scp $IDENT crls/* root@$IP:/etc/ipsec.d/crls
```

Was jedoch geprüft wird, ist, ob die Files kopiert wurden. Für alle kopierten Files im Quellverzeichnis (siehe Tabelle) wird ein MD5 Hash berechnet. Das gleiche wird mittels SSH im Zielverzeichnis des Security Gateways durchgeführt.

```
# remove old files
rm /tmp/md5sum.remote
rm /tmp/md5sum.host

# get the md5 checksum of all the remote files just sent
for i in `ssh $IDENT $IP find /etc/ipsec.* -type f`; do ssh $IDENT
$IP md5sum $i | awk '{print $1}' >> /tmp/md5sum.remote ; done;
# get the md5 checksum of the source files
for i in `find $DIR -type f`; do  md5sum $i | awk '{print $1}' >>
/tmp/md5sum.host ; done;

# sort them
sort -o /tmp/sort.remote /tmp/md5sum.remote
sort -o /tmp/sort.host /tmp/md5sum.host

# check whether they are equal or not
# done in send.phps
```

Alle Hash-Werte werden in entsprechende Files (`.host`, beziehungsweise `.remote`) geschrieben. Da die Reihenfolge dieser Einträge für einen Vergleich in beiden Files die selbe sein muss, werden diese noch sortiert. Nun könne diese beiden Files in einen String gelesen und verglichen werden. Sind diese beiden Strings identisch, wurden alle Files korrekt kopiert.
Beim Schliessen der instanzierten Klasse sending, werden alle Verzeichnisse und somit deren Einträge in `vpnadmin/tmp/config` gelöscht.

8 Funktions-Test

Das Testen der einzelnen Module (Datenbankzugriff, GUI, Meshing, XML-File generieren, Linux-Filter) wurde während der gesamten Implementation fortlaufend durchgeführt. Man kann drei verschiedene Grade des Testens unterscheiden:

- Keine Testszenarien nötig
- Während der Implementation bis zum Ergebnis, schrittweises Testen
- gesamter Funktionstest

Einzelne Module, welche die gesamte Implementationszeit in vollem Umfang gebraucht wurden, der Database Abstraction Layer (DAL) und das Generieren des XML-Files, wurden nicht speziellen Testszenarios ausgesetzt. Für diese Module gibt es keine Unterscheidung zwischen Implementation und Produktion, da diese durchwegs gebraucht wurden.

Einzelne Module wurden während der Implementation ständig weiterentwickelt, um sie den verschiedenen Anforderungen des Entwicklungsstandes anzupassen, oder funktionale Schwächen zu verbessern. Diese wurden immer wieder mit anderen Modulen zusammengehängt und so von verschiedenen Anwendern getestet und auf spezielle Funktionalität überprüft. Fehler wurden schnell erkannt und mussten dementsprechend schnell behoben werden.

Schlussendlich erfolgte der gesamte Funktions-Test dieser Applikation. Dieser wurde in zwei Teile unterteilt:

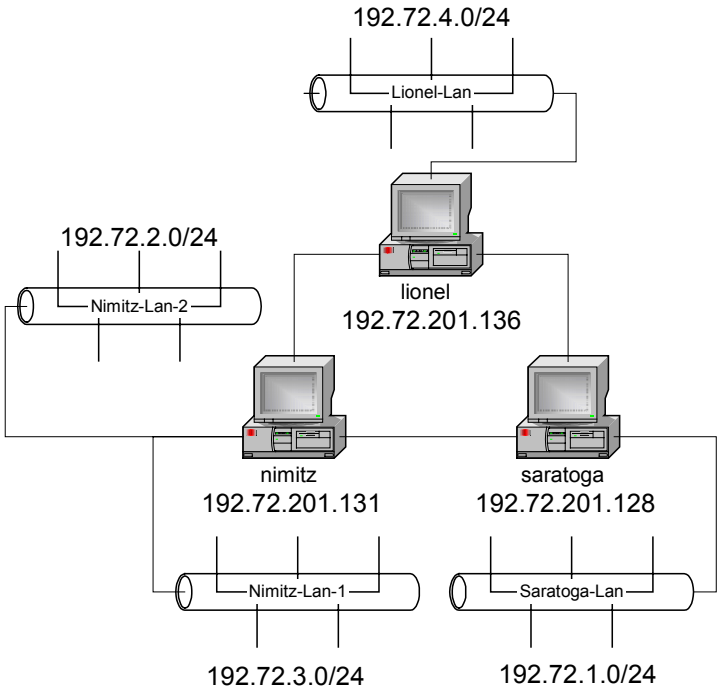
- Bedienung: Gibt es Fehler in den Bedienelementen.
- Output: Kommt bei einer bestimmten Eingabe das richtige Ergebnis heraus.

Der erste Teil bestand daraus alle Bedienungsmöglichkeiten zu testen, indem die Datenbank mit Testwerten gefüllt, diese geändert, gelöscht oder nur angezeigt wurden. Es sollten möglicht viele Kombinationen der verschiedenen Bedienelemente getestet werden.

Im zweiten Teil wurden Daten eingegeben, welche verwendet werden konnten, sprich IPSec-Verbindungen mit den generierten Konfigurationsfiles aufbauen. Es konnten nur einzelne Verbindungen getestet werden, da die Anzahl der Testgeräte nicht für einen vollumfänglichen Funktionstest ausreichten. Auch konnte das Verteilen der Files nicht so getestet werden, wie es im Betrieb eingesetzt werden soll, da sie benötigten SSH Trusted-Host-Verbindungen nicht hergestellt werden konnten. Es wurde bei den Testgeräten immer die Passphrase verlangt, welche im Betrieb nicht benötigt werden. Das Testen der Funktionalität konnte jedoch durchgeführt werden, indem das von der Applikation aufgerufene Shell-Skript von der Konsole aus gestartet wurde.

8.1 Testszenario

Der Funktionstest wurde mit 3 Linux-Rechnern durchgeführt. Es wurden alle Möglichen Meshing Typen und Authentifizierungs-Varianten eingesetzt.



Meshning-Type	Name	Authentifizierung	IPSec-Mode
FULL	Full-Mesh	preshared	Tunnel
HUB	Hub-Mesh	Zertifikate	Tunnel
POINT	Point-1	Zertifikate	Transport
POINT	Point-2	Zertifikate	Transport

Die Protected-Networks (LAN) sind fiktiv, um die Richtigkeit der Konfigurationsfiles im Tunnel-Mode testen zu können (für jedes Protected-Network eine eigene Verbindung) . Die beiden point to Point Verbindungen zwischen Saratoga und Lionel, beziehungsweise Saratoga und Nimitz wurden mit verschiedenen Zertifikaten durchgeführt. Hier konnte auch getestet werden, ob eine IP-Adresse als Subject-Alt-Name eingetragen wird, wenn als Common-Name eine solche beim Erstellen eines Zertifikates übergeben wird.

Bei den Preshared Verbindungen müssen die generierten Preshared-Secrets bei allen Verbindungen übereinstimmen, damit eine Verbindung aufgebaut werden kann. Weiter müssen im `ipsec.secrets` von Saratoga die Private-Key für das Point-1- und das Point-2 Mesh unterschiedlich sein und verschiedene Subject-Alt-Name-Einträge aufweisen.

Es konnte kein Testfall für einen Security Gateway mit mehreren Interfaces durchgeführt werden, da alle Testrechner mit nur einer Netzwerkkarte ausgestattet waren.

Die gesamten Testergebnisse befinden sich im Anhang, sowie die erstellten Verzeichnisse und Zertifikate sind auf der CD enthalten.

Das Testen hat sich als nicht so einfach erwiesen, als anfänglich angenommen, da die Infrastruktur für ein gesamten Funktionstest nicht ausreichend und schlussendlich auch die nötige Zeit zu knapp war.

9 Integration

Die Integration auf das aktuelle System der Firma Open System AG konnte nicht durchgeführt werden, da zu diesem Zeitpunkt keine neue Software auf den Webserver aufgespielt werden konnte.

In Absprache mit unserem Betreuer Seitens Open System AG, Lionel Gresse, wurde der Entscheid getroffen, dass die Integration unserer Applikation von ihm vorgenommen wird.

10 Gebrauchsanweisung

10.1 Installation und Konfiguration

Um diese Applikation einwandfrei zu gebrauchen müssen folgende Pakete (Version gleich oder höher) installiert sein :

- Apache 1.3.20 [17]
- Apache php-Modul 4.0.6 [17]
- openssl 0.9.6 [11]

Zuerst muss das Paket vpnadmin_1.0.tar.gz in das gewünschte Verzeichnis entpackt werden. Danach sollte für alle installierten Verzeichnisse und Files der Besitzer auf `wwwrun` geändert werden. Dies ist sehr wichtig, da der ausführende Prozess der Webserver ist und somit die Rechte für diesen gesetzt werden müssen.

Der nächste Schritt ist, im `/etc/php.ini` in den `include`-Pfad das Verzeichnis `vpnadmin` einzutragen, damit die `include`-Anweisungenin den PHP-Files richtig interpretiert werden.

```
include_path = <VPN-Directory>
```

Der letzte Schritt beinhaltet das anpassen der vpn_config Datei. In dieser muss der erste Eintrag, `$VPNDIR` auf das soeben installierte vpnadmin-Verzeichnis gesetzt werden. Alle anderen Verzeichnis-Einträge sind Unterverzeichnisse, welche nicht geändert werden muss, ausser man möchte die vorgegebene Verzeichnisstruktur anpassen.

Für die Datenbankverbindung müssen der Benutzer, das Passwort, der Datenbankname, sowie die Verbindungsart (nur für eine Oracle Datenbank) angegeben werden:

```
$configurations = array("conf1" =>
                        array( "dbuser=> "vpn",
                               "dbpassword=> "c_sharp",
                               "dbconnectString" => "stopdev",
                               /* OCI Driver use only
                               persistent - new - default */
                               "dbLogon=> "persistent"),
                                          ...
```

Im Verzeichnis `vpnadmin/src/utils/openssl.orig` befindet sich ein `openssl.cnf` File, welches als Vorlage zur Generierung einer eigenen Zertifizierungsstelle dienen soll. Nachdem dieses angepasst wurde, muss es in das Verzeichnis /vpnadmin/src/utils kopiert werden, bevor die CA erstellt wird. Dies ist, weil zum Erstellen einer CA [4.XX] ein solches File vorhanden sein muss und dies in diesem Verzeichnis gesucht wird. Nach dem Erzeugen einer CA wurde es zusammen mit dem `index.txt` File in das für diese CA erstellte Verzeichnis `vpnadmin/src/utils/<ca-name>` kopiert.

10.2 Bedienung

Der zentrale Bestandteil des VPNadmin-Tools ist die Datenbank. In ihr werden sämtliche Informationen zu einem Netzwerk gespeichert. Der Benutzer greift über das grafische Userinterface auf die Datenbank zu, welches eine einfache Manipulation der Daten ermöglicht und deren Konsistenz gewährleistet. Im folgenden wird das Vorgehen beim Arbeiten mit dem Tool beschrieben.

Zu Beginn ist die Datenbank fast leer. Lediglich die Tabellen `IPSec_algorithm`, `key_type` und `meshing_type` enthalten einige Werte. In diese Tabellen können keine Daten mit dem GUI eingegeben werden, da es sich um statische Daten handelt, welche bei normalen Gebrauch nicht geändert werden. Sollte es dennoch einmal zu Neuerung oder Anpassung kommen, können diese vom Datenbankadministrator mit einem SQL-Tool vorgenommen werden.

- In einem ersten Schritt muss eine CA zu generiert werden. Dies geschieht mit der Eingabemaske CA im Bereich insert.
- Als nächstes werden alle bekannten Security Gateways mit der Maske node eingetragen. Jedem Knoten muss ein Typ zugewiesen werden.
- Den erstellten Knoten müssen ein oder mehrere LAN's zugewiesen werden.
- Den Knoten werden Interfaces zugewiesen. Ein Security Gateway kann ein oder mehrere Interfaces besitzen.
- Wenn ein VPN mit Zertifikaten erstellt wird, benötigen die beteiligten Interfaces mindestens ein Zertifikat. Handelt es sich bei einer Verbindung um eine preshared Verbindung ist es nicht nötig, Zertifikate zu generieren.
- Der wichtigste Schritt ist das Zusammenstellen der Meshings. Es wird zwischen den drei Meshingarten Full, Hub oder Point-to-Point unterschieden. Ein Interface kann gleichzeitig in mehreren Meshings vorkommen. An jedem Meshing sind mindestens zwei Interfaces beteiligt, wobei bei einem Hub-Meshing noch ein zentrales Interface angegeben werden muss. Optional kann noch eine CA angegeben werden. Ist dies der Falle, wird geprüft, dass alle Zertifikate der angegebenen Interfaces von dieser CA stammen.
- Sind alle Meshings in der Datenbank eingetragen, kann der User die Konfigurationsfiles generieren und deren Verteilung an die entsprechenden Knoten auslösen. Um die Konfigurationsfiles generieren zu können, wird ein XML-File pro Knoten erstellt, welches direkt, zusammen mit dem Erstellungsdatum und dem Namen des Users, in die Datenbank geschrieben wird.

Mit den unter update aufgeführten Masken können Änderungen an den in der Datenbank befindenden Daten vorgenommen werden. Aus Zeitgründen konnten allerdings nur die update-Masken für Gateways und Meshings realisiert werden. Mit den Masken aus dem delete-Bereich können ganze Datensätze aus der Datenbank gelöscht werden.

Die Einträge der Datenbank können im Bereich show angezeigt werden. Mit den Buttons am rechten Rand der show-Masken kann in die entsprechende Maske aus dem Bereich update oder delete gesprungen werden, wobei das ausgewählte Element übernommen wird.

Änderungen an der Datenbank, egal ob durch updaten oder löschen von Daten, haben in den meisten Fällen zur Folge, dass die Konfigurationsfiles der Security Gateways neu generiert und verschickt werden müssen. Obwohl das Programm das Generieren und Verschicken übernimmt, ist der User selber dafür Verantwortlich dieses Ereignis auszulösen. Dies macht er in einer speziellen Maske, in welche er über den Link "apply" in der Titelleistennavigation gelangt. Es werden alle Konfiguratiosfiles neu generiert und verschickt, egal ob ein Gateway von einer Modifikation betroffen ist oder nicht.

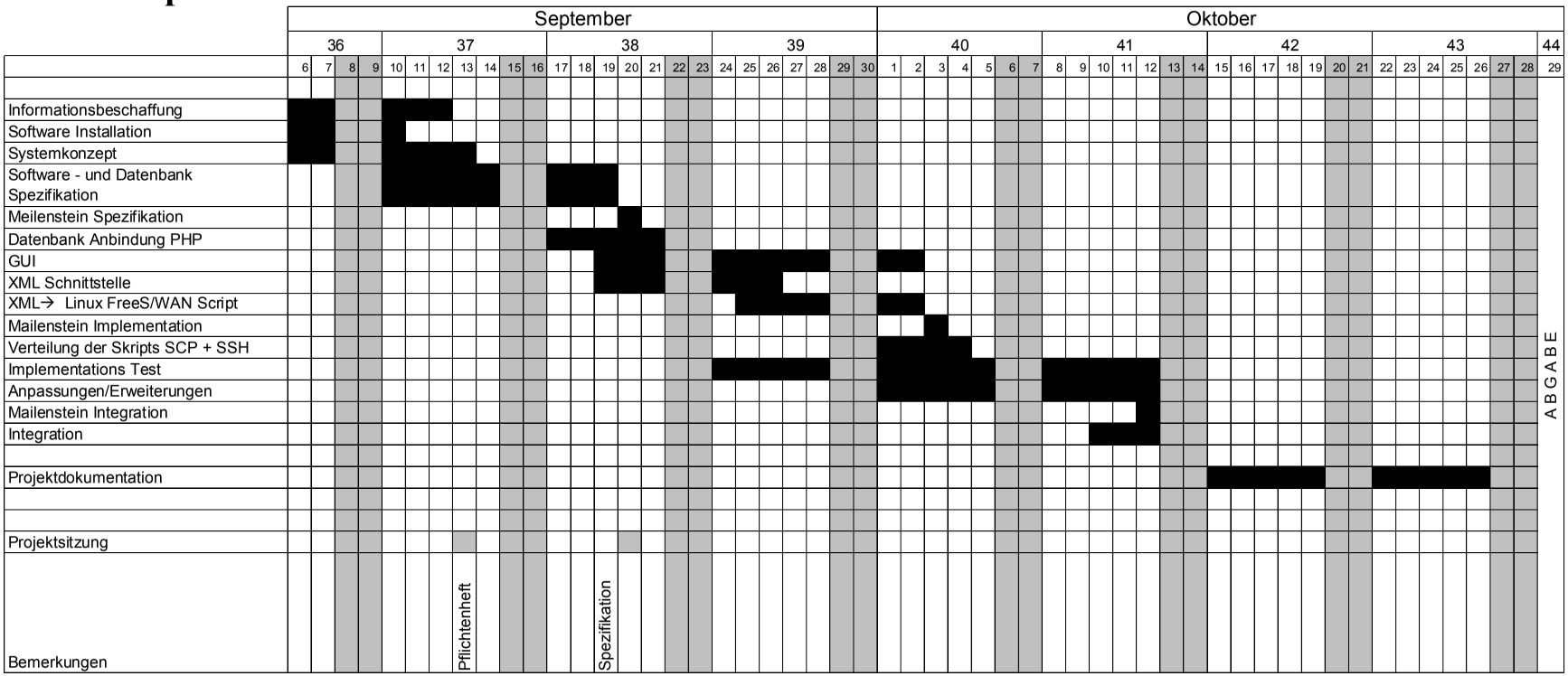
Erstellen eines Zertifikates

Beim Erstellen eines Zertifikates ist zu beachten, dass zwei Möglichkeiten für den Subject-Alt-Name unterstützt werden:

- Email-Adresse
- IPv4-Adresse

Die Unterscheidung bei der Eingabe der verschiedenen Parameter werden beim "Common Name" gemacht. Gibt man dort eine IP-Adresse an, wird diese als Subject-Alt-Name übernommen. Wird ein Name oder eine sonstige Angabe eingegeben, wird die Email-Adresse vom Eingabefeld "email" als Subject-Alt-Name übernommen.

11 Zeitplan


September
Oktober
36
37
38
39
40
41
42
43
44
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 24 25 26 27 28 29
Informationsbeschaffung
Software Installation
Systemkonzept
Software - und Datenbank Spezifikation
Meilenstein Spezifikation
Datenbank Anbindung PHP
GUI
XML Schnittstelle
XML→ Linux FreeS/WAN Script
Mailenstein Implementation
Verteilung der Skripts SCP + SSH
Implementations Test
Anpassungen/Erweiterungen
Mailenstein Integration
Integration
Projektdokumentation
Projektsitzung
Bemerkungen
Pflichtenheft
Spezifikation
ABGABE

12 Ausblick

Das in dieser Arbeit entwickelte VPN-Administrations-Tool stellt ein Filter für die Linux IPSec-Implementation (FreeS/WAN) zur Verfügung. Ineinem weiteren Schritt könnte hier verschiedene andere IPSec Produkte unterstützt werden. Zusätzlich können verschiedenen Features, welche sich beim intensiven Benutzten des Tools als nützlich erweisen, hinzugefügt werden.

Im Laufe dieser Arbeit wurden wir auf einige Verbesserungen aufmerksam, welche aus zeitlichen Gründen nicht mehr aufgegriffen werden konnten. Zum Beispiel werden beim Versenden der Konfigurationsfiles, für alle in der Datenbank vorhandenen Knoten diese neu generiert und auf die jeweiligen Security Gateways kopiert. Dies ist sicher noch so zu Implementieren, dass nur die Files neu generiert werden, für welche Einträge Änderungen hervorrufen.

Ebenfalls aus Zeitmangel konnten nicht alle Funktionen in der update-Oberfläche implementiert werden. Funktionell können mit den vorhandenen Bedienelementen jedoch die erforderlichen Eingaben durchgeführt werden.

Wir hoffen, das diese Arbeit eine Basis für eine ernsthafte Anwendung stellt, welche kontinuierlich verbessert und erweitert wird.

13 Schlusswort und Danksagung

13.1 Schlusswort

In dieser Diplomarbeit konnten wir die Theorie der verschiedensten Themen in eine praktische Anwendung einfliessen lassen. Gerade dieser Umstand gab der Arbeit einen besonderen Reiz. Auch haben wir gemerkt, dass wir eine der wichtigsten Fähigkeiten, welche uns in dieser Schule vermittelt wurde, nämlich das lernen zu lernen tatsächlich in die Tat umsetzen konnten. So haben wir uns in kurzer Zeit in PHP und XML eingearbeitet.

Ein weiterer positiver Aspekt war, dass wir während in der Firma Open Systems AG arbeiten konnten. Dies hat uns einen Einblick in eine mögliche Berufswelt gebracht. Auch konnten wir steht's auf die Erfahrung und das Know-how der Leute von Open Systems zurückgreifen.

Zu Beginn der Arbeit haben wir eine ausführliche Spezifikation ausgearbeitet. Dies brachte uns sicher den Vorteil, dass wir von Anfang an wussten, was genau zu Implementieren ist, was der Kunde, in unserem Fall die Firma Open Systems AG, genau wollte und welche Features wir definitiv aus dieser Version der Arbeit ausschliessen konnten. Aber es bot auch einen Nachteil: Da wir uns an diese Spezifikation zu halten hatten legten wir uns einige Steine in den Weg. Denn im Laufe der Zeit sammelten wir immer mehr Erkenntnisse über die verschiedensten Punkte, welche wir heite sicher anders anpacken würden. Vor allem beim Datenbankschema würden wir einige Änderungen vornehmen.

Die Zeit war ein entscheidender Faktor, welcher gegen Ende dieser Arbeit immer erdrückender wurde. Der Grat zwischen einer lauffähigen Implementaion und genügend Zeit für die Dokumentation ist sehr schmal. Deshalb musste der von uns erstellte Zeitplan öfters als uns genehm war, korrigiert werden.

13.2 Danksagung

An dieser Stelle möchten wir uns bei allen Personen bedanken, die uns bei unserer Diplomarbeit unterstützt haben. Von Seiten der ZHW ist dies Herr Steffen, der uns an den wöchentlichen Meetings, aber auch zwischendurch per e-Mail immer wieder wertvolle Tipps, Vorschläge und Antworten gegeben hat. In der Firma Open Systems AG möchten wir besonders bei Lionel Gresse und Oliver Kessler bedanken. Sie haben uns geholfen Linux zu überlisten, die Datenbank aufzusetzen, die Spezifikation auszuarbeiten und, sofern möglich, auf unsere Fragen Antworten zu geben. Natürlich geht unser Dank auch an alle anderen Personen in der Open Systems AG, welche hier nicht speziell aufgelistet sind.

14 Quellen

Literatur

[1] Jesus Castagnetto, "Proffessional PHP Programming", Wrox Press Ltd.
IDBN 1-861002-96-3

[2] Erik T. Ray, "Learning XML", O'REILLY & Assosiates Inc.
ISBN 0-596-00046-4

[3] Musciano & Kennedy, "HTML & XHTML", O'REILLY & Assosiates Inc.
ISBN 0-596-00026-X

[4] David Flanagan, "Java Script", O'REILLY & Assosiates Inc.
ISBN 1-5692-392-8

[5] Elmasri & Navathe, "Fundamentals of Database Systems", Addison-Wesley
ISBN 0-201-54263-3

[6] David C. Kreins, "Oracle SQL", O'REILLY & Assosiates Inc.
ISBN 1-56592-697-8

[7] Neil Mathew & Richard Stones, "Linux Programming", Wrox Press Ltd.
ISBN 3-8266-0569-1

Links

[8]	FreeS/WAN Project:	http://www.freeswan.org
[9]	X.509-Patch-FreeS/WAN	http://www.strongsec.com
[10]	OpenSSL Project:	http://www.openssl.org
[11]	Das OpenSSL Handbuch:	http://www.cert.dfn.de/dfnpca/certify/ssl/handbuch/ossl092/
[12]	X.509 Zertifikate:	http//freecert.entrust.com
[13]	PHP Programmierung:	http//www.php.net http://www.php.com http://www.phpbuilder.com http://php-center.com
[14]	Database Abstract Layer	http://www.sourceforge.net
[15]	SSH	http://www.bauv.unibw-muenchen.de/institute/inst6_2/forschung/regi/bericht98.24.07.98/node29.html
[16]	/dev/random:	http://www.rt.com/man/random.4.html
[17]	SuSE-Linix Distribution	http/www.suse.de

Bisherige Arbeiten

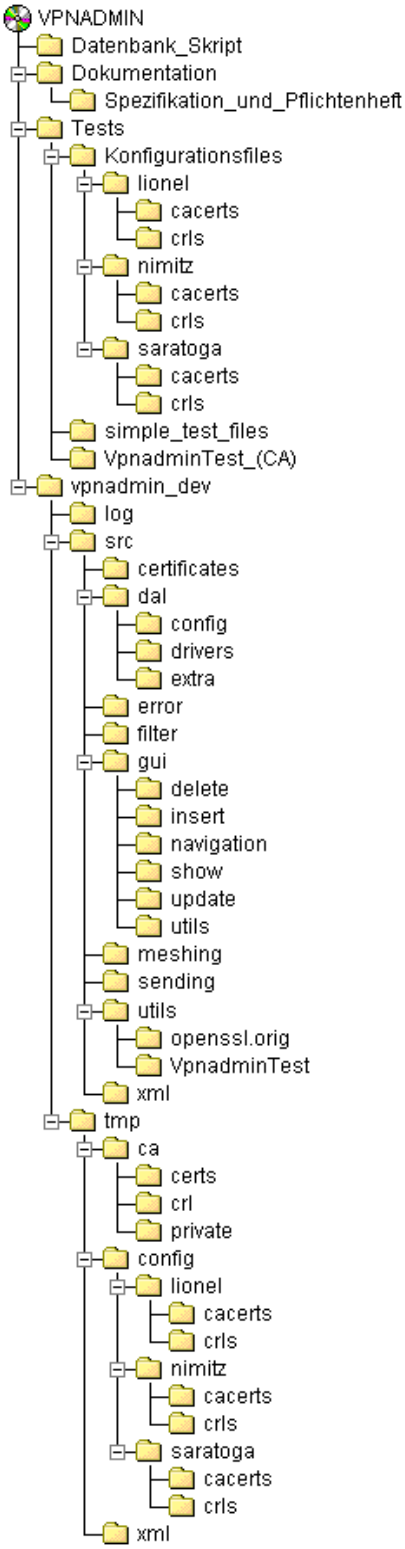
[18] Diplomarbeit 1999, "Virtual Private Network", Olivier Gärtner, Berkant Uenal

[19] Projektarbeit 2000, "IPSec mit PGP und X.509 Zertifikaten",
Andreas Hess, Patric Lichtsteiner, Roger Wegmann

[20] Projektarbeit, "X.509 Trust Chain Unterstützung",
Marco Bertossa, Andreas Schleiss

15 Anhang

15.1 Verzeichnisstuktur der CD



15.2 XML Parameter

Element	Attribut / Element	Value	Optional	Beschreibung
Node	Name		No	Hostname des Knotens
	Mgmt-IP1	xxx.xxx.xxx.xxx	No	IP, mit welcher eine SSH – Verbindung aufgebaut wird, um die Skripts zu verteilen.
	Mgmt-IP2	xxx.xxx.xxx.xxx	Yes	2. Management IP des Knotens, falls vorhanden
Node-Type	Name	FreeS/WAN, CheckPoint, Nokia, …	No	Typ des IPSec-Gateways (
	OS	Linux, Unix, NT	No	Betriebssystem des Knotens
	Version		No	Version der IPSec Implementaion
	Ipsec-Alg	des-cbc, 3des-cbc, blowfish-cbc, cast128-cbc, des-deriv, 3des-deriv, rijndael-cbc	Yes	Angabe der unterstützten Verschlüsselungs Algorithmen der IPSec Implementation
Meshing	Name		No	Name des Meshings
	Typ	FULL | HUB | POIN	No	Typ des Meshings
	IPSec-Mode	tunnel | transport	No	Definiert den IPSec Mode
	IPSec-Type	certificate | preshared	No	Art der IPSec Authentisierung
	IKE-Lifetime	1 – 8 Stunden	No	Zeit der Gültigkeit eines ausgehandelten Schlüsselpaares
	PFS	Yes | no	No	Aktivieren / deaktivieren der "Perfekt Forward Secrecy"
Interface	IP	xxx.xxx.xxx.xxx	No	IPSec Gateway IP Adresse
	Name	z.B "eth0"	No	Name der Netzwerkkarte im Betiebssystem

	Central Node	xxx.xxx.xxx.xxx	Yes	Nur vohanden bei einem HUB-Meshing
	CA		Yes	CA Zertifikat im PEM (base 64) Format, falls IPSec Type = certificate
	CRL		Yes	CRL Zertifikat im PEM (base 64) Format, falls IPSec Type = certificate
	Certificate		Yes	X.509 Zertifikat im PEM (base 64) Format, falls IPSec Type = certificate
	Private Key		Yes	Zum Zertifikat gehöriger Private Key PEM (base 64) Format
	Passphrase		Yes	Passwort des Private Keys als Base 64 String
	Subject-Alt-Name	IP | EMail	Yes	Subject Alt Name des Zertifikates
	Public Key		Yes	Zum Zertifikat gehöriger Public Key (HEX Format)
Protected Network	IP	xxx.xxx.xxx.xxx	Yes	Netzwerk hinter dem Security Gateway
	Netmask	xx	Yes	Subnetz Maske in der /XX schreibweise
	Netmask	xx	Yes	Subnetz Maske in der /XX schreibweise
Connection	Number		No	Anzahl der Verbindungen
	Remote IP	xxx.xxx.xxx.xxx	No	Remote IP Adresse
	Remote Name		No	Hostname des Remote Security Gateways
	Preshared Key		Yes	Preshared Key einer Verbindung im Base 64 Format, falls IPSec-Type = preshared

15.2 openssl.cnf

```
    # OpenSSL example configuration file.
    # This is mostly being used for generation of certificate
requests.
    #
    # To use this configuration file with the "-extfile" option of
the
    # "openssl x509" utility, name here the section containing the
    # X.509v3 extensions to use:
    # extensions            =
    # (Alternatively, use a configuration file that has only
    # X.509v3 extensions in its main [= default] section.)

    path                    = $VPNDIR
    RANDFILE                = $path/src/utils/.rand


##################################################################

    [ ca ]

    default_ca      = User_CA                # The default ca
section


##################################################################


    [ Root_CA ]     # Abschnitt fuer eine Root CA

    dir             = $path/tmp/ca           # Where everything is
kept
    #certs           = $dir/certs             # Where the issued
certs are kept
    crl_dir         = $dir/crl               # Where the issued crl
are kept
    database        = $path/src/utils/index.txt # database index
file.
    new_certs_dir   = $dir/certs             # default place for new
certs.

    certificate     = $dir/RootCA.pem        # The CA certificate
    serial          = $path/src/utils/serial  # The current serial
number
    crl             = $dir/crl.pem           # The current CRL
    private_key     = $dir/private/CAkey.pem # The private key

    x509_extensions = root_ext                # The extentions to
add to the cert

    # Extensions to add to a CRL. Note: Netscape communicator chokes
on V2 CRLs
    # so this is commented out by default to leave a V1 CRL.
    #crl_extensions = crl_ext                 # Extensions to add to
CRL

    default_days    = 730                    # how long to certify
for
    default_crl_days= 30                     # how long before next
CRL
    default_md      = md5                    # which md to use.
    preserve        = no                     # keep passed DN
ordering

    # A few difference way of specifying how similar the request
should look
```

```
    # For type CA, the listed attributes must be the same, and the
optional
    # and supplied fields are just that :-)

    policy          = policy_match


#####################################################################
#

    [ User_CA ]         # Abschnitt fuer eine User CA

    dir             = $path/tmp/ca            # Where everything is
kept
    #certs           = $dir/certs              # Where the issued
certs are kept
    crl_dir         = $dir/crl                # Where the issued crl
are kept
    database        = $path/src/utils/index.txt # database index
file.
    new_certs_dir   = $dir/certs              # default place for new
certs.

    certificate     = $dir/RootCA.pem         # The CA certificate
    serial          = $path/src/utils/serial  # The current serial
number
    crl             = $dir/crl.pem            # The current CRL
    private_key     = $dir/private/CAkey.pem # The private key

    x509_extensions = user_ext                # The extentions to add
to the cert
    #crl_extensions = crl_ext                 # Extensions to add to
CRL
    default_days    = 365                     # how long to certify
for
    default_crl_days= 30                      # how long before next
CRL
    default_md      = md5                     # which md to use.
    preserve        = no                      # keep passed DN
ordering

    policy          = policy_anything


#####################################################################
#

    # For the CA policy
    # Auch hier gilt:
    # ... you must list all acceptable 'object' types.

    [ policy_match ]

    countryName             = match
    stateOrProvinceName     = optional
    localityName            = optional
    organizationName        = supplied
    organizationalUnitName  = optional
    commonName              = supplied
    emailAddress            = optional

    # For the 'anything' policy
    # At this point in time, you must list all acceptable 'object'
    # types.

    [ policy_anything ]

    countryName             = match
```

```
    stateOrProvinceName     = optional
    localityName            = optional
    organizationName        = optional
    organizationalUnitName  = optional
    commonName              = supplied
    emailAddress            = optional


#####################################################################

    [ req ]

    default_bits        = 1024
    default_keyfile     = privkey.pem
    distinguished_name  = req_distinguished_name
    attributes          = req_attributes
    x509_extensions     = v3_ca     # The extentions to add to the
self signed cert

    # Passwords for private keys if not present they will be prompted
for
    # input_password = secret
    # output_password = secret

    # This sets a mask for permitted string types. There are several
options.
    # default: PrintableString, T61String, BMPString.
    # pkix   : PrintableString, BMPString.
    # utf8only: only UTF8Strings.
    # nombstr : PrintableString, T61String (no BMPStrings or
UTF8Strings).
    # MASK:XXXX a literal mask value.
    # WARNING: current versions of Netscape crash on BMPStrings or
UTF8Strings
    # so use this option with caution!
    string_mask = nombstr

    # req_extensions = v3_req # The extensions to add to a
certificate request


#####################################################################
#

    [ req_distinguished_name ]

    countryName                     = Country Name (2 letter code)
    countryName_default             = XX
    countryName_min                 = 2
    countryName_max                 = 2

    #stateOrProvinceName            = State or Province Name (full
name)
    #stateOrProvinceName_default    = State

    localityName                    = Locality Name (eg, city)
    localityName_default            = Locality

    0.organizationName              = Organization Name (eg, company)
    0.organizationName_default      = Organisation

    # we can do this but it is not needed normally :-)
    #1.organizationName             = Second Organization Name (eg,
company)
    #1.organizationName_default     = Organisation

    organizationalUnitName          = Organizational Unit Name (eg,
section)
```

```
    organizationalUnitName_default  = Unit

    commonName                      = Common Name (eg, YOUR name)
    commonName_max                  = 64

    emailAddress                    = Email Address
    emailAddress_max                = 60

    # SET-ex3                       = SET extension number 3


#####################################################################
#

    [ req_attributes ]

    # Das Challenge Password dient dazu, sich bei Verlust des
geheimen Schluessels
    # gegenueber der Herausgeber-CA fuer einen Zertifikatwiderruf
auszuweisen.
    # Wird bei Erstellung der Zertifikat-Anforderung erfragt.

    challengePassword               = A challenge password
    challengePassword_min           = 4
    challengePassword_max           = 20

    #unstructuredName                = An optional company name


#####################################################################
#

    [ root_ext ]

    # This goes against PKIX guidelines but some CAs do it and some
software
    # requires this to avoid interpreting an end user certificate as
a CA.
    basicConstraints         = critical, CA:TRUE

    # Moeglich: digitalSignature, nonRepudiation, keyEncipherment,
    #           dataEncipherment, keyAgreement, keyCertSign,
    #           cRLSign, encipherOnly, decipherOnly
    keyUsage                 = cRLSign, keyCertSign

    # PKIX recommendations
    subjectKeyIdentifier     = hash
    authorityKeyIdentifier   = keyid,issuer:always

    # Import the email address.
    subjectAltName           = email:copy

    # Copy subject details
    issuerAltName            = issuer:copy

    crlDistributionPoints    = URI:http:/.../ca/PCA.crl

    # Moeglich: client, server, email, objsign, reserved, sslCA,
emailCA, objCA
    nsCertType               = sslCA, emailCA, objCA

    # Hier kann der den folgenden Url's gemeinsame Url-Stamm
angegeben werden.
    nsBaseUrl                = https://.../ca/

    # Die Seite mit der CA-Policy
    nsCaPolicyUrl            = http://.../ca/policy.html
```

```
    # Ein Text, der von Netscape-Browsern angezeigt wird
    nsComment               = This certificate was issued by a
PCA\n\
    just for testing.

    # Hier kann eine Online-Zertifikatspruefung stattfinden, indem
auf die
    # Url in der Form ../foo.cgi?aaaa zugegriffen wird. "aaaa" ist
dabei
    # die ASCII-kodierte Seriennummer des Zertifikats. Dann kann das
Zertifikat
    # per OpenSSL geprueft werden. Wird vermutlich nur in
Serverzertifikaten und
    # durch Netscape-Browser unterstuetzt
    # Zurueckgegeben wird eine dezimale 0 oder 1
    # nsRevocationUrl         = cgi/non-CA-rev.cgi?

    # Nur gueltig in CA-Zertifikaten. Bedeutung nicht ganz klar.
    # nsCaRevocationUrl       = cgi/CA-rev.cgi?

    # Wird verwendet, um einem Benutzer die Erneuerung seines
Zertifikats zu
    # erleichtern. Ueblicherweise steckt dahinter ein CGI-Script, auf
das per
    # HTTP GET in der Form ../foo.cgi?aaaa zugegriffen wird. "aaaa"
ist wieder
    # Seriennummer. Zurueckgegeben werden kann ein Antrags-Formular
zur Erneuerung
    # des Zertifikats.
    # nsRenewalUrl            = cgi/check-renw.cgi?


###################################################################
#

    [ user_ext ]

    # basicConstraints        = critical, CA:FALSE
    keyUsage                = digitalSignature, keyEncipherment,
keyAgreement
    subjectKeyIdentifier    = hash
    authorityKeyIdentifier  = keyid,issuer:always
    subjectAltName          = email:copy
    issuerAltName           = issuer:copy
    crlDistributionPoints   = URI:http://.../ca/UCA.crl
    nsCertType              = client, email, objsign
    nsBaseUrl               = https://.../ca/
    nsCaPolicyUrl           = http://.../policy.html
    nsComment               = This certificate was issued by a User
CA
    nsRevocationUrl         = cgi/non-CA-rev.cgi?
    # nsRenewalUrl          = cgi/check-renw.cgi?


###################################################################
#

    [ v3_ca ]

    basicConstraints        = critical, CA:TRUE
    subjectKeyIdentifier    = hash
    authorityKeyIdentifier  = keyid:always,issuer:always
    keyUsage                = cRLSign, keyCertSign
    nsCertType              = sslCA, emailCA, objCA
    subjectAltName          = email:copy
    issuerAltName           = issuer:copy
    crlDistributionPoints   = URI:http://.../ca/RCA.crl
    nsBaseUrl               = https://.../ca/
```

```
    nsCaPolicyUrl            = http://.../ca/policy.html
    nsComment                = This certificate is a Root CA
Certificate


    # RAW DER hex encoding of an extension: beware experts only!
    # 1.2.3.5=RAW:02:03
    # You can even override a supported extension:
    # basicConstraints       = critical, RAW:30:03:01:01:FF


#####################################################################
#

    [ crl_ext ]

    # CRL extensions.
    # Only issuerAltName and authorityKeyIdentifier make any sense in
a CRL.

    issuerAltName            = issuer:copy
    authorityKeyIdentifier   = keyid:always,issuer:always
```

15.3 saratoga.xml

```
<?xml version="1.0"?>
<!-- Created 2001.10.27:15:05:49 -->

  <node name="saratoga">
    <head>
      <mgmt_IP_1>193.72.201.128</mgmt_IP_1>
      <mgmt_IP_2>
      </mgmt_IP_2>
      <node_type name="FreeS/WAN">
        <os>Linux</os>
        <version>0.9.1</version>
        <ipsec_alg>
        </ipsec_alg>
      </node_type>
      <protected_network>
        <ip>193.72.1.0</ip>
        <netmask>2</netmask>
      </protected_network>
    </head>
    <meshing name="Full_Mesh" type="FULL">
      <ipsec_type>preshared</ipsec_type>
      <central_node>n/a</central_node>
      <ipsec_mode>tunnel</ipsec_mode>
      <ike_lifetime>3600</ike_lifetime>
      <pfs>yes</pfs>
      <interface ip="193.72.201.128" name="eth0">
        <subject_alt_name>n/a</subject_alt_name>
        <ca><![CDATA[n/a]]></ca>
        <crl><![CDATA[n/a]]></crl>
        <certificate><![CDATA[n/a]]></certificate>
        <passphrase>n/a</passphrase>
        <private_key><![CDATA[n/a]]></private_key>
        <public_key>n/a</public_key>
      </interface>
      <connections number="2">
        <connection name="saratoga-nimitz">
          <remote_ip>193.72.201.131</remote_ip>
          <remote_name>nimitz</remote_name>
          <remote_subject>n/a</remote_subject>
          <protected_network>
            <ip>193.72.2.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <protected_network>
            <ip>193.72.3.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <preshared_key>p1c/C+jVlSpJP6mHYVAH</preshared_key>
        </connection>
        <connection name="saratoga-lionel">
          <remote_ip>193.72.201.136</remote_ip>
          <remote_name>lionel</remote_name>
          <remote_subject>n/a</remote_subject>
          <protected_network>
            <ip>193.72.4.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <preshared_key>AmPHKgu8XfY5z0YjtbLW</preshared_key>
        </connection>
      </connections>
    </meshing>
    <meshing name="Point-1" type="POINT">
      <ipsec_type>certificate</ipsec_type>
      <central_node>n/a</central_node>
```

```
      <ipsec_mode>transport</ipsec_mode>
      <ike_lifetime>3600</ike_lifetime>
      <pfs>yes</pfs>
      <interface ip="193.72.201.128" name="eth0">
        <subject_alt_name>193.72.201.128</subject_alt_name>
        <ca><![CDATA[-----BEGIN CERTIFICATE-----
MIIFTjCCBDagAwIBAgIBADANBgkqhkiG9w0BAQQFADBzMQswCQYDVQQGEwJDSDEP
MA0GA1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxME
dGVzdDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1p
bi5jaDAeFw0wMTEwMjIxNjE0NTNaFw0wMjAxMzAxNjE0NTNaMHMxCzAJBgNVBAYT
AkNIMQ8wDQYDVQQHEwZ6dXJpY2gxFDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYD
VQQLEwR0ZXN0MREwDwYDVQQDEwhWcG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBu
QGFkbWluLmNoMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwwLYiURD
B1xLZUsIjRy9e6ia617qz1hnXwGkuVYQev8ahWOUJEqZR0bjNdm6LPqItZ9sclXj
NS/pNtirZqubGgj6rQrm5gYodizQSvK03oeiuLiBsxefqUpqaHjLBPfiNWxnZxi7
5YTxP0q3lj1D5s3rt60bP1CZmAYwNgANUT0OmaUk4hVUcPraCzPJ/CdlDc9KP+qZ
1iTXea+/zyuDGxalxJtvRq6xlgx4O79R7BWLKb7mg2Ty2fqwjROfPZ7+0Sf3vY2n
BiX6BDlXG8VF7cXUY29jHUxbFAhH8pGIspA6Sc24SzTuvgt7erueXayRcF7AdqdB
T9usiDaoRK/AbQIDAQABo4IB6zCCAecwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUgq5/01Dht0FWizCpdJHOv1n1Mw4wgZ0GA1UdIwSBlTCBkoAUgq5/01Dht0FW
izCpdJHOv1n1Mw6hd6R1MHMxCzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZ6dXJpY2gx
FDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYDVQQLEwR0ZXN0MREwDwYDVQQDEwhW
cG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBuQGFkbWluLmNoggEAMAsGA1UdDwQE
AwIBBjARBglghkgBhvhCAQEEBAMCAAcwFwYDVR0RBBAwDoEMdnBuQGFkbWluLmNo
MBcGA1UdEgQQMA6BDHZwbkBhZG1pbi5jaDAvBgNVHR8EKDAmMCSgIqAghh5odHRw
Oi8vd3d3Lm9wZW4uY2gvY2Evb3Blbi5jcmwwJgYJYIZIAYb4QgECBBkWF2h0dHBz
Oi8vd3d3Lm9wZW4uY2gvY2EvMDAGCWCGSAGG+EIBCAQjFiFodHRwOi8vd3d3Lm9w
ZW4uY2gvY2EvcG9saWN5Lmh0bWwwOAYJYIZIAYb4QgENBCsWKVRoaXMgY2VydGlm
aWNhdGUgaXMgYSBSb290IENBIENlcnRpZmljYXRlMA0GCSqGSIb3DQEBBAUAA4IB
AQCoVrifwabh3NvbL1fck5gfPcbgb+C4LBMacMOHyD94WY1aqatb4yO+nHy0r3jt
ZFSW8btDhsJ+xdvth5ABjlhbuBF32kN0OLi4kbC85H8zYo2OCm+oGzGe+Zqny2JF
/wbL2+oc3rMhacR9AtJxPDKrRAmZrMBOxwvn/syctXF9DKw4nCVA8B40nYQLyjV5
bH1jK5e1WrFAJHjOJ9ZPU+bcIHr7XD6UTJewI5ILBxMjjNtSFUe4QChKFapdkGmt
saW65CERm0ZMGSYRLMNT65MkJeLUkvCBqfsKc/OFokyEE3dsSKEqHd5+ZwM/dnvZ
BprS0bGcCgRVDYWHVEbxelIR
-----END CERTIFICATE-----
]]></ca>
        <crl><![CDATA[-----BEGIN X509 CRL-----
MIIBuTCBojANBgkqhkiG9w0BAQQFADBzMQswCQYDVQQGEwJDSDEPMA0GA1UEBxMG
enVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxMEdGVzdDERMA8G
A1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1pbi5jaBcNMDEx
MDI5MDAzNjQ4WhcNMDExMTI4MDAzNjQ4WjANBgkqhkiG9w0BAQQFAAOCAQEAEAaF
h4Hey2gKg3bLBLs8hIE5aeprZV1hEt4/XRFWU2fZTesgooT/poq8fwM5A1T1+He4
ObBSD2e4kkPtc90AX1sIwLpeRVKEjUmYdcxQoimL0ABheCbnefBPnwCQLCzQN0jH
MPiuXD5/OCMeuUlmrXU9qemI+5tmerYx37l+k4qVKCesu4L3bEUa583Kc17zZRor
nM8sZQBinjhqFj8d0izCJ3yeZEZRyjXb4Y7q9AOEQ5ts99oFCBYyfV1MLwUiVQ/k
ty/Db+edCOLm5UoR2lfnC/HUcj0A21kAYT8/pZQbLhNIeZJE/mqDhk600yu3x2Ka
gcWLXQP8mzXmc9SArA==
-----END X509 CRL-----
]]></crl>
        <certificate><![CDATA[-----BEGIN CERTIFICATE-----
MIIE3zCCA8egAwIBAgIBAzANBgkqhkiG9w0BAQUFADBzMQswCQYDVQQGEwJDSDEP
MA0GA1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxME
dGVzdDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1p
bi5jaDAeFw0wMTEwMjcwOTUyMjJaFw0wMjAyMDQwOTUyMjJaMHgxCzAJBgNVBAYT
AkNIMRMwEQYDVQQHEwp3aW50ZXJ0aHVyMQwwCgYDVQQKEwN6aHcxCzAJBgNVBAsT
Aml0MRcwFQYDVQQDEw4xOTMuNzIuMjAxLjEyODEgMB4GCSqGSIb3DQEJARYRYmVy
dG9tYXJAemh3aW4uY2gwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANDNYUW8
Hat6Y7HSHoXyi+SfkFbCD3MIEExn6OQJDuTB3W6p6O9iPCuj2eMxUNTD2OwFuNuP
9uymkyVZ1lIYSLKcyMkR+ABy3SesGOYUqfyQsDsitiz9wlfLsJ1+/kNfzDqvHWDD
H3pqRTpnU96n8uUOL7+3nA8sOwMBtBHLZ/wlAgMBAAGjggH7MIIB9zALBgNVHQ8E
BAMCA6gwHQYDVR0OBBYEFE+gVCuYX7ApF86+h406Zrj9c4DlMIGdBgNVHSMEgZUw
gZKAFIKuf9NQ4bdBVoswqXSRzr9Z9TMOoXekdTBzMQswCQYDVQQGEwJDSDEPMA0G
A1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxMEdGVz
dDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1pbi5j
aIIBADAXBgNVHRIEEDAOgQx2cG5AYWRtaW4uY2gwLwYDVR0fBCgwJjAkoCKgIIYe
aHR0cDovL3d3dy5vcGVuLmNoL2NhL29wZW4uY3JsMBEGCWCGSAGG+EIBAQQEAwIE
sDAmBglghkgBhvhCAQIEGRYXaHR0cHM6Ly93d3cub3Blbi5jaC9jYS8wMAYJYIZI
```

```
AYb4QgEIBCMWIWh0dHA6Ly93d3cub3Blbi5jaC9jYS9wb2xpY3kuaHRtbDA9Bglg
hkgBhvhCAQ0EMBYuVGhpcyBjZXJ0aWZpY2F0ZSB3YXMgaXNzdWVkIGJ5IE9wZW4g
U3lzdGVtcyBBRzAiBglghkgBhvhCAQMEFRYTY2dpL25vbi1DQS1yZXYuY2dpPzAP
BgNVHREECDAGhwTBSMmAMA0GCSqGSIb3DQEBBQUAA4IBAQCCNXc4NfhOxwoHQrz6
V3RXdVJLObJloQ7oAznmxZ0xR5eb2F3WTIvbEFk0PXOkpiI2RV5EF9tbR48t3xgc
BhFNrDU4R5MUsIDC7muzmO7UInOSLGSJsMjw5GZZ0mWIlsejudEkDu1YjpVY6Rv4
84S4qV81BVe9fFqz1h8BW2WKq4GOE0CVVx/jAKT/yALIYxhGNfz91mlxj2ZGF8db
pUGu5CjAEce/NcJjJj4/+57b2ZoqJE9c6dFfK7auxK/pSIM/L94fwXJJxady7EAn
TPi//x031CIjLzt+g/hGj2yY8y9r7hBATNtVBjOGlZxKKt2Le+IkDyrAawxT0oTl
gYCr
-----END CERTIFICATE-----
]]></certificate>
        <passphrase>5x8EgM+4ZekqRv+WSwc</passphrase>
        <private_key><![CDATA[-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,F73307E3CEF70FF3

M1MxXm+zoNA7cwxFWfea4QvhOYAQWevJvxEefdGx/fEPmRDOihusfMqISWAZkWKX
TYwQmyKP9+dGXgol82OnD6k1xawtfst9Vb+2ooiljM+uepVeeV4gcVR8Ts2g5gxr
RNK/kqcgP1fuYwLlf9bA6lWoFaK1jzmbbga/RY/Uqu/hJtTmfg/TW+UDpCmaTiM3
AY2EanltGr+Qs92pO5TT0/jYcs+JD5NqOp8u8q2pAwnaIr4wv5mezHvYxmnlWHxl
vysZRXljvDfdQWXVzJueLUrV1cGVLJxwiCs7FWmdiqrHMGv+NH2cKnw1JFYxcjb0
XFZhJC+8MvRqetFVtlQPzKwT9EJy7tkjtMjKEEAa8m5Z0wV1k3RgMmEV0RG1rR41
+28z9nAGKYioFh75IL1emNp+WVybLZ8qw/sMu3dxTb9TcQbb2t3pQcKUfRZWHjp4
EPGeJUwIOV3pH5vMf24whuAaIbvvda7cRbh/vSzBPNh8wOU49v6DfhQfqeiHBP7h
Wj/ybMYhdYR1b5ofg/TUgGJsIrHMf9t5Preer7pHAQ3nlJ5aPPBUW+Bcg2GCfdHK
YC+MdgArZcHOsAFpd/TawTMX0NgQ+bqmWFL72ja8WFDfOjeapg4wyMJYtIMDpuuT
kre36MxScIGfl0cpbmsn+x7C5n4hc3//KiDqvGfHpt9lE7jOInWo8OPkZEcyvQqd
cstVBfzUPDWrxsgv9HF2aMsLmvkyJOMuy30RYO4JtVIjKGalXYbR/71OQcqg23zJ
WvTLULWROm+hyax+/IMpp3WdHDjDeRMkDKkMunvxguhXY6U7VQjteA==
-----END RSA PRIVATE KEY-----
]]></private_key>

<public_key>0x03010001D0CD6145BC1DAB7A63B1D21E85F28BE49F9056C20F73081
04C67E8E4090EE4C1DD6EA9E8EF623C2BA3D9E33150D4C3D8EC05B8DB8FF6ECA69325
59D6521848B29CC8C911F80072DD27AC18E614A9FC90B03B22B62CFDC257CBB09D7EF
E435FCC3AAF1D60C31F7A6A453A6753DEA7F2E50E2FBFB79C0F2C3B0301B411CB67FC
25</public_key>
      </interface>
      <connections number="1">
        <connection name="saratoga-nimitz">
          <remote_ip>193.72.201.131</remote_ip>
          <remote_name>nimitz</remote_name>
          <remote_subject>193.72.201.131</remote_subject>
          <protected_network>
            <ip>193.72.2.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <protected_network>
            <ip>193.72.3.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <preshared_key>n/a</preshared_key>
        </connection>
      </connections>
    </meshing>
    <meshing name="Point-2" type="POINT">
      <ipsec_type>certificate</ipsec_type>
      <central_node>n/a</central_node>
      <ipsec_mode>transport</ipsec_mode>
      <ike_lifetime>3600</ike_lifetime>
      <pfs>yes</pfs>
      <interface ip="193.72.201.128" name="eth0">
        <subject_alt_name>bertomar@zhwin.ch</subject_alt_name>
        <ca><![CDATA[-----BEGIN CERTIFICATE-----
MIIFTjCCBDagAwIBAgIBADANBgkqhkiG9w0BAQQFADBzMQswCQYDVQQGEwJDSDEP
MA0GA1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxME
dGVzdDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1p
```

```
bi5jaDAeFw0wMTEwMjIxNjE0NTNaFw0wMjAxMzAxNjE0NTNaMHMxCzAJBgNVBAYT
AkNIMQ8wDQYDVQQHEwZ6dXJpY2gxFDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYD
VQQLEwR0ZXN0MREwDwYDVQQDEwhWcG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBu
QGFkbWluLmNoMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwwLYiURD
B1xLZUsIjRy9e6ia617qz1hnXwGkuVYQev8ahWOUJEqZR0bjNdm6LPqItZ9sclXj
NS/pNtirZqubGgj6rQrm5gYodizQSvK03oeiuLiBsxefqUpqaHjLBPfiNWxnZxi7
5YTxP0q3lj1D5s3rt60bP1CZmAYwNgANUT0OmaUk4hVUcPraCzPJ/CdlDc9KP+qZ
1iTXea+/zyuDGxalxJtvRq6xlgx4O79R7BWLKb7mg2Ty2fqwjROfPZ7+0Sf3vY2n
BiX6BDlXG8VF7cXUY29jHUxbFAhH8pGIspA6Sc24SzTuvgt7erueXayRcF7AdqdB
T9usiDaoRK/AbQIDAQABo4IB6zCCAecwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUgq5/01Dht0FWizCpdJHOv1n1Mw4wgZ0GA1UdIwSBlTCBkoAUgq5/01Dht0FW
izCpdJHOv1n1Mw6hd6R1MHMxCzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZ6dXJpY2gx
FDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYDVQQLEwR0ZXN0MREwDwYDVQQDEwhW
cG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBuQGFkbWluLmNoggEAMAsGA1UdDwQE
AwIBBjARBglghkgBhvhCAQEEBAMCAAcwFwYDVR0RBBAwDoEMdnBuQGFkbWluLmNo
MBcGA1UdEgQQMA6BDHZwbkBhZG1pbi5jaDAvBgNVHR8EKDAmMCSgIqAghh5odHRw
Oi8vd3d3Lm9wZW4uY2gvY2Evb3Blbi5jcmwwJgYJYIZIAYb4QgECBBkWF2h0dHBz
Oi8vd3d3Lm9wZW4uY2gvY2EvMDAGCWCGSAGG+EIBCAQjFiFodHRwOi8vd3d3Lm9w
ZW4uY2gvY2EvcG9saWN5Lmh0bWwwOAYJYIZIAYb4QgENBCsWKVRoaXMgY2VydGlm
aWNhdGUgaXMgYSBSb290IENBIENlcnRpZmljYXRlMA0GCSqGSIb3DQEBBAUAA4IB
AQCoVrifwabh3NvbL1fck5gfPcbgb+C4LBMacMOHyD94WY1aqatb4yO+nHy0r3jt
ZFSW8btDhsJ+xdvth5ABjlhbuBF32kN0OLi4kbC85H8zYo2OCm+oGzGe+Zqny2JF
/wbL2+oc3rMhacR9AtJxPDKrRAmZrMBOxwvn/syctXF9DKw4nCVA8B40nYQLyjV5
bH1jK5e1WrFAJHjOJ9ZPU+bcIHr7XD6UTJewI5ILBxMjjNtSFUe4QChKFapdkGmt
saW65CERm0ZMGSYRLMNT65MkJeLUkvCBqfsKc/OFokyEE3dsSKEqHd5+ZwM/dnvZ
BprS0bGcCgRVDYWHVEbxelIR
-----END CERTIFICATE-----
]]></ca>
        <crl><![CDATA[-----BEGIN X509 CRL-----
MIIBuTCBojANBgkqhkiG9w0BAQQFADBzMQswCQYDVQQGEwJDSDEPMA0GA1UEBxMG
enVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxMEdGVzdDERMA8G
A1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1pbi5jaBcNMDEx
MDI5MDAzNjQ5WhcNMDExMTI4MDAzNjQ5WjANBgkqhkiG9w0BAQQFAAOCAQEAE5iZ
UFHUc2IzRQqmyY/dMcUyySnWkyI59GsIpUI14A3wYnDpqxC0k675fSrpxkYBSc6H
v2MB8g5qPZNu8+Ses2x3obxwfAzgrOr8XcLEmn+jQETqsTQiUR+ERzNdz5NJBQJW
wQ4Zik2aBA/cBr1OASgeaBfKqg0lbxVMKTtNqnCoMRR2NPHY1lolN29SDKnTAihi
l3l2yLtYOSyDcgD51HTXW9c0/Gxkig+R+ZAgh3D/s42Zm/+G9CBKun+sPXy3nZOS
hnYuvsLTBrrnu+a3GIkd52bNn3XvqAdRvhxJqRmY+ZiNim6nlJpIDr8Hk5KzRAhR
CT01oXvQyAxbw3MAiw==
-----END X509 CRL-----
]]></crl>
        <certificate><![CDATA[-----BEGIN CERTIFICATE-----
MIIE7jCCA9agAwIBAgIBATANBgkqhkiG9w0BAQUFADBzMQswCQYDVQQGEwJDSDEP
MA0GA1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxME
dGVzdDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1p
bi5jaDAeFw0wMTEwMjIxNjE2MDJaFw0wMjAxMzAxNjE2MDJaMHoxCzAJBgNVBAYT
AkNIMRMwEQYDVQQHEwp3aW50ZXJ0aHVyMQ4wDAYDVQQKEwV6aHdpbjELMAkGA1UE
CxMCaXQxFzAVBgNVBAMTDk1hcmNvIEJlcnRvc3NhMSAwHgYJKoZIhvcNAQkBFhFi
ZXJ0b21hckB6aHdpbi5jaDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAywKK
ijx7ttKOJu5KCa3FlGPCEuZ2BPeMdbIxqBsny/uC2mdqmJpakyARm5VrOBZPlaqB
fvJwTT/5U0WJf/r0XX+Fy6LaRmNYCMOsd4kkmWgOGi+B4rnFtR5a1nbS5LRJI48j
/e8mvYIovgKyvdMhDbeeAxJxXJcJZO2Q/MyoEjkCAwEAAaOCAggwggIEMAsGA1Ud
DwQEAwIDqDAdBgNVHQ4EFgQU3EWIZL7qWpKywBFTRr3hSq6+Y7gwgZ0GA1UdIwSB
lTCBkoAUgq5/01Dht0FWizCpdJHOv1n1Mw6hd6R1MHMxCzAJBgNVBAYTAkNIMQ8w
DQYDVQQHEwZ6dXJpY2gxFDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYDVQQLEwR0
ZXN0MREwDwYDVQQDEwhWcG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBuQGFkbWlu
LmNoggEAMBcGA1UdEgQQMA6BDHZwbkBhZG1pbi5jaDAvBgNVHR8EKDAmMCSgIqAg
hh5odHRwOi8vd3d3Lm9wZW4uY2gvY2Evb3Blbi5jcmwwEQYJYIZIAYb4QgEBBAQD
AgSwMCYGCWCGSAGG+EIBAgQZFhdodHRwczovL3d3dy5vcGVuLmNoL2NhLzAwBglg
hkgBhvhCAQgEIxYhaHR0cDovL3d3dy5vcGVuLmNoL2NhL3BvbGljeS5odG1sMD0G
CWCGSAGG+EIBDQQwFi5UaGlzIGNlcnRpZmljYXRlIHdhcyBpc3N1ZWQgYnkgT3Bl
biBTeXN0ZW1zIEFHMCIGCWCGSAGG+EIBAwQVFhNjZ2kvbm9uLUNBLXJldi5jZ2k/
MBwGA1UdEQQVMBOBEWJlcnRvbWFyQHpod2luLmNoMA0GCSqGSIb3DQEBBQUAA4IB
AQAr1CWKdAT4us+ufAtN5RIrlYWE6XU1418Mji7lcffITG+SWPDmrhEuvuQmI8IK
C9MuKIfc+FvS4QMtH3rQdtBNqKfldA1Bt+txLyn3YwKMY3w4nH5fB6RRRfqcQtJ8
tbI6A5XcqO52zePkkNzqyO1SJ2fYREgRt28MYWJFMFTLAqdYyjcE69NPUayuz4Mm
bWdnOMUOyrHn3M3RW0YAKBl1fENR/TpgDKdSl3/zsnWKd1EToKgODUCR5YC7QL8P
U7AIdUj5F179mwazroSqvYk+K4YqSDhNl/WoB6hGyuIuwnDZY8nGLv6dr8J3aylz
```

```
9mXtVZATG9giO3U9Yb0Ji715
-----END CERTIFICATE-----
]]></certificate>
        <passphrase>I+P0Sp2x5Qwu1Oz1csC</passphrase>
        <private_key><![CDATA[-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,F7BFB8F40DA8959C

GM3x0rcxke/5weBIbetKaPKZttB87F5LlfgN7p3+Jxy7e+cQtHZo/xvcTwYJavfH
u4qHaG51aATVncb3gh/H78SxMdBWpWYiBSlWhZRlNA7WGqo35RGaH8YSISnDvGOH
G84MJUEZITH3jiB9L37TbMkJSDjYVkAcl8VvUNz29Ql4ibvo4FG14zimOp4UVP1e
TWDakJmLz+0VOsMzzqqeRkbJ/X7xr2VlvA0tKY7zuiK/W53q7UAA31IiKHTR774i
m3XpnWqGWd39LLeO8EuUGNJRmQRSSBcwbzHmR39UUSrBiEtwwezJft/rUBm2vyau
AUmqqNS21FYPah0o1d3aa41AoJHw7RYJPWes0z54CJeSfhodVU5Iyntrb2pvNrHq
k+Kux0dGj43nxw2M9sMYDAIYxv5DIR7FJaw9f9U0W/QX6R7/GkX2n3cNAjNRFzCw
5ZepP0KJ+PEUphLPbt5UYxD/GGcAZX3lQPe/GdzpvaoaOiN8d8FyXFu2QxWKHAG0
2KlzEeEcQJxv887n72L9JTRSukSf8OOiKp4b4p/sv4dEdlVnhy9OjgBXBJOnFC2k
2VRHf5pnUn1gZ/MXcFz0UsI2XW14mtjNDE/Or6GQ8vHGOtzZcAd1sLMkHBvb0kfK
yXLubWe4T2GBYNbV0Anf146WrB2HOy/J+soBw57vlATiTx74B8KJT6BKHgxANkR6
R0BI7f49imAFtYMgWATgAkFUVAnq9O0GNsoHnkey3fBlLkf0/l9tQKwdz/2Guor2
/H3P/MjWcUEes1k396PFFgAW0PO2ST0KEZu4qsmNwJYznjsp6JpRIA==
-----END RSA PRIVATE KEY-----
]]></private_key>

<public_key>0x03010001CB028A8A3C7BB6D28E26EE4A09ADC59463C212E67604F78
C75B231A81B27CBFB82DA676A989A5A9320119B956B38164F2DAA817EF2704D3FF953
45897FFAF45D7F85CBA2DA46635808C3AC77892499680E1A2F81E2B9C5B51E5AD676D
2E4B449238F23FDEF26BD8228BE02B2BDD3210DB79E0312715C970964ED90FCCCA812
39</public_key>
      </interface>
      <connections number="1">
        <connection name="saratoga-lionel">
          <remote_ip>193.72.201.136</remote_ip>
          <remote_name>lionel</remote_name>
          <remote_subject>lionel@open.ch</remote_subject>
          <protected_network>
            <ip>193.72.4.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <preshared_key>n/a</preshared_key>
        </connection>
      </connections>
    </meshing>
    <meshing name="Hub-Mesh" type="HUB">
      <ipsec_type>certificate</ipsec_type>
      <central_node>193.72.201.136</central_node>
      <ipsec_mode>tunnel</ipsec_mode>
      <ike_lifetime>3600</ike_lifetime>
      <pfs>no</pfs>
      <interface ip="193.72.201.128" name="eth0">
        <subject_alt_name>bertomar@zhwin.ch</subject_alt_name>
        <ca><![CDATA[-----BEGIN CERTIFICATE-----
MIIFTjCCBDagAwIBAgIBADANBgkqhkiG9w0BAQQFADBzMQswCQYDVQQGEwJDSDEP
MA0GA1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxME
dGVzdDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1p
bi5jaDAeFw0wMTEwMjIxNjE0NTNaFw0wMjAxMzAxNjE0NTNaMHMxCzAJBgNVBAYT
AkNIMQ8wDQYDVQQHEwZ6dXJpY2gxFDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYD
VQQLEwR0ZXN0MREwDwYDVQQDEwhWcG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBu
QGFkbWluLmNoMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwwLYiURD
B1xLZUsIjRy9e6ia617qz1hnXwGkuVYQev8ahWOUJEqZR0bjNdm6LPqItZ9sclXj
NS/pNtirZqubGgj6rQrm5gYodizQSvK03oeiuLiBsxefqUpqaHjLBPfiNWxnZxi7
5YTxP0q3lj1D5s3rt60bP1CZmAYwNgANUT0OmaUk4hVUcPraCzPJ/CdlDc9KP+qZ
1iTXea+/zyuDGxalxJtvRq6xlgx4O79R7BWLKb7mg2Ty2fqwjROfPZ7+0Sf3vY2n
BiX6BDlXG8VF7cXUY29jHUxbFAhH8pGIspA6Sc24SzTuvgt7erueXayRcF7AdqdB
T9usiDaoRK/AbQIDAQABo4IB6zCCAecwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUgq5/01Dht0FWizCpdJHOv1n1Mw4wgZ0GA1UdIwSBlTCBkoAUgq5/01Dht0FW
izCpdJHOv1n1Mw6hd6R1MHMxCzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZ6dXJpY2gx
FDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYDVQQLEwR0ZXN0MREwDwYDVQQDEwhW
```

```
cG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBuQGFkbWluLmNoggEAMAsGA1UdDwQE
AwIBBjARBglghkgBhvhCAQEEBAMCAAcwFwYDVR0RBBAwDoEMdnBuQGFkbWluLmNo
MBcGA1UdEgQQMA6BDHZwbkBhZG1pbi5jaDAvBgNVHR8EKDAmMCSgIqAghh5odHRw
Oi8vd3d3Lm9wZW4uY2gvY2Evb3Blbi5jcmwwJgYJYIZIAYb4QgECBBkWF2h0dHBz
Oi8vd3d3Lm9wZW4uY2gvY2EvMDAGCWCGSAGG+EIBCAQjFiFodHRwOi8vd3d3Lm9w
ZW4uY2gvY2EvcG9saWN5Lmh0bWwwOAYJYIZIAYb4QgENBCsWKVRoaXMgY2VydGlm
aWNhdGUgaXMgYSBSb290IENBIENlcnRpZmljYXRlMA0GCSqGSIb3DQEBBAUAA4IB
AQCoVrifwabh3NvbL1fck5gfPcbgb+C4LBMacMOHyD94WY1aqatb4yO+nHy0r3jt
ZFSW8btDhsJ+xdvth5ABjlhbuBF32kN0OLi4kbC85H8zYo2OCm+oGzGe+Zqny2JF
/wbL2+oc3rMhacR9AtJxPDKrRAmZrMBOxwvn/syctXF9DKw4nCVA8B40nYQLyjV5
bH1jK5e1WrFAJHjOJ9ZPU+bcIHr7XD6UTJewI5ILBxMjjNtSFUe4QChKFapdkGmt
saW65CERm0ZMGSYRLMNT65MkJeLUkvCBqfsKc/OFokyEE3dsSKEqHd5+ZwM/dnvZ
BprS0bGcCgRVDYWHVEbxelIR
-----END CERTIFICATE-----
]]></ca>
        <crl><![CDATA[-----BEGIN X509 CRL-----
MIIBuTCBojANBgkqhkiG9w0BAQQFADBzMQswCQYDVQQGEwJDSDEPMA0GA1UEBxMG
enVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxMEdGVzdDERMA8G
A1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1pbi5jaBcNMDEx
MDI5MDAzNjQ5WhcNMDExMTI4MDAzNjQ5WjANBgkqhkiG9w0BAQQFAAOCAQEAE5iZ
UFHUc2IzRQqmyY/dMcUyySnWkyI59GsIpUI14A3wYnDpqxC0k675fSrpxkYBSc6H
v2MB8g5qPZNu8+Ses2x3obxwfAzgrOr8XcLEmn+jQETqsTQiUR+ERzNdz5NJBQJW
wQ4Zik2aBA/cBr1OASgeaBfKqg0lbxVMKTtNqnCoMRR2NPHY1lolN29SDKnTAihi
l3l2yLtYOSyDcgD51HTXW9c0/Gxkig+R+ZAgh3D/s42Zm/+G9CBKun+sPXy3nZOS
hnYuvsLTBrrnu+a3GIkd52bNn3XvqAdRvhxJqRmY+ZiNim6nlJpIDr8Hk5KzRAhR
CT01oXvQyAxbw3MAiw==
-----END X509 CRL-----
]]></crl>
        <certificate><![CDATA[-----BEGIN CERTIFICATE-----
MIIE7jCCA9agAwIBAgIBATANBgkqhkiG9w0BAQUFADBzMQswCQYDVQQGEwJDSDEP
MA0GA1UEBxMGenVyaWNoMRQwEgYDVQQKEwtkZXZlbG9wbWVudDENMAsGA1UECxME
dGVzdDERMA8GA1UEAxMIVnBuYWRtaW4xGzAZBgkqhkiG9w0BCQEWDHZwbkBhZG1p
bi5jaDAeFw0wMTEwMjIxNjE2MDJaFw0wMjAxMzAxNjE2MDJaMHoxCzAJBgNVBAYT
AkNIMRMwEQYDVQQHEwp3aW50ZXJ0aHVyMQ4wDAYDVQQKEwV6aHdpbjELMAkGA1UE
CxMCaXQxFzAVBgNVBAMTDk1hcmNvIEJlcnRvc3NhMSAwHgYJKoZIhvcNAQkBFhFi
ZXJ0b21hckB6aHdpbi5jaDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAywKK
ijx7ttKOJu5KCa3FlGPCEuZ2BPeMdbIxqBsny/uC2mdqmJpakyARm5VrOBZPLaqB
fvJwTT/5U0WJf/r0XX+Fy6LaRmNYCMOsd4kkmWgOGi+B4rnFtR5a1nbS5LRJI48j
/e8mvYIovgKyvdMhDbeeAxJxXJcJZO2Q/MyoEjkCAwEAAaOCAggwggIEMAsGA1Ud
DwQEAwIDqDAdBgNVHQ4EFgQU3EWIZL7qWpKywBFTRr3hSq6+Y7gwgZ0GA1UdIwSB
lTCBkoAUgq5/01Dht0FWizCpdJHOv1n1Mw6hd6R1MHMxCzAJBgNVBAYTAkNIMQ8w
DQYDVQQHEwZ6dXJpY2gxFDASBgNVBAoTC2RldmVsb3BtZW50MQ0wCwYDVQQLEwR0
ZXN0MREwDwYDVQQDEwhWcG5hZG1pbjEbMBkGCSqGSIb3DQEJARYMdnBuQGFkbWlu
LmNoggEAMBcGA1UdEgQQMA6BDHZwbkBhZG1pbi5jaDAvBgNVHR8EKDAmMCSgIqAg
hh5odHRwOi8vd3d3Lm9wZW4uY2gvY2Evb3Blbi5jcmwwEQYJYIZIAYb4QgEBBAQD
AgSwMCYGCWCGSAGG+EIBAgQZFhdodHRwczovL3d3dy5vcGVuLmNoL2NhLzAwBglg
hkgBhvhCAQgEIxYhaHR0cDovL3d3dy5vcGVuLmNoL2NhL3BvbGljeS5odG1sMD0G
CWCGSAGG+EIBDQQwFi5UaGlzIGNlcnRpZmljYXRlIHdhcyBpc3N1ZWQgYnkgT3Bl
biBTeXN0ZW1zIEFHMCIGCWCGSAGG+EIBAwQVFhNjZ2kvbm9uLUNBLXJldi5jZ2k/
MBwGA1UdEQQVMBOBEWJlcnRvbWFyQHpod2luLmNoMA0GCSqGSIb3DQEBBQUAA4IB
AQAr1CWKdAT4us+ufAtN5RIrlYWE6XU1418Mji7lcffITG+SWPDmrhEuvuQmI8IK
C9MuKIfc+FvS4QMtH3rQdtBNqKfldA1Bt+txLyn3YwKMY3w4nH5fB6RRRfqcQtJ8
tbI6A5XcqO52zePkkNzqyO1SJ2fYREgRt28MYWJFMFTLAqdYyjcE69NPUayuz4Mm
bWdnOMUOyrHn3M3RW0YAKBl1fENR/TpgDKdSl3/zsnWKd1ETOKgODUCR5YC7QL8P
U7AIdUj5F179mwazroSqvYk+K4YqSDhNl/WoB6hGyuIuwnDZY8nGLv6dr8J3aylz
9mXtVZATG9giO3U9Yb0Ji715
-----END CERTIFICATE-----
]]></certificate>
        <passphrase>I+P0Sp2x5Qwu1Oz1csC</passphrase>
        <private_key><![CDATA[-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,F7BFB8F40DA8959C

GM3x0rcxke/5weBIbetKaPKZttB87F5LlfgN7p3+Jxy7e+cQtHZo/xvcTwYJavfH
u4qHaG51aATVncb3gh/H78SxMdBWpWYiBSlWhZRlNA7WGqo35RGaH8YSISnDvGOH
G84MJUEZITH3jiB9L37TbMkJSDjYVkAcl8VvUNz29Ql4ibvo4FG14zimOp4UVP1e
TWDakJmLz+0VOsMzzqqeRkbJ/X7xr2VlvA0tKY7zuiK/W53q7UAA31IiKHTR774i
m3XpnWqGWd39LLeO8EuUGNJRmQRSSBcwbzHmR39UUSrBiEtwwezJft/rUBm2vyau
```

```
AUmqqNS21FYPah0o1d3aa41AoJHw7RYJPWes0z54CJeSfhodVU5Iyntrb2pvNrHq
k+Kux0dGj43nxw2M9sMYDAIYxv5DIR7FJaw9f9U0W/QX6R7/GkX2n3cNAjNRFzCw
5ZepP0KJ+PEUphLPbt5UYxD/GGcAZX3lQPe/GdzpvaoaOiN8d8FyXFu2QxWKHAG0
2KlzEeEcQJxv887n72L9JTRSukSf8OOiKp4b4p/sv4dEdlVnhy9OjgBXBJOnFC2k
2VRHf5pnUn1gZ/MXcFz0UsI2XW14mtjNDE/Or6GQ8vHGOtzZcAd1sLMkHBvb0kfK
yXLubWe4T2GBYNbV0Anf146WrB2HOy/J+soBw57vlATiTx74B8KJT6BKHgxANkR6
R0BI7f49imAFtYMgWATgAkFUVAnq9O0GNsoHnkey3fBlLkf0/l9tQKwdz/2Guor2
/H3P/MjWcUEes1k396PFFgAW0PO2ST0KEZu4qsmNwJYznjsp6JpRIA==
-----END RSA PRIVATE KEY-----
]]></private_key>

<public_key>0x03010001CB028A8A3C7BB6D28E26EE4A09ADC59463C212E67604F78
C75B231A81B27CBFB82DA676A989A5A9320119B956B38164F2DAA817EF2704D3FF953
45897FFAF45D7F85CBA2DA46635808C3AC77892499680E1A2F81E2B9C5B51E5AD676D
2E4B449238F23FDEF26BD8228BE02B2BDD3210DB79E0312715C970964ED90FCCCA812
39</public_key>
      </interface>
      <connections number="1">
        <connection name="saratoga-lionel">
          <remote_ip>193.72.201.136</remote_ip>
          <remote_name>lionel</remote_name>
          <remote_subject>lionel@open.ch</remote_subject>
          <protected_network>
            <ip>193.72.4.0</ip>
            <netmask>24</netmask>
          </protected_network>
          <preshared_key>n/a</preshared_key>
        </connection>
      </connections>
    </meshing>
  </node>
```

15.4 ipsec.conf

```
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file
#
# Hostname: saratoga
#
# created Saturday 27.Oct 2001 15:15:46
#

# basic configuration
config setup
      # THIS SETTING MUST BE CORRECT or almost nothing will work;
      # %defaultroute is okay for most simple cases.
      interfaces="ipsec0=eth0"
      # Debug-logging controls:  "none" for (almost) none, "all" for
lots.
      klipsdebug=none
      plutodebug=none
      # Use auto= parameters in conn descriptions to control startup
actions.
      plutoload=%search
      plutostart=%search
      # Close down old connection when new one using same ID shows
up.
      #uniqueids=yes

# defaults for all the connections
conn %default
      #keytries=3

# Meshing: Full_Mesh
conn saratoga-nimitz-0-0
      type=tunnel
      authby=secret
      # Host
      ikelifetime=3600
      pfs=yes
      left=193.72.201.128
      leftsubnet=193.72.1.0/2
      # Remote
      right=193.72.201.131
      rightsubnet=193.72.2.0/24
      auto=start

conn saratoga-nimitz-0-1
      type=tunnel
      authby=secret
      # Host
      ikelifetime=3600
      pfs=yes
      left=193.72.201.128
      leftsubnet=193.72.1.0/2
      # Remote
      right=193.72.201.131
      rightsubnet=193.72.3.0/24
      auto=start

conn saratoga-lionel-0-0
      type=tunnel
      authby=secret
      # Host
      ikelifetime=3600
      pfs=yes
      left=193.72.201.128
      leftsubnet=193.72.1.0/2
      # Remote
      right=193.72.201.136
      rightsubnet=193.72.4.0/24
```

```
      auto=start

# Meshing: Point-1
conn saratoga-nimitz
      type=transport
      authby=rsasig
      # Host
      leftrsasigkey=%cert
      ikelifetime=3600
      pfs=yes
      left=193.72.201.128
      leftcert=Point-1.pem
      # Remote
      rightrsasigkey=%cert
      rightcert=Point-1.pem
      right=193.72.201.131
      auto=start

# Meshing: Point-2
conn saratoga-lionel
      type=transport
      authby=rsasig
      # Host
      leftrsasigkey=%cert
      ikelifetime=3600
      pfs=yes
      left=193.72.201.128
      leftcert=Point-2.pem
      # Remote
      rightrsasigkey=%cert
      rightcert=Point-2.pem
      right=193.72.201.136
      auto=start

# Meshing: Hub-Mesh
conn saratoga-lionel-0-0
      type=tunnel
      authby=rsasig
      # Host
      leftrsasigkey=%cert
      ikelifetime=3600
      pfs=no
      left=193.72.201.128
      leftcert=Hub-Mesh.pem
      leftsubnet=193.72.1.0/2
      # Remote
      rightrsasigkey=%cert
      rightcert=Hub-Mesh.pem
      right=193.72.201.136
      rightsubnet=193.72.4.0/24
      auto=start
```

15.5 ipsec.secrets

```
# /etc/ipsec.secrets - FreeS/WAN
#
# Hostname: saratoga
#
# created Saturday 27.Oct 2001 15:15:46
#
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication.  See ipsec_pluto(8) manpage, and HTML
documentation.

# Shared secret (an arbitrary character string, which should be both
long
# and hard to guess, enclosed in quotes) for a pair of negotiating
hosts.
# Must be same on both; generate on one and copy to the other.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.  Put ONLY the "pubkey" part into
connection
# descriptions on the other host(s); it need not be kept secret, and
can
# be extracted conveniently with "ipsec showhostkey".

# Meshing: Full_Mesh
# Connection: "saratoga-nimitz"
193.72.201.128 193.72.201.131: PSK "p1c/C+jVlSpJP6mHYVAH"
# Connection: "saratoga-lionel"
193.72.201.128 193.72.201.136: PSK "AmPHKgu8XfY5z0YjtbLW"

# Meshing: Point-1
193.72.201.128: RSA {
      Modulus:
0xD0CD6145BC1DAB7A63B1D21E85F28BE49F9056C20F7308104C67E8E4090EE4C1DD6
EA9E8EF623C2BA3D9E33150D4C3D8EC05B8DB8FF6ECA6932559D6521848B29CC8C911
F80072DD27AC18E614A9FC90B03B22B62CFDC257CBB09D7EFE435FCC3AAF1D60C31F7
A6A453A6753DEA7F2E50E2FBFB79C0F2C3B0301B411CB67FC25
      PublicExponent:  0x010001
      PrivateExponent:
0x2A6CD08ADEC2188971A03BD53FDA1C5A83A14C3F79EC21834E42FEAAF6BC33F6C00
B3032C810E9B0E445ED47A3E8D73248229180EB2B7CC20BBBD71918ED1E37EE6B31C8
1E9B69449C1A1205700CD78E8B0F2E858367F302CCE7ABAE96CE10954D563D43B6793
41472B40D42501A62B9CEB0D9A724B98BF240FDB876A7148501
      Prime1:
0xEEA659FD49E138FBFEF2CC666DF596242EC39D0F184BBEC046CFC17772867B67430
F2782015DBCC041201FBF46DE4E88207AF9AFF2EAEFDEC30C20DBBF572D11
      Prime2:
0xDFFB84C79E7182EF20E7C85129EDDA17DD8F54A76DA3F8C5E42FD310CD8D39D7AA4
5B4EDD59452204FC33AC50382D6F59FE8C056E61C25FC7DC98F6F93BBADD5
      Exponent1:
0xC281267AFFC9E68DCBFECB84F83DDEFEE2765FFDBB89909059A65E42223F6538863
945B3F9B5126F8724CD7B322161D424D4D5C807AE5F8E295E2B31AEC33861
      Exponent2:
0x553B96348C43AFCAA59FEE278CA819651987D0740211F2BA03727B841A64528D921
B0295BABFBD5CD45D7B80BB2C4FF69A4A6CDEE4A3F38498AD59EAE0B1FB9D
      Coefficient:
0x15C3F74F2CCBA2E2CC29B8C34BC2E42BE5591B46AA39A72F3E65CB17836649BD97D
79507C07E0ADE60A4A822AA3E3C0AE125EDF551F21CDBCB6A965698B75BF3
  }

# Meshing: Point-2
bertomar@zhwin.ch: RSA {
      Modulus:
0xCB028A8A3C7BB6D28E26EE4A09ADC59463C212E67604F78C75B231A81B27CBFB82D
A676A989A5A9320119B956B38164F2DAA817EF2704D3FF95345897FFAF45D7F85CBA2
```

```
DA46635808C3AC77892499680E1A2F81E2B9C5B51E5AD676D2E4B449238F23FDEF26B
D8228BE02B2BDD3210DB79E0312715C970964ED90FCCCA81239
      PublicExponent:  0x010001
      PrivateExponent:
0xB3DBB1805DABC1C2F3FC5BDBAE8D8D6235ABC8F0F398DCF54D852D67617FB5C2442
159C46025489FD6A6A2F013892282CDD3A55D22500AB0E62537936596A20DEA092B6C
08682D5A74E815258C33E20C6B7A9E8CD4E36007EDDD49D26A291353DB548A4FE259F
EC69F086D805F6F266CF46189B1377B840941B9645C762A005D
      Prime1:
0xFCACB0A1181461497486DE599FF65F2710BB8638637FC15164AB6EA460D00C6846F
F65FC7628FBBF017C01F3095B70171EA00B9DAC133812A965640B6BFBFCEB
      Prime2:
0xCDAE8574FBF7316706E54CA5E7B15E2C353005CFFDE0F6E58B64E2CDC0079B7D46E
324FE6A0CB62F9B5C231BEDA9CA98C53D182C09F8A3CA132632C36EF9146B
      Exponent1:
0xDE9D17111731FD74C6217D0CE18E22CF3C42BB423B0777BECA3EEFC00B3BC172ADF
A9E540B969815F4ACD8DAD347A83DF651736A7B0C83BE8D5833088D1EF9F9
      Exponent2:
0x96E367087127E0D82218E967F4570C2D063FDAA3C12DDE327B5661846B51A961F83
4FE989D9801B75F13F2B5B45E91400DC24A0F54A9510F17893848630ADB4D
      Coefficient:
0x1800CA23A1E5E9392453D473FDCBA8E51C75BDBF33E82FB1EAEC39B380710FA3AF7
9AD96C45E3645D7FBF0130F9940976F909C9AB0EA82FB7A94D5499341F62D
  }

# Meshing: Hub-Mesh
bertomar@zhwin.ch: RSA {
      Modulus:
0xCB028A8A3C7BB6D28E26EE4A09ADC59463C212E67604F78C75B231A81B27CBFB82D
A676A989A5A9320119B956B38164F2DAA817EF2704D3FF95345897FFAF45D7F85CBA2
DA46635808C3AC77892499680E1A2F81E2B9C5B51E5AD676D2E4B449238F23FDEF26B
D8228BE02B2BDD3210DB79E0312715C970964ED90FCCCA81239
      PublicExponent:  0x010001
      PrivateExponent:
0xB3DBB1805DABC1C2F3FC5BDBAE8D8D6235ABC8F0F398DCF54D852D67617FB5C2442
159C46025489FD6A6A2F013892282CDD3A55D22500AB0E62537936596A20DEA092B6C
08682D5A74E815258C33E20C6B7A9E8CD4E36007EDDD49D26A291353DB548A4FE259F
EC69F086D805F6F266CF46189B1377B840941B9645C762A005D
      Prime1:
0xFCACB0A1181461497486DE599FF65F2710BB8638637FC15164AB6EA460D00C6846F
F65FC7628FBBF017C01F3095B70171EA00B9DAC133812A965640B6BFBFCEB
      Prime2:
0xCDAE8574FBF7316706E54CA5E7B15E2C353005CFFDE0F6E58B64E2CDC0079B7D46E
324FE6A0CB62F9B5C231BEDA9CA98C53D182C09F8A3CA132632C36EF9146B
      Exponent1:
0xDE9D17111731FD74C6217D0CE18E22CF3C42BB423B0777BECA3EEFC00B3BC172ADF
A9E540B969815F4ACD8DAD347A83DF651736A7B0C83BE8D5833088D1EF9F9
      Exponent2:
0x96E367087127E0D82218E967F4570C2D063FDAA3C12DDE327B5661846B51A961F83
4FE989D9801B75F13F2B5B45E91400DC24A0F54A9510F17893848630ADB4D
      Coefficient:
0x1800CA23A1E5E9392453D473FDCBA8E51C75BDBF33E82FB1EAEC39B380710FA3AF7
9AD96C45E3645D7FBF0130F9940976F909C9AB0EA82FB7A94D5499341F62D}
```

15.6 Oracle sql-Installationsskript

```
/* CHANGELOG

      September 20, 2001 lionel@open.ch
      - Created the schema "vpn"

      October 3, 2001 ok@open.ch
      - Changed some precision things based on user requests
      - added further tables from other script

      October 11, 2001 ok@open.ch
      - changed some attributes based on user requests
```

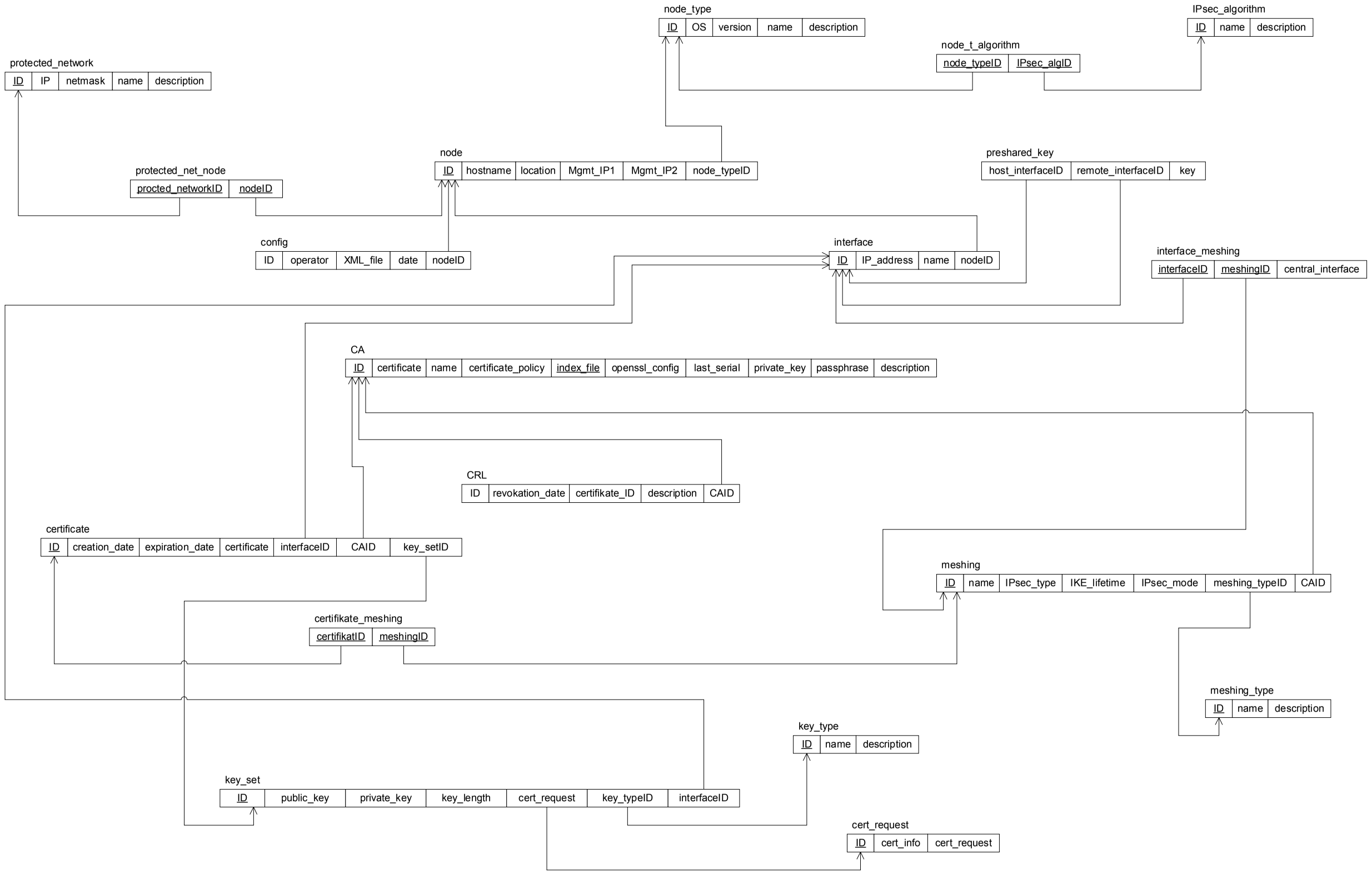
```
            October 12, 2001 ok@open.ch
      - changed some attributes based on user requests

      October 17, 2001 ok@open.ch
      - changed FK relation on meshing to accept NULL (feature??)

      October 18, 2001 ok@open.ch
      - added table preshared_key

*/

/* User check */
execute os_util.verify_user('vpn');


/* Create the sequences */
CREATE SEQUENCE interface_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE node_type_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE IPSEC_alg_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE protected_network_sequence START WITH 1 INCREMENT BY
1 ORDER;
CREATE SEQUENCE node_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE config_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE certificate_sequence START WITH 1 INCREMENT BY 1
ORDER;
CREATE SEQUENCE key_set_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE cert_request_sequence START WITH 1 INCREMENT BY 1
ORDER;
CREATE SEQUENCE key_type_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE crl_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE ca_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE meshing_sequence START WITH 1 INCREMENT BY 1 ORDER;
CREATE SEQUENCE meshing_type_sequence START WITH 1 INCREMENT BY 1
ORDER;



/* Create the tables */
CREATE TABLE interface (
      id NUMBER(11) DEFAULT 0 NOT NULL, /*uses a sequence, be
prepared... */
      ip_address VARCHAR2(15) DEFAULT '' NOT NULL,
      name VARCHAR2(20) DEFAULT'',
      nodeid NUMBER(11) NOT NULL
);

CREATE TABLE node_type (
      id NUMBER(11) DEFAULT 0 NOT NULL, /*uses a sequence, be
prepared... */
      os VARCHAR2(20) DEFAULT '',
      version VARCHAR2(20) DEFAULT'',
      name VARCHAR2(20) DEFAULT'' NOT NULL,
      description VARCHAR2(200) DEFAULT''
);

CREATE TABLE node_t_algorithm (
      node_typeid NUMBER(11) DEFAULT 0 NOT NULL,
      ipsec_algid NUMBER(11) DEFAULT 0 NOT NULL
);

CREATE TABLE ipsec_alg (
      id NUMBER(11) DEFAULT 0 NOT NULL, /*uses a sequence, be
prepared... */
      name VARCHAR2(20) DEFAULT'',
      description VARCHAR2(200) DEFAULT''
);
```

```
CREATE TABLE protected_network (
      id NUMBER(11) DEFAULT 0 NOT NULL, /*uses a sequence, be
prepared... */
      ip VARCHAR2(15) DEFAULT '' NOT NULL,
      netmask VARCHAR2(15) DEFAULT '' NOT NULL,
      name VARCHAR2(20) DEFAULT'' NOT NULL,
      description VARCHAR2(200) DEFAULT''
);

CREATE TABLE protected_net_node (
      nodeid NUMBER(11) DEFAULT 0 NOT NULL,
      protected_networkid NUMBER(11) DEFAULT 0 NOT NULL
);

CREATE TABLE node (
      id NUMBER(11) DEFAULT 0 NOT NULL, /*uses a sequence, be
prepared... */
      hostname VARCHAR2(20) DEFAULT '' NOT NULL,
      location VARCHAR2(50) DEFAULT '',
      mgmt_ip1 VARCHAR2(15) DEFAULT '' NOT NULL,
      mgmt_ip2 VARCHAR2(15) DEFAULT '',
      node_typeid NUMBER(11) NOT NULL
);


CREATE TABLE config (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       operator VARCHAR2(20) DEFAULT '' NOT NULL,
       configdate DATE DEFAULT TO_DATE('01-01-2001 00:00:00','MM-DD-
YYYY HH24:MI:SS'),
       nodeid NUMBER(11) DEFAULT 0 NOT NULL,
       xml_file CLOB
);

CREATE TABLE certificate (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       certificate CLOB,
       creation_date DATE DEFAULT TO_DATE('01-01-2001 00:00:00','MM-
DD-YYYY HH24:MI:SS'),
       expiration_date DATE DEFAULT TO_DATE('01-01-2001
00:00:00','MM-DD-YYYY HH24:MI:SS'),
       caid NUMBER(11) DEFAULT 0 NOT NULL,
       key_setid NUMBER(11) DEFAULT 0 NOT NULL,
       interfaceid NUMBER(11) DEFAULT 0 NOT NULL
       subject_alt_name VARCHAR(20 DEFAULT '' NULL
);

CREATE TABLE key_set (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       public_key CLOB,
       private_key CLOB,
       key_length NUMBER(5) DEFAULT 0 NOT NULL,
       interfaceid NUMBER(11) DEFAULT 0 NOT NULL,
       cert_requestid NUMBER(11) DEFAULT 0 NOT NULL,
       key_typeid NUMBER(11) DEFAULT 0 NOT NULL,
       passphrase VARCHAR2(20) DEFAULT '' NOT NULL
);

CREATE TABLE cert_request (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       cert_info VARCHAR2(200) DEFAULT '' NOT NULL,
       cert_request CLOB
);

CREATE TABLE key_type (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       name VARCHAR2(20) DEFAULT '' NOT NULL,
       description VARCHAR2(200) DEFAULT ''
```

```
);

CREATE TABLE crl (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       revocation_date DATE DEFAULT TO_DATE('01-01-2001
00:00:00','MM-DD-YYYY HH24:MI:SS'),
       description VARCHAR2(200) DEFAULT '',
       caid NUMBER(11) DEFAULT 0 NOT NULL,
       certificateid NUMBER(11) NOT NULL
);

CREATE TABLE ca (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       name VARCHAR2(20) DEFAULT '' NOT NULL,
       certificate_policy VARCHAR2(200) DEFAULT '',
       description VARCHAR2(200) DEFAULT '',
       last_serial VARCHAR2(20) NOT NULL,
       certificate CLOB,
       private_key CLOB;
       index_file CLOB,
       openssl_config CLOB,
       passphrase VARCHAR2(20) NUT NULL DEFAULT ''
);

CREATE TABLE meshing (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       name VARCHAR2(20) DEFAULT '' NOT NULL,
       ike_lifetime NUMBER(7) DEFAULT 0,
       ipsec_type VARCHAR2(20) DEFAULT '' NOT NULL,
       ipsec_mode VARCHAR2(20) DEFAULT '' NOT NULL,
       caid NUMBER(11),
       meshing_typeid NUMBER(11) DEFAULT 0 NOT NULL
);

CREATE TABLE meshing_type (
       id NUMBER(11) DEFAULT 0 NOT NULL, /* sequence!! */
       name VARCHAR2(20) DEFAULT '' NOT NULL,
       description VARCHAR2(200) DEFAULT ''
);

CREATE TABLE certificate_meshing (
       meshingid NUMBER(11) DEFAULT 0 NOT NULL,
       certificateid NUMBER(11) DEFAULT 0 NOT NULL
);

CREATE TABLE interface_meshing (
       meshingid NUMBER(11) DEFAULT 0 NOT NULL,
       interfaceid NUMBER(11) DEFAULT 0 NOT NULL,
       central_interface NUMBER(2) DEFAULT 0 NOT NULL
);


CREATE TABLE preshared_key (
       host_interfaceid NUMBER(11) NOT NULL,
       remote_interfaceid NUMBER(11) NOT NULL,
       key VARCHAR2(100) NOT NULL
);


/* add primary key constraints */
ALTER TABLE interface ADD CONSTRAINT interface_pk1 PRIMARY KEY (id);
ALTER TABLE node_type ADD CONSTRAINT node_type_pk1 PRIMARY KEY (id);
ALTER TABLE ipsec_alg ADD CONSTRAINT ipsec_alg_pk1 PRIMARY KEY (id);
ALTER TABLE protected_network ADD CONSTRAINT protected_network_pk1
PRIMARY KEY (id);
ALTER TABLE node ADD CONSTRAINT node_pk1 PRIMARY KEY (id);
ALTER TABLE config ADD CONSTRAINT config_pk1 PRIMARY KEY (id);
```

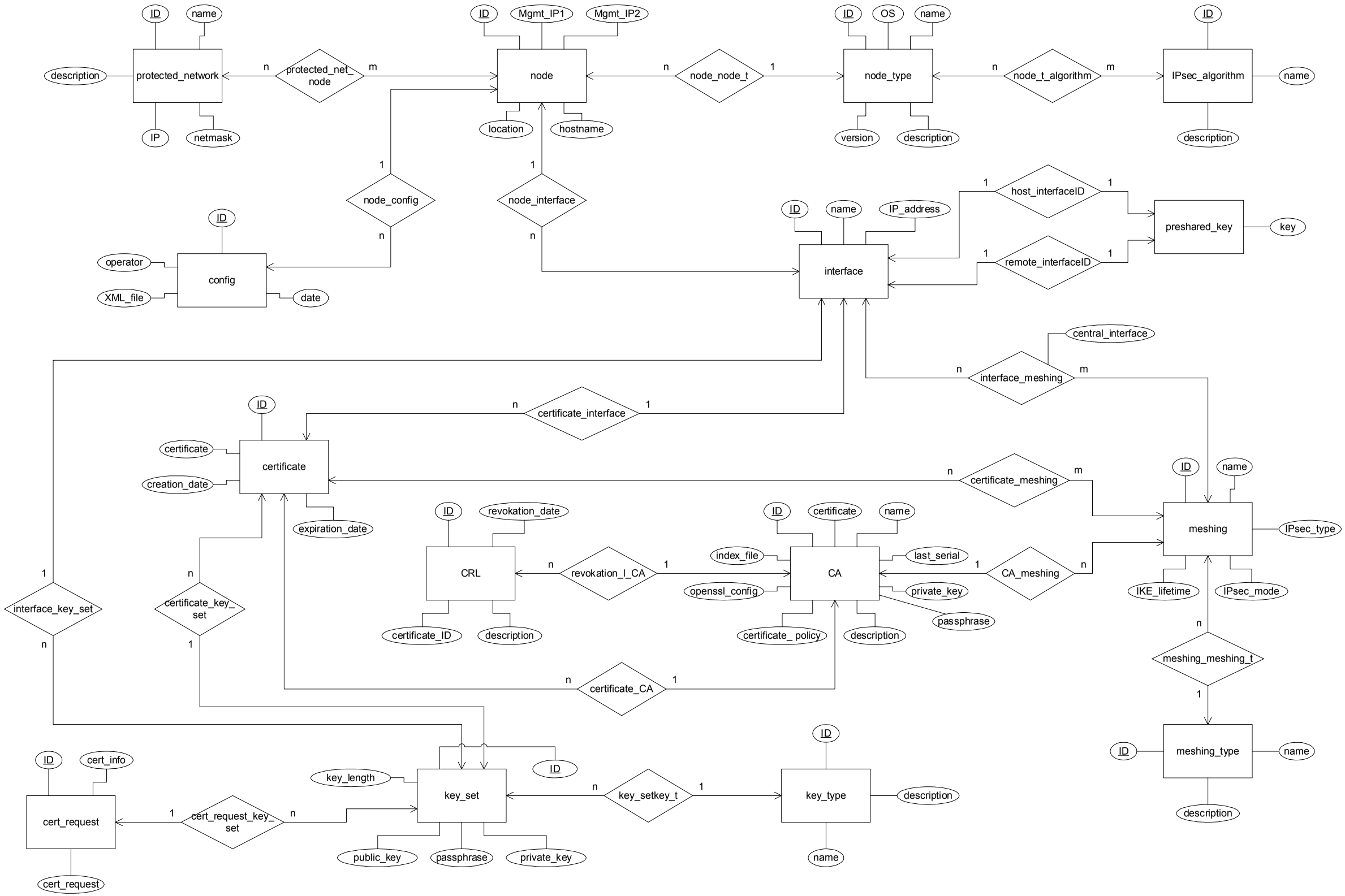
```
ALTER TABLE certificate ADD CONSTRAINT certificate_pk1 PRIMARY KEY
(id);
ALTER TABLE key_set ADD CONSTRAINT key_set_pk1 PRIMARY KEY (id);
ALTER TABLE cert_request ADD CONSTRAINT cert_request_pk1 PRIMARY KEY
(id);
ALTER TABLE key_type ADD CONSTRAINT key_type_pk1 PRIMARY KEY (id);
ALTER TABLE crl ADD CONSTRAINT crl_pk1 PRIMARY KEY (id);
ALTER TABLE ca ADD CONSTRAINT ca_pk1 PRIMARY KEY (id);
ALTER TABLE meshing ADD CONSTRAINT meshing_pk1 PRIMARY KEY (id);
ALTER TABLE meshing_type ADD CONSTRAINT meshing_type_pk1 PRIMARY KEY
(id);



/* add foreign key constraints */

ALTER TABLE interface ADD (
      CONSTRAINT interface_fk1 FOREIGN KEY (nodeid) REFERENCES
node(id)
);

ALTER TABLE node_t_algorithm ADD (
      CONSTRAINT node_t_algorithm_fk1 FOREIGN KEY (node_typeid)
REFERENCES node_type(id),
      CONSTRAINT node_t_algorithm_fk2 FOREIGN KEY (ipsec_algid)
REFERENCES ipsec_alg(id)
);

ALTER TABLE protected_net_node ADD (
      CONSTRAINT protected_net_node_fk1 FOREIGN KEY (nodeid)
REFERENCES node(id),
      CONSTRAINT protected_net_node_fk2 FOREIGN KEY
(protected_networkid) REFERENCES protected_network(id)
);

ALTER TABLE node ADD (
      CONSTRAINT node_fk1 FOREIGN KEY (node_typeid) REFERENCES
node_type(id)
);

ALTER TABLE config ADD (
      CONSTRAINT config_fk1 FOREIGN KEY (nodeid) REFERENCES node(id)
);

ALTER TABLE certificate ADD (
      CONSTRAINT certificate_fk1 FOREIGN KEY (caid) REFERENCES
ca(id),
      CONSTRAINT certificate_fk2 FOREIGN KEY (key_setid) REFERENCES
key_set(id),
      CONSTRAINT certificate_fk3 FOREIGN KEY (interfaceid) REFERENCES
interface(id)
);

ALTER TABLE key_set ADD (
      CONSTRAINT key_set_fk1 FOREIGN KEY (interfaceid) REFERENCES
interface(id),
      CONSTRAINT key_set_fk2 FOREIGN KEY (cert_requestid) REFERENCES
cert_request(id),
     CONSTRAINT key_set_fk3 FOREIGN KEY (key_typeid) REFERENCES
key_type(id)
);

ALTER TABLE crl ADD (
      CONSTRAINT crl_fk1 FOREIGN KEY (caid) REFERENCES ca(id)
);

ALTER TABLE meshing ADD (
      CONSTRAINT meshing_fk1 FOREIGN KEY (caid) REFERENCES ca(id),
```

```
      CONSTRAINT meshing_fk2 FOREIGN KEY (meshing_typeid) REFERENCES
meshing_type(id)
);


ALTER TABLE certificate_meshing ADD (
      CONSTRAINT certificate_meshing_fk1 FOREIGN KEY (meshingid)
REFERENCES meshing(id),
      CONSTRAINT certificate_meshing_fk2 FOREIGN KEY (certificateid)
REFERENCES certificate(id)
);

ALTER TABLE interface_meshing ADD (
      CONSTRAINT interface_meshing_fk1 FOREIGN KEY (meshingid)
REFERENCES meshing(id),
      CONSTRAINT interface_meshing_fk2 FOREIGN KEY (interfaceid)
REFERENCES interface(id)
);

ALTER TABLE preshared_key ADD (
      CONSTRAINT preshared_key_fk1 FOREIGN KEY (host_interfaceid)
REFERENCES interface(id),
      CONSTRAINT preshared_key_fk2 FOREIGN KEY (remote_interfaceid)
REFERENCES interface(id)
);
```



Relationales Datenbank Schema

19. Oktober 2001
Version 1.0



ER - Diagramm

19. Oktober 2001
Version 1.0