Einführung kryptographischer Techniken zur gesicherten Nutzung des Internet bei der Propack Data GmbH

Diplomarbeit am Institut für Telematik
Fakultät für Informatik
Universität Karlsruhe (TH)

von
Michael Ströder

Betreuer:
Prof. Dr. Dr. h.c. mult. G. Krüger
Dr. F. Krüger (Propack Data GmbH)
Dipl.−Inform. S. Dresler
Dipl.−Inform. F. Pählke
Dipl.−Inform. G. Schäfer

Tag der Ausgabe: 01. August 1998
Tag der Abgabe: 31. Januar 1999

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 31. Januar 1999

Inhaltsverzeichnis

1 Einleitung

Es taucht in Unternehmen immer öfter die Fragestellung auf, ob es nicht Vorteile bringen würde, Dienstleistungen über das Internet anzubieten, welche über die meist üblichen Anwendungen wie unverbindliche, kurze E−Mail−Nachrichten und WWW−Seiten zu Werbezwecken hinausgehen. Speziell bei der Propack Data GmbH als Software−Firma reicht die Palette der möglichen Dienstleistungen dabei von der simplen Bereitstellung von Daten zum bequemen Laden über das Internet, wie z.B. neue Produktversionen über den FTP− oder WWW−Dienst, über den Zugang zu On−Line−Datenbanken zur Organisation der eigenen Arbeitsabläufe (Workflow) bis hin zur Integration von Internet−Diensten in eigene Softwareprodukte. Aufgrund der steigenden räumlichen Verteilung und einer vermehrten kooperativen Zusammenarbeit mit anderen Firmen werden zudem die Anforderungen für kontrollierten Dokumentenaustausch und zur verteilten Software−Entwicklung immer dringlicher.
Die Vorteile liegen dabei klar auf der Hand: Neben offensichtlichen Zeit− und Kostenvorteilen garantiert die Verwendung der im Internet üblichen offenen Standards eine Interoperabilität über Systemgrenzen hinweg. Zudem lassen sich durch die Integration einer Internet−Anbindung in eigene Produkte, neue Anwendungsfelder und damit neue Märkte erschließen.

Nachdem man zum Schluß gekommen ist, daß es tatsächlich viele Vorteile bringt, Dienste über das Internet zugänglich zu machen, erkennt man schnell, daß dies mit einer Reihe von Sicherheitsproblemen behaftet ist: Manche Angebote müssen zwingend auf einen bestimmten Benutzerkreis beschränkt bleiben, d.h. es ist zum einen eine sichere Authentifizierung von Benutzern und eine darauf basierende Autorisierung der Nutzung der Internet−Dienste notwendig. Zum anderen müssen die Dienste zuverlässig angeboten werden: Neben der Zuverlässigkeit der verwendeten Systeme muß gewährleistet sein, daß die Daten nicht mitgelesen oder verfälscht werden.
Eine detailliertere Analyse der im Internet vorherrschenden Sicherheitsprobleme führt sehr schnell zur Erkenntnis, daß sich diese Probleme derzeit nur durch den konsequenten Einsatz von Verschlüsselungstechniken in den Griff bekommen lassen. Mit dem Willen zum Einsatz von Verschlüsselung ist es aber nicht getan, es muß vielmehr konsequent eine entsprechende Infrastruktur aufgebaut werden. Eine solche Infrastruktur umfaßt zum einen die relevanten technischen Standards und die darauf aufbauenden Implementierungen und zum anderen die betreuenden und benutzenden Instanzen dieser Infrastruktur.

1.1 Aufgabenstellung

Ziel der Diplomarbeit war es, ein Konzept zu entwickeln, mit dem weite Teile der internen und externen Kommunikationsbeziehungen mit einem vertretbaren Maß an Sicherheit und Aufwand über das Internet abgewickelt werden können. Speziell sollte dabei der praktische Einsatz von Verschlüsselungstechniken zur Sicherung von Internet−Diensten berücksichtigt werden.
Dazu sollten zuerst die konkret bestehenden Gefahren der Internet−Nutzung ohne Sicherungsmaßnahmen aufgezeigt werden. Zur Sicherung der Internet−Dienste sollten existierende kryptographische Standards

recherchiert, auf ihre Anwendbarkeit für die bei der Propack Data GmbH benötigten Kommunikationsbeziehungen untersucht und geeignete Techniken ausgewählt werden. Anschließend sollte eine Infrastruktur für die Nutzung kryptographischer Techniken konzipiert und realisiert werden. Zudem sollte diese Infrastruktur anhand von Beispielanwendungen auf ihre praktische Brauchbarkeit untersucht werden.

1.2 Gliederung

In Kapitel 2 werden zuerst Risiken der Internet−Nutzung und konventionelle Sicherungsmaßnahmen dargestellt, um eine Motivation für den Einsatz der in Kapitel 3 eingeführten kryptographischen Techniken zu geben. Basierend auf den in Kapitel 4 beschriebenen Anforderungen und Voraussetzungen, wird in Kapitel 5 bereits verfügbare Software vorgestellt, die für den praktischen Einsatz in Frage kam. Kapitel 6 beschreibt dann die Realisierung einer Zertifizierungsstelle als kryptographische Basisinfrastruktur und Kapitel 7 einige weitergehende Beispielanwendungen. Zu guter letzt faßt Kapitel 8 noch einmal die gewonnenen Erkenntnisse und erreichten Ziele zusammen und formuliert eine Zielrichtung für die zukünftige Weiterentwicklung.

2 Sicherheitsaspekte im Internet

In diesem Kapitel werden verschiedene grundlegende Aspekte erläutert, welche die Einführung kryptographischer Techniken zur Internet−Nutzung motivieren. Es werden sehr kurz mögliche Angriffe auf die bei der Internet−Nutzung beteiligten Komponenten beschrieben. Die Mechanismen sind aber so komplex, daß die Angriffsszenarien hier nur angedeutet werden können. Auch wird in dieser Arbeit keine komplexe Risikoanalyse durchgeführt.

2.1 Grundlegende Definitionen

Im nun folgenden Abschnitt werden einige grundlegende Begriffe definiert, insbesondere ist die genaue Unterscheidung in *intern* und *extern* wichtig.

2.1.1 Intern und Extern

Die genaue Unterscheidung zwischen *intern* und *extern* ist für die Auswahl der zu ergreifenden Sicherheitsmaßnahmen von eklatanter Bedeutung. Immer wieder muß bei der Inbetriebnahme, Modifikation und Wartung von Systemen entschieden werden, ob man das System als vertrauenswürdig ansieht oder nicht. Dabei reduziert man bei niedrigen bis mittleren Sicherheitsanforderungen die Beurteilung meist darauf, ob ein System als internes oder externes System angesehen wird, was natürlich nicht alle Sicherheitsaspekte in die Beurteilung mit einbezieht, aber eben praktikabler ist, als jedesmal eine umfassende, individuelle Risikoanalyse durchzuführen. Eine Definition anhand des physikalischen Standortes innerhalb oder außerhalb der Firmenräumlichkeiten würde zu kurz greifen. Z.B. sollten Systeme von Kunden, welche sich zu Service−Zwecken in den Räumen einer Firma befinden, nicht dieselben Rechte bekommen wie *interne Systeme* dieser Firma.

> Als *interne Systeme* werden Systeme bezeichnet, die sich unter vollständiger administrativer Aufsicht von entsprechend autorisierten, internen Mitarbeitern befinden. Dementsprechend sind als *externe Systeme* solche Systeme zu bezeichnen, bei denen die vollständige administrative Aufsicht durch autorisierte, interne Mitarbeiter nicht sicher gegeben ist.

Wie leicht ersichtlich ist, kann die Forderung vollständiger administrativer Aufsicht nicht immer einfach verwirklicht werden, sondern hängt vielmehr z.B. auch von Sicherheitsmaßnahmen bezüglich des physikalischen Zugangs ab. Strenggenommen sind auch Systeme extern, welche zwar definitionsgemäß von internen Mitarbeitern gepflegt werden, aber bei denen man z.B. unbeobachtet ein Boot−Speichermedium einlegen kann. Solche Fragestellungen sind aber nicht Gegenstand dieser Arbeit.
Die o.g. Definition ermöglicht auch verschieden granulierte interne Sicherheitsbereiche, indem man die jeweilige Menge der autorisierten Administratoren entsprechend anpaßt (Abbildung 1). In unserem Beispiel sieht der Bereich A alle Systeme als extern an, die nicht in A liegen, wogegen z.B. für C auch die Systeme des Bereichs B als intern gelten.

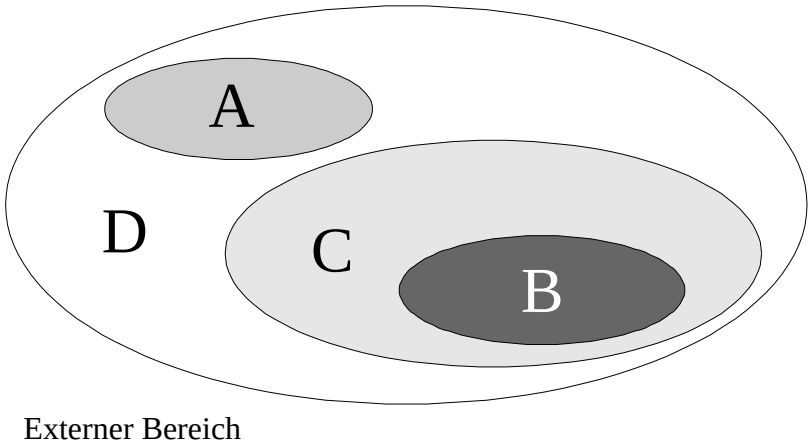


Abbildung 1: Einteilung in verschiedene Sicherheitsbereiche

Bei der Einteilung von Systemen in verschiedene Sicherheitszonen ist zu beachten, daß eine Transitivität gewahrt bleiben sollte, d.h. es sollten keine überlappenden Sicherheitsbereiche wie in Abbildung 2 existieren. Haben zwei Sicherheitsbereiche A und B eine nicht leere Schnittmenge C, so sollte diese Schnittmenge C entweder eine Teilmenge von A oder eine Teilmenge von B sein.

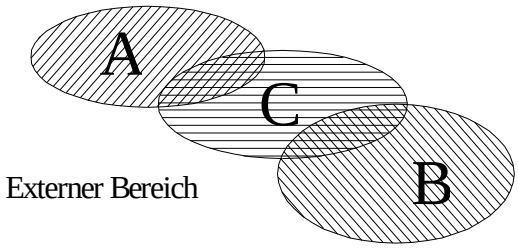


Abbildung 2: Nicht empfehlenswerte Einteilung in Sicherheitsbereiche

Externe Kommunikation wird in dieser Arbeit wie folgt definiert:

> Unter e*xterne Kommunikation* fallen alle Kommunikationsbeziehungen von internen Systemen zu externen Systemen.

Dedizierte Leitungen sollen folgendermaßen verstanden werden:

> *Als dedizierte Leitungen* werden Übertragungswege angesehen, welche sich entweder unter vollständiger administrativer Aufsicht von entsprechend autorisierten, internen Mitarbeitern befinden, oder deren Vertraulichkeit und Integrität als hinreichend gesichert angesehen wird.

Letztenendes kann meist nie hinreichend geklärt werden, ob Vertraulichkeit und Integrität einer Übertragungsstrecke als hinreichend gesichert angesehen werden können. Dies erfordert wieder eine individuelle Risikoanalyse für jedes System, in der geklärt werden muß, wieviel Aufwand der Angriff für einen Angreifer bedeutet, welche Verluste bei einem Angriff auftreten können, die wiederum auch von Haftungsfragen abhängen, etc. Auch solche Fragestellungen werden in dieser Arbeit nicht behandelt.

Mit der oben genannten Definition können externe Kommunikationsbeziehungen noch einmal unterteilt werden:

1. Kommunikationsbeziehung mit unbekanntem Partner über das öffentliche Internet (z.B. öffentlicher WWW−Server zu Werbezwecken)
2. Kommunikationsbeziehung mit bekanntem Partner über das öffentliche Internet (z.B. private Bereiche auf dem WWW−Server)
3. Kommunikationsbeziehung mit unbekanntem Partner über dedizierte Leitungen
4. Kommunikationsbeziehung mit bekanntem Partner über dedizierte Leitungen (z.B. Fernwartung)

Je nach Zugehörigkeit zu den einzelnen Klassen können die Maßnahmen zur Sicherung der Kommunikation unterschiedlich ausfallen. Es ist an dieser Stelle sinnvoll, auch die möglichen Angriffsarten grob zu klassifizieren.

2.1.2 Angriffsarten

2.1.2.1 Interne Angriffe

Angriffe direkt am Client−System oder im LAN sind meist technisch besonders leicht und lohnend. Der interne Angreifer weiß z.B. bereits die Identität des abgehörten Benutzers und kann das durch die Abhöraktion gewonnene Wissen gezielter einsetzen. Ein interner Angreifer weiß wahrscheinlich mehr über den Sinn und Verwendungszweck von übertragenen Daten und kann sie entsprechend fälschen, so daß die enthaltene Information immer noch plausibel wirkt, der verfälschte Inhalt ihm selbst aber einen persönlichen Vorteil bringt (z.B. Daten der Lohnbuchhaltung).
Das heißt aber nicht, daß das tatsächliche Risiko interner Angriffe deshalb höher ist. Ein potentieller interner Angreifer wird u.U. durch zu erwartende Sanktionen von einem Mißbrauch abgehalten, da der juristische Kontext präzise gefaßt ist.

2.1.2.2 Externe Angriffe

Angriffe von außen sind für den Angreifer meist gefahrloser, allerdings auch technisch schwieriger. Ein externer Angriff ist meist nur schwer zum Urheber zurückzuverfolgen. Juristische Sanktionen sind, gerade im Hinblick auf die globale Ausbreitung des Internets, nur schwer durchzusetzen. Oft hat der externe Angreifer auch keinen persönlichen Vorteil von dem Angriff, da er gewonnene Daten nicht interpretieren kann. Meist geht es dem Angreifer nur um einen Image−Gewinn in der "Hacker−Szene" oder den Spaß an der Zerstörung. Häufig werden deshalb sehr prestigeträchtige Server angegriffen (WWW−Server der CIA, Air Force etc. vergleiche [2600]). Dies bedeutet aber nicht, daß externe Angriffe auf weniger bekannte Institutionen nicht erfolgen. Es werden viele Rechner angegriffen, um anonymisierte Ausgangspunkte für neue Angriffe zu haben.

2.1.2.3 Passives und aktives Abhören

Beim *passiven Abhören* greift der Angreifer lediglich die auf der Übertragungsstrecke übertragenen Daten ab (siehe Abbildung 3). Dies kann je nach Art der Übertragungsstrecke mit verschiedenen physikalischen Methoden geschehen (z.B. induktiv, kapazitiv etc.). Das übertragene Signal wird dabei nicht verändert, die abgehörten Teilnehmer können das Abhören normalerweise nicht bemerken.

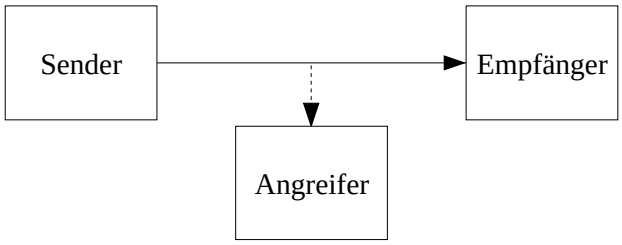


Abbildung 3: Passives Abhören

Im Gegensatz dazu unterbricht der Angreifer beim *aktiven Abhören* die Übertragungsstrecke und schaltet sich derart zwischen die eigentlichen Kommunikationsteilnehmer, daß er jedem Teilnehmer als der jeweils andere Teilnehmer erscheint (siehe Abbildung 4).

Abbildung 4: Aktives Abhören

Dabei kann der Angreifer die übertragenen Daten ändern. Diese Art des Angriffs nennt man auch *Man−in−the−middle−attack.*

2.2 Bedrohungen bei der Internet−Nutzung

Es gibt viele Punkte auf dem Weg vom Client−Rechner zu einem Server, an denen die Kommunikation belauscht, verfälscht oder unmöglich gemacht werden kann. In diesem Abschnitt werden einige Sicherheitsrisiken bei der Nutzung von Internet−Diensten dargestellt und ein kurzer Überblick der Sicherungsmöglichkeiten gegeben. Die Zusammenhänge sind in [Ches96], [Gar97], [Ker91], [Ste96] und [BSI98a] weitaus detaillierter dargestellt. Eine exaktere Taxonomie von im Internet möglichen Angriffsarten versucht [Ranu98].

2.2.1 Zugriff auf Übertragungsstrecken

Im Prinzip kann jede Übertragungsstrecke abgehört bzw. gestört werden, nur der erforderliche Aufwand ist je nach Übertragungstechnik und administrativen Gegebenheiten unterschiedlich. Beim Abhören kann man passives und aktives Abhören unterscheiden.
Im täglichen Gebrauch von Datennetzen ergeben sich beispielsweise folgende lohnende Angriffspunkte für das physikalische Abhören:

- Wählleitungen (ISDN und Modem), Telefonanlagen
- Funkstrecken
- LAN−Leitungen und LAN−Netzwerkkomponenten

Der technische Aufwand für diese Arten von Angriffen ist je nach Art der Übertragungsstrecke vergleichsweise hoch. Der Aufwand für konventionelle Schutzmechanismen gegen physikalisches Abhören ist aber ebenfalls erheblich, die Schutzmaßnahmen oft nicht praktisch durchführbar. In den

folgenden Unterabschnitten sollen ein paar kurze Beispiele für Abhörmöglichkeiten der Übertragungsstrecke gegeben werden.

2.2.1.1 Wählleitungen

Wählleitungen des Telefonnetzes werden gerne als temporäre Datenkommunikationsverbindungen benutzt, speziell auch, um Mitarbeitern zuhause oder Partnerfirmen den Zugriff aufs interne Firmennetz zu ermöglichen. Es ergeben sich aber mehrere mögliche Sicherheitsprobleme:

- Modem
  Das passive Abhören von Modem−Leitungen bei Modem−Geschwindigkeiten ab 14,4kBit/sek. ist signaltechnisch vergleichsweise aufwendig, aber inzwischen möglich. Aktives Abhören ist signaltechnisch einfacher möglich, aber ein Angreifer muß sich an einer definierten Stelle des Übertragungswegs einklinken.
- ISDN
  Das passive Abhören von ISDN−Leitungen zwischen Endgerät und Vermittlungsstelle ist möglich. Zudem können von einem Angreifer zusätzliche ISDN−Merkmale ausgenutzt werden, wie z.B. die im deutschen Telefonnetz verbotene Zeugenschaltung, um ISDN−Daten an Dritte weiterzuleiten. Oft erfolgt z.B. in ISDN−Routern eine Authentifizierung basierend auf ISDN−Nummern des Senders, welche aber nicht immer zuverlässig sein muß.
- Leitungswege
  Oft sind Telefonleitungen, speziell in alten Gebäuden, für jeden zugänglich in Kellerräumen oder an Hauswänden angebracht, Manipulationen also ohne großen Aufwand möglich. Auch werden heute längst nicht mehr alle Telefon−Verbindungen über geschlossene Leitungssysteme geführt. Vielmehr muß insbesondere bei Auslandsverbindungen davon ausgegangen werden, daß die Verbindungen über leicht abhörbare Funkstrecken geleitet werden.
- Telefonanlagen
  Heutige Telefonanlagen bieten komfortable Mechanismen wie flexibles Routing, Weiterleitungen, Fernwartungszugänge o.ä. Sie werden somit zunehmend komplexer und stellen damit ein immer höheres Sicherheitsrisiko dar. Zudem stehen TK−Anlagen meist nicht unter interner administrativer Kontrolle, sondern werden vom Hersteller (fern−)gewartet. Angreifer können somit TK−Anlagen sabotieren, Verbindungsdaten mitprotokollieren oder Verbindungsinhalte weiterleiten.

Meist wird den Wählleitungen blindlings vertraut, oft noch aus der traditionellen Vorstellung einer staatlich beaufsichtigten, streng leitungsvermittelten Kommunikationsverbindung heraus. Die o.g. kurze Aufzählung sollte schon deutlich gemacht haben, daß man die Vertraulichkeit und Integrität von Wählleitungen keinesfalls als gegeben annehmen kann.

2.2.1.2 Funkstrecken

Beliebt ist auch der Einsatz von Richtfunkstrecken zur kostengünstigen Überbrückung von auseinander liegenden Gebäuden. Leider lassen sich Funkstrecken besonders leicht unbemerkt passiv abhören oder auch der übertragene Datenstrom stören, eine gezielte Verfälschung (aktives Abhören) ist aber nicht möglich.

2.2.1.3 LAN

Client und Server sind meist jeweils an einem lokalen Netz (LAN) angeschlossen − typischerweise ein Ethernet oder Token−Ring. In solchen Broadcast−Netzen ist das Mithören, Umleiten und Verfälschen von Datenpaketen besonders leicht, wenn sich der Rechner des Angreifers im selben Netzsegment befindet:

- Man versetzt die Netzwerkkarte eines zur Verfügung stehenden Rechners in den *promiscuous mode* und kann alle auf dem LAN−Segment gesendeten Pakete empfangen (z.B. mit *tcpdump* unter Unix, *LANalyzer* für Windows o.ä. Netzwerkmonitoren). In der Regel gehen Kennwörter und sonstige vertrauliche Informationen im Klartext über das Netz und können somit von einem Angreifer bequem abgehört und protokolliert werden.
- Zudem bietet auf dieser Ebene das zustandslose *Address Resolution Protocol* (ARP) einen guten Angriffspunkt, einem Rechner gefälschte MAC−Adressen in seinen ARP−Cache zu schreiben und somit IP−Pakete auf einen anderen Ziel−Rechner im Segment umzuleiten [Schm97b].
- Es können bereits bestehende TCP−Verbindungen von einem Angreifer übernommen werden [Schm97a].

2.2.2 DNS

Viele Mechanismen bei der Nutzung des Internet basieren auf der korrekten Umsetzung symbolischer, leicht zu merkender Adressen auf die eigentliche numerische IP−Adresse. Diese Umsetzung bewerkstelligt der Namensdienst *DNS* (Domain Name Services, siehe auch 3.3.8.3.1). Durch eine Manipulation dieser Namens−Adressen−Umsetzung (DNS−Spoofing) ergeben sich für einen Angreifer verschiedene Möglichkeiten (siehe auch [Mraz97]):

- Ein Zugriff auf bestimmte Adressen kann unmöglich gemacht werden.
- Der Zugriff auf eine namentliche Adresse kann umgeleitet werden.
- Namensbasierte Sicherheitsmechanismen können umgangen werden.

2.2.3 Routing

Mehrere Angriffsarten auf IP−Routing−Ebene sind möglich:

- IP−Spoofing
  Ein Angreifer kann seinen IP−Paketen eine gefälschte Sender−Adresse geben, um IP−Filterregeln zu umgehen oder IP−adreßbasierte Authentifizierungsmechanismen zu täuschen. Diese Angriffsart nennt man *IP−Spoofing*.
- ICMP (Internet Control Message Protocol)
  Ein Router kann gefälschte *ICMP Redirect*−Pakete an Hosts oder Router schicken, um die Routen zu ändern. Dies ist eigentlich eine nützliche Eigenschaft, welche der dynamischen Optimierung von Routen im Internet dient, kann aber leicht von Angreifern mißbraucht werden (siehe [Schm97a]).

- Fragmentierte IP−/ICMP−Pakete
  In der jüngeren Vergangenheit gab es diverse Denial−Of−Service−Angriffsmöglichkeiten, bei denen man Rechner mittels fragmentierter IP− oder ICMP−Pakete mit unzulässigen Sequenznummernfolgen zum Absturz bringen konnte. Solche Fehler sind typische Beispiele für unzureichende Sicherheitsabfragen in gängigen IP−Protokollimplementierungen − die Fehlerbehebung bestand meist aus nur einer weiteren Abfrage.

2.2.4 Proxies und Relays

Oft wird der Verkehr für einen bestimmten Internet−Dienst aus Gründen der Sicherheit und Effizienz über einen *Proxy*− oder *Relay*−Server abgewickelt. Dieser übernimmt gegenüber dem Client die Rolle des Servers, und gegenüber dem eigentlichen Server ist der Proxy ein Client. Die Übertragung wird auf Anwendungsebene weitergeleitet. Je nach Implementierung kann ein solcher Proxy−Server alle Daten mitprotokollieren und zwischenspeichern, was z.B. bei WWW−Cache−Proxies auch erwünscht ist. Meist werden dabei vom Proxy−Programm nur die WWW−Adresse (URL) und ein Zeitstempel des Zugriffs mitgeschrieben. Der Proxy kann aber auch so ausgelegt sein, daß er allen HTTP−Verkehr einfach mitprotokolliert oder WWW−Inhalte in einem Zwischenspeicher (Cache) ablegt, was im Prinzip bereits ein aktives Abhören der Kommunikation darstellt (Abschnitt 2.1.2.3). Mögliche Probleme dabei sind:

- Die Protokoll−Dateien der Proxy− oder Relay−Server sind schon als sicherheitsrelevant einzustufen, da sie einem Angreifer mindestens Informationen über das Anwenderverhalten geben[1]. Oft sind z.B. bei WWW−Zugriffen auch Parameter in den URLs enthalten (GET−Zugriff auf CGI−BIN−Programme), welche geheim gehalten werden sollten (vergleiche Abschnitt 2.4.3).
- Der Zwischenspeicher (Cache) enthält u.U. Daten, welche nicht allgemein zugänglich sein sollten, je nach Proxy−Konfiguration aber auf einmal allen Proxy−Benutzern zur Verfügung stehen (vergleiche Abschnitt 2.4.4).
- Ein unberechtigter Proxy−Benutzer kann womöglich bei unzureichend sorgfältiger Konfiguration der Proxy− bzw. WWW−Server an WWW−Inhalte gelangen, welche eigentlich nicht für ihn zugänglich sein sollten.
- Ein Proxy−Server stellt einen zentralen Angriffspunkt dar, um alle WWW−Zugriffe einer Institution abzuhören, auf falsche Server umzuleiten oder zu behindern.

2.2.5 Client−Systeme

Eine weitere Methode, an vertrauliche Informationen zu gelangen, ist eine Manipulation des Client−Systems an sich. Die meisten Arbeitsplatzrechner haben gar keine oder nur unzureichende Sicherungsmechanismen. Eingebaute Diskettenlaufwerke, Betriebssysteme ohne Benutzerverwaltung und ohne Mechanismen für eine Zugriffskontrolle (z.B. DOS, Windows 3.1/95, OS/2 etc.) machen weitreichende Modifikationen an der Betriebssystem− und Applikationskonfiguration möglich, wenn man physischen oder netzwerkbasierten Zugang hat. Einige Beispiele für Client−basierte Angriffsarten:

[1] Das Mitschreiben der Anwenderaktionen in Protokolldateien ist ohnehin datenschutzrechtlich bedenklich und bedarf u.U. besonderer Betriebsvereinbarungen innerhalb einer Institution.

- Die meisten Benutzer lassen vertrauliche Informationen von den verwendeten Programmen einfach auf der lokalen Platte des Arbeitsplatzrechners speichern (z.B. Zugangskennungen, Mails etc.). Oft sind diese Daten dann gar nicht oder sehr nachlässig gesichert und können leicht eingesehen bzw. an andere Teilnehmer übermittelt werden.
- Fehlerhafte bzw. falsch konfigurierte Software kann von einem Angreifer benutzt werden, um in den Besitz vertraulicher Informationen zu gelangen (z.B. diverse Fehler der gängigen WWW−Browser, welche Zugriff auf Dateien erlaubten [BrPr98][Kuba98]).
- Es wird von einem Angreifer Software installiert, welche, neben der vom Anwender gewünschten Funktion, weitgehend unsichtbare Nebeneffekte hat, die es einem Angreifer ermöglichen, an sensitive Zugangsdaten zu gelangen oder den Client−Rechner gleich vollständig aus der Ferne zu steuern (sog. "Trojanisches Pferd", z.B. BackOrifice unter Windows 95).
- Es wird von einem Angreifer Software installiert, welche den vertrauensseligen Benutzer zur Eingabe vertraulicher Daten auffordert, um angeblich eine vom Anwender gewünschte Tätigkeit ausführen zu können, diese Eingaben aber unbemerkt an den Angreifer übermittelt. Besonders gefährlich ist diese Variante bei dynamisch aus dem Internet ladbaren Programmen wie z.B. bei Java, ActiveX, JavaScript.
- Eine weitere Variante ist die Installation neuer Software oder Modifikation bestehender Software, welche berechtigterweise nach den vertraulichen Informationen fragt, aber unerwünschte Seiteneffekte hat (z.B. fällt pikanterweise die Anmeldeprozedur zum MSN (Microsoft Services Network) von Windows 95 in diese Kategorie, welche das komplette Festplatteninhaltsverzeichnis und die Systemkonfiguration an den Anmelde−Server übermittelt [Schul96]).
- Ein schlecht konfiguriertes X−Windows gestattet es, alle Benutzereingaben und −ausgaben mitzulesen bzw. zu fälschen.
- Modifizierte Treibersoftware (z.B. Tastatur− oder Netzwerktreiber) kann vertrauliche Eingaben etc. mitlesen und an einen Angreifer übermitteln.
- Modifikationen an der Hardware (z.B. Tastatur− oder Netzwerkanschluß) ermöglichen es einem Angreifer ebenfalls, vertrauliche Eingaben mitzulesen.
- Elektromagnetische Abstrahlungen von Rechnerkomponenten (insbesondere Tastatur und Bildschirm) ermöglichen das Abhören von Client−Systemen [Kuhn98].

Besonders gefährlich und unüberschaubar sind Kombinationen der Sicherheitslücken, beispielsweise ein Fehler im WWW−Browser, welcher es ermöglicht, eine modifizierte Software auf dem Client−Rechner zu installieren.

2.2.6 Server−Systeme

Ein besonders exponiertes Ziel für Angriffe sind die öffentlichen Server (z.B. WWW−Server) einer Institution, da die Dienstzugangspunkte und damit auch die Angriffspunkte wohlbekannt sind. Aber auch interne Server sind für Angreifer interessant, da sie oft noch sensitivere Daten enthalten. Beispielsweise sind Manipulationen an einem Server mit Daten der Lohnbuchhaltung durch Mitarbeiter für diese u.U. sehr lohnend. Die möglichen Sicherheitsprobleme bei Servern sind vielfältig:
Die Server−Software ist oft fehlerhaft oder fängt Fehleingaben nicht kontrolliert ab. Beispielsweise kann ein Angreifer Server−Prozessen zu viele Daten schicken, so daß ein im Server−Prozeß vorgesehener Pufferbereich überschrieben wird (Buffer overrun) und die vom Anwender gesendeten Daten dann auf den Betriebssytem−Stapelspeicher (Stack) mit den Rücksprungadressen gelangen, von wo aus durch

manipulierte Rücksprungadressen beliebige Befehle mit den Privilegien des Server−Prozesses ausgeführt werden können. Solche, geradezu klassisch zu nennenden, Angriffe lassen sich durch geeignete Prüfroutinen in den Server−Prozessen leicht abfangen, aber gerade solche Prüfroutinen werden bei Implementierungen gerne schlicht und einfach vergessen.
Die Server−Konfiguration wird bei größeren Installationen schnell kompliziert und unübersichtlich, insbesondere wenn eine bestehende Installation weiter ausgebaut wird. Dies stellt eine nicht unerhebliche Fehlerquelle dar, die dazu führen kann, daß Benutzer falsche Berechtigungen zugewiesen bekommen, oder interne Daten nach außen sichtbar werden.
Auch auf Servern kann von einem Angreifer Software installiert werden, welche, neben der vom Anwender oder Administrator gewünschten Funktion, weitgehend unsichtbare Nebeneffekte hat, die es einem Angreifer ermöglichen, an sensitive Zugangsdaten zu gelangen oder andere vom Angreifer installierte Software zu verbergen (z.B. sog. Root−Kits für Unix). Die Wirkung ist besonders verheerend, wenn diese Software mit administrativen Privilegien ausgeführt wird.
Ebenso sind viele Dienste− und Protokollimplementierungen nicht gegen Angriffe der Art Denial−Of−Service gefeit, welche die Nutzung der Dienste unmöglich machen oder sogar Server−Systeme zum Absturz bringen können.

2.3 Konventionelle Sicherung

In diesem Abschnitt werden kurz Maßnahmen beschrieben, welche ergriffen werden sollten, um vernetzte Systeme zu sichern. Diese Übersicht dient lediglich dazu, konventionelle Sicherungsmaßnahmen vorzustellen, die auch ergriffen werden müssen, um flankierend die Benutzung kryptographischer Techniken abzusichern. Einen vollständigeren Überblick über notwendige Sicherungsmaßnahmen geben [BSI98a] und [Ches96].

2.3.1 Sicherung von LAN−Segmenten

Lokale Netzwerke lassen sich gegen aktives und passives Abhören sichern, wenn die übertragenen Inhalte erst gar nicht physikalisch zugänglich sind. Z.B. kann in einem Ethernet−LAN durch den konsequenten Einsatz von Switches die Datenkommunikation auf verschiedene physikalische Segmente verteilt werden. Switches leiten Datenpakete nur auf dem Ethernet−Segment weiter, in welchem sich der Empfänger befindet, d.h. es gelangen nicht mehr alle Ethernet−Pakete anderer Rechner an das Netzwerk−Interface eines möglichen Angreifers. Hat man nun jeden einzelnen Arbeitsplatzrechner in einem getrennten Segment, so sind Angriffe durch einfaches Abhören nicht mehr möglich. Manche Switches bieten auch die Möglichkeit, auf MAC−Adressen basierte Filterregeln zu definieren (vergleiche hierzu auch [Ste96]).
Diese Lösung ist aber meist sehr kostenintensiv, da sie eine strukturierte Verkabelung voraussetzt und Switches wesentlich teurer sind als sog. Hubs (Broadcast−Verteiler im Ethernet). Zudem dürfen dem normalen Anwender keine LAN−Anschlüsse zu Segmenten zugänglich sein, die Datenpakete mehrerer Stationen transportieren. Es sind also u.U. abschließbare Schaltschränke für die Netzwerkkomponenten erforderlich.
Für die gesamte LAN−Kommunikation in einer Firma oder Organisation ist diese Lösung meist nicht praktikabel, aber evtl. zur zusätzlichen Sicherung sehr sensitiver Bereiche geeignet.

2.3.2 Sicherung des IP−Routings

IP−Filter−Router können durch Filterregeln einige Angriffsarten auf das Routing selbst und dahinter liegende Server−Dienste effektiv abwehren. Heutige Router−Software unterstützt das Filtern von Paketen mit Regeln basierend auf Quell− und Ziel−IP−Adressen, Quell− und Ziel−Port−Adressen, Protokollarten (ICMP, UDP, TCP) und physikalischer Schnittstelle. Hier sollen nur einige Grundregeln der Router−Konfiguration genannt werden:

- Als *source−routed* markierte Pakete sind vom Router zu verwerfen, da es keinen praktischen Anwendungsfall für solche Pakete gibt.
- IP−Spoofing kann bedingt verhindert werden, wenn Filterregeln aufgesetzt werden, die prüfen, ob eine bestimmte IP−Quell−Adresse überhaupt über eine bestimmte physikalische Schnittstelle kommen kann.
- ICMP Redirect−Pakete sind zu verwerfen. Optimierungen von Routen sollten bei sensitiven Routern (z.B. bei der "Firewall" einer Firma) von Hand vorgenommen werden.
- Hängt ein Filter−Router in einem LAN und die Integrität des Routings ist sehr sensitiv, so sollte die Zuordnung der MAC−Adressen zu IP−Adressen in festen Tabellen festgelegt werden, das ARP (Address Resolution Protocol) also abgeschaltet werden (vergleiche 2.2.3).
- Die Zugriffe auf die Server−Dienste sind mit geeigneten Filterregeln einzuschränken.
- Abgewiesene Pakete sollten mitprotokolliert werden, was allerdings die Gefahr eines DoS−Angriffs birgt, welcher die Protokolldatei vollaufen läßt.
- Das Abweisen der Pakete sollte dem Sender nicht bekannt gemacht werden, um die Anwendung von Filterregeln vor dem Angreifer zu verbergen (z.B. unter Linux *deny* statt *reject* als policy in den Filterregeln bei der Konfiguration von *ipfwadm*).
- In sehr sensitiven Anwendungen kann ein zusätzlicher Netzwerkmonitor die Einhaltung der auf dem Router definierten Filterregeln getrennt beobachten und Alarmmeldungen ausgeben, um eine evtl. Kompromittierung des Filter−Routers zu aufzudecken.

Immer zu beachten ist, daß die Anwendung von IP−Filterregeln eine sehr grobmaschige Vorgehensweise ist: Man sollte sich nicht fälschlicherweise in Sicherheit wiegen, wenn man einen Filter−Router hat, es kommen noch genug Daten auf höheren Protokollschichten durch, die kompromittierenden Inhalt haben können. Außerdem lassen sich IP−Filterregeln, speziell innerhalb eines internen Netzes, leicht durch IP−Spoofing umgehen.
Filter−Router werden oft auch fälschlicherweise als die "Firewall" an sich verstanden, was grundsätzlich falsch ist. Eine echte "Firewall" ist nicht genau eine Maschine, sondern vielmehr ein ganzes Sicherheitskonzept mit verschiedenen Komponenten, welches auf die individuelle Anwendung abgestimmt sein muß. Aus diesem Grund kann man eine "Firewall" auch nicht einfach im Laden kaufen, wie beispielsweise ein fahrbereites Auto.

2.3.3 Sicherung von Client−Systemen

Viele Sicherheitsprobleme lassen sich vermeiden, wenn man auf Client−Systemen nur Software aus zuverlässigen Quellen installiert. Insbesondere können in einer Firma das selbständige Laden und die eigenmächtige Installation von Software aus dem Internet durch eine restriktive Richtlinie unterbunden werden. Es ist allerdings schwer zu beurteilen, ob eine Quelle zuverlässig ist und die Software keine Hintertüren hat. Zudem sind solche Richtlinien bei der Nutzung von Java, JavaScript o.ä. wirkungslos.
Viele Betriebssysteme ermöglichen es auch, Client−Systeme komplett bzw. deren Konfiguration von einer zentralen Server−Installation zu starten. Änderungen an der Konfiguration können dann nur von einem Administrator auf dem Boot−Server durchgeführt werden, das Client−System selbst wird bei jedem Neustart in den Ursprungszustand zurückgesetzt.
Je nach Anwendung kann Benutzern der Zugriff auf das System stark eingeschränkt werden. Kommen dann durch Unachtsamkeit des Benutzers Programme mit unerwünschten Nebeneffekten zur Ausführung, so laufen sie wenigstens in einer niedrig privilegierten Umgebung ab. Insbesondere sollten auch Administratoren unbekannte Programme nicht mit ihren Administratorrechten ausführen.

2.3.4 Sicherung von Server−Systemen

Durch den Einsatz verschiedener Sicherungsmechanismen sollte eine Server−Installation gegen eine Reihe von Gefahren geschützt werden:

- IP−Paketfilter
  Filter auf IP−Ebene lassen nur erlaubte IP−Pakete durch. Als Parameter für die Filterregeln können Quell− und Ziel−Adresse (IP−Adresse und Port−Nummer) sowie die Protokollart (ICMP/TCP/UDP) verwendet werden.
- Gateway−/Proxy−Dienste
  Wenn möglich, sollten Server−Dienste über Gateways der Anwendungsschicht geleitet werden, da diese das Anwendungsprotokoll kennen und damit eine bessere Filterung und Protokollierung ermöglichen. Dies erfordert eine Einteilung des Netzes in verschiedene, physikalisch getrennte Sicherheitsbereiche.
- Authentifizierung
  Bei benutzerbezogenen Server−Diensten sollten weitergehende Authentifizierungsmechanismen stattfinden. Keinesfalls sollte man sich bei sensitiven Anwendungen auf IP−Quell−Adressen verlassen. Vielmehr sollten zumindestens benutzerspezifische Kennwörter bzw. Einmal−Kennwörter zum Einsatz kommen.
- Restriktive Installation/Konfiguration
  Bei der Installation von Servern sollte nur gut geprüfte Software zum Einsatz kommen, und die Konfiguration sollte so einfach wie möglich gehalten werden. Auf jedem Server sollten möglichst wenige Dienste gleichzeitig laufen, damit nicht durch einen kompromittierten Dienst die Integrität aller anderen Dienste gefährdet wird. Evtl. sollte die eigentliche Betriebssysteminstallation gegen Veränderungen geschützt werden, z.B. durch die Verwendung von kryptographischen Mitteln oder ganz einfach durch schreibgeschützte Boot−Medien (z.B. SCSI−Platten mit Schreibschutzschalter oder CD−ROMs).

- Wartung
  Auch während des Betriebs sollte ein Server immer wieder gewartet werden. Dies bedeutet, daß z.B. alle Protokolldateien regelmäßig gesichtet und ausgewertet werden, Verzeichnisse mit temporären Dateien überprüft werden etc.
- Auditing
  Die Server−Installation sollte von Dritten überprüft werden. Hierbei kommen oft Konfigurationsfehler zutage, welche bei der Installation übersehen wurden. Für das Auditing sind vor allem auch eine ausführliche Dokumentation der Installation und eine Protokollierung möglichst vieler Betriebsdaten erforderlich.
- Informationsbeschaffung/Updates
  Zudem sollte man sich über bekannt gewordene Sicherheitsprobleme informieren. Hier bieten sich z.B. diverse Mailing−Listen und Sicherheitsforen an (z.B. CERT und Rootshell etc., siehe auch Anhang B).

Weitere Hinweise zur Sicherung von Servern, insbesondere für Unix−Derivate, finden sich in [Gar91], [Ches96], [Gar91] und [BSI98a], speziell zur Sicherung von WWW−Servern auch in [BSI98b].

2.4 Sichere CGI−BIN−Programmierung

Ein *CGI−BIN−Programm* ist ein einfacher zustandsloser Prozeß, der auf einem WWW−Server unter Kontrolle der WWW−Server−Software abläuft und je nach Aufruf−Methode seine Eingabeparameter über eine Umgebungsvariable oder über die Standardeingabe (stdin) vom WWW−Server erhält. Auf Standardausgabe (stdout) erzeugt ein CGI−BIN−Programm einen HTML−Text, welcher von der WWW−Server−Software über HTTP an den WWW−Browser weitergeleitet wird. Insbesondere ist also die eigentliche Netzwerkprogrammierung (Sockets o.ä.) nicht Aufgabe des CGI−BIN−Programms. Ein CGI−BIN−Programm läuft üblicherweise als Benutzerprozeß des Benutzers, der dem WWW−Server zugeordnet wurde und dessen Zugriffsrechte im System stark eingeschränkt sein sollten.
Da CGI−BIN−Programmen bei der Bereitstellung von Dienstleistungen im Internet im allgemeinen und in dieser Arbeit eine wichtige Rolle zukommt, sollen im folgenden einige für die Sicherheit relevante Aspekte detaillierter beschrieben werden, die bei der CGI−BIN−Programmierung beachtet werden müssen (vergleiche auch [CERT97.25][Gund96][Gar97]). Abschnitt 2.4.5 enthält eine kurze Beschreibung der sicherheitsrelevanten Eigenschaften des erstellten Python−Moduls *cgiforms.py*, welches die Eingaben für fast alle im Rahmen der Arbeit entstandenen CGI−BIN−Programme behandelt.

2.4.1 Programmierumgebung

Relevant für die Sicherheit von CGI−BIN−Programmen ist die gewählte Programmierumgebung nebst allen beteiligten Bibliotheken und der Konfiguration. Programmierumgebungen können selbst Fehler und Sicherheitslöcher enthalten, über welche ein Angreifer aus diesen Umgebungen zur Laufzeit ausbrechen kann.
Beispielsweise wurde die speziell für CGI−BIN−Programme beliebte Programmiersprache Perl früher auf Unix−Systemen meist mit gesetztem SUID−Root−Bit installiert, was zu einer Reihe ernster Sicherheitsprobleme führte. Konnte ein Angreifer aus dem CGI−BIN−Programm ausbrechen und im Perl−Interpreter Kommandos zur Ausführung bringen, so stand ihm das ganze System offen.

2.4.2 Aufruf externer Programme

Oft rufen CGI−BIN−Programme über ein Shell−Kommando externe Programme mit vorher über ein Formular eingegebenen Parametern auf. Ein typisches Beispiel ist das Versenden einer E−Mail mittels des meist auf Unix−Systemen vorhandenen *sendmail* an eine von einem Benutzer angegebene Adresse. Die Gefahr dabei ist, daß ein böswilliger Benutzer bei der Eingabe von Parametern Sonderzeichen angeben kann, welche von der Shell u.U. als Steuerzeichen interpretiert werden [CERT97.25]. Abhilfe schafft zuerst einmal, statt externer Programme möglichst Bibliotheksmodule der jeweiligen Programmierumgebung zur Erfüllung einer Aufgabe zu benutzen. Um beim Beispiel der E−Mail−Versendung und der Programmiersprache Python zu bleiben: Die Standardbibliothek der Python−Distribution hat ein Modul *smtplib.py*, welches das direkte (und nebenbei auch schnellere) Versenden einer E−Mail ermöglicht. Der externe Aufruf von *sendmail* ist also unnötig.

Als Beispiel für den Mißbrauch eines externen Shell−Aufrufs soll hier ein kurzes Python−Skript dienen, welches ohne Parameter aufgerufen ein Eingabefeld anzeigt bzw. mit Parameter einen *finger*−Zugriff mit der Benutzereingabe aus der Umgebungsvariable QUERY_STRING durchführt:

```
#!/usr/bin/python
import sys, os, urllib
if not os.environ['QUERY_STRING']:
  print 'Content-type: text/html\n'
  print '<ISINDEX>'
else:
  param = urllib.unquote_plus(os.environ['QUERY_STRING'])
  print 'Content-type: text/plain\n'
  sys.stdout.flush()
  os.system('/usr/bin/finger %s' % param)
```

Dieses Skript birgt, neben dem an sich schon mit Vorsicht zu genießenden Weiterleiten des Finger−Dienstes über einen WWW−Server, das Problem, daß ein Benutzer beliebige Parameter angeben kann, welche ungeprüft von der Shell ausgeführt werden. Der kritische Aufruf verbirgt sich in der Python−Anweisung:

```
  os.system('/usr/bin/finger %s' % param)
```

Beispielsweise erzeugt eine Eingabe von

```
michael@queen.ms.my && ls -l
```

neben der Ausgabe des Finger−Kommandos auch gleich noch ein Inhaltsverzeichnis des CGI−BIN−Programm−Verzeichnisses − ein Startpunkt für weitere Angriffe. Es können beliebige Kommandos nach der Shell−Meta−Sequenz && angehängt werden, welche dann mit den Privilegien des CGI−BIN−Programms ausgeführt werden.

```
[queen.ms.my]

Welcome to Linux version 2.0.35 at www.ms.my !

  6:07pm  up  1:14, 10 users,  load average: 0.30, 0.19, 0.12

Login: michael                          Name: Michael Stroeder
Directory: /home/michael             Shell: /bin/bash
On since Sat Oct 17 16:56 (CEST) on :0 from console
[..]
Mail last read Sat Oct 17 17:39 1998 (CEST)
No Plan.
-rwxr-xr-x   1 root  www            60 Aug 14 20:57 ldap-client-cgi.py
-rwxr-xr-x   1 root  www            54 Oct 17 17:44 malicious.py
-rwxr-xr-x   1 root  www           120 Nov 18  1997 printenv
-rwxr-xr-x   1 root  www           757 Nov 18  1997 test-cgi
-rwxr-xr-x   1 root  www            43 Jun 24 20:06 test-cgi.py
```

Ist ein Aufruf externer Programme unumgänglich, so schaffen bestimmte Maßnahmen mehr Sicherheit:

- Alle Eingaben sollten genauestens auf unzulässige Sonderzeichen untersucht werden. Z.B. enthält eine E−Mail−Adresse nach RFC822 kein Ampersand (&), eine solche Eingabe ist also als fehlerhaft abzulehnen.
- Die Übergabe der Parameter kann beim Zusammenbauen der Kommandozeile einfach gesichert werden, indem die Parameter wenn möglich (je nach verwendeter Shell) in Anführungszeichen eingeschlossen werden, um die Interpretation von in Parametern vorhandenen Shell−Meta−Zeichen durch die Shell zu unterbinden. In o.g. Beispiel würde diese Maßnahme das Skript bereits völlig entschärfen: Ein Aufruf mit os.system('/usr/bin/finger "%s"' % param) verhindert das Ausführen von Meta−Sequenzen durch die Shell, das *finger*−Kommando würde einfach mit einem ungültigen Parameter aufgerufen.
- CGI−BIN−Programme sollten immer in einer restriktiven Umgebung laufen. Z.B. bietet der WWW−Server Apache die Möglichkeit, nach Binden an die TCP/IP−Sockets unter einer Benutzerkennung mit weniger Privilegien zu laufen. Bricht ein Angreifer aus einem CGI−BIN−Programm aus, so ist wenigstens der Zugriff auf das Gesamtsystem stark eingeschränkt.

2.4.3 GET vs. POST

Es gibt zwei verschiedene HTTP−Methoden, ein CGI−BIN−Programm aufzurufen: *POST* und *GET*. Mit welcher Methode das CGI−BIN−Programm aufgerufen wird, wird mittels der URL in einem Hyperlink (<A HREF>−Tag) oder in einem HTML−Formular (METHOD−Parameter im <FORM>−Tag) festgelegt [Gund96][Mün98]. Hyperlinks lösen immer einen GET−Zugriff aus.

Beispiel: Das HTML−Formular

```
<FORM ACTION="http://www.ms.my/cgi-bin/test-cgi.py" METHOD="GET">
  <INPUT NAME=param1 VALUE="data1">
  <INPUT NAME=param2 VALUE="data2">
  <INPUT TYPE=SUBMIT VALUE="Send">
</FORM>
```

zeigt zwei Eingabefelder und einen Sendeknopf an. Beim Betätigen des Sendeknopfs wird das CGI−BIN−Programm mit der HTTP−Methode GET aufgerufen, und es werden die Parameter, als [Parameter]=[Datum]−Paare und jeweils durch ein Ampersand−Zeichen (&) getrennt, an die URL angehängt:

```
http://www.ms.my/cgi-bin/test-cgi.py?param1=data1&param2=data2
```

Die Ampersand− und Prozent−Zeichen, ISO−Sonderzeichen etc. werden dabei noch einmal getrennt kodiert, was aber im Beispiel vernachlässigt wird, um es nicht unnötig kompliziert zu machen. Das CGI−BIN−Programm bekommt den Parameteranteil der URL (ohne das Fragezeichen) vom WWW−Server in der Umgebungsvariable *QUERY_STRING* übergeben und muß die Parameternamen−/Datum−Paare selbst auswerten.

Zusätzlich besteht noch die Möglichkeit, Parameter als zusätzliche Pfadinformation mit der URL mitzugeben:

```
http://www.ms.my/cgi-bin/test-cgi.py/param1=data1/param2=data2
```

Auch hier bekommt das CGI−BIN−Programm den Parameteranteil der URL (der Teil hinter dem Skriptnamen) vom WWW−Server übergeben und muß die Parameternamen−/Datum−Paare selbst

auswerten, allerdings sind die Parameter in diesem Fall in der Umgebungsvariable *PATH_INFO* enthalten.

Die Vorteile der GET−Methode sind:

- Auch in Shell−Skripten ist der Zugriff auf Umgebungsvariablen problemlos möglich.
- Mittels GET kann auf CGI−BIN−Programme auch ohne ein HTML−Formular, nur durch Angabe einer URL, zugegriffen werden.
- Man kann bei der Entwicklung von CGI−BIN−Programmen testweise die URL verändern.

Ein Nachteil der GET−Methode ist:

- Die Länge der Eingaben ist beschränkt, der WWW−Server schneidet u.U. die Eingaben ab.

Die sicherheitsrelevanten Nachteile der GET−Methode sind:

- U.U. wird eine aufgerufene URL als Referer−Eintrag im HTTP−Kopf des nächsten WWW−Zugriffs übermittelt. Somit können vom Benutzer eingegebene Parameter auf Server gelangen, für die sie eigentlich gar nicht gedacht waren[2].
- WWW−Server protokollieren die URL eines jeden Zugriffs detailliert mit. Werden z.B. beim Aufruf eines CGI−BIN−Programms Kennwörter zur Authentifizierung übermittelt, so tauchen diese auch in der Protokolldatei des WWW−Servers auf.

Bei der POST−Methode werden die Eingabedaten vom WWW−Server auf die Standardeingabe des CGI−BIN−Programms gelenkt und tauchen somit nicht in der URL auf. Vertrauliche Daten werden also vom WWW−Server nicht gleich automatisch mitprotokolliert. Die Eingabedaten können in diesem Fall beliebig lang sein und verschiedene Darstellungsformen haben. Es wird vom WWW−Server kein EOF gesendet, weshalb das CGI−BIN−Programm erst die Länge der Eingabedaten in der Umgebungsvariable *CONTENT_LENGTH* ermitteln muß. Das Behandeln der POST−Methode ist ein wenig aufwendiger, und es ist ein HTML−Formular für die Eingabe der Daten notwendig. Für vertraulich zu behandelnde Eingaben ist die Methode POST jedoch eindeutig zu bevorzugen.

2.4.4 Sicherheitsprobleme mit WWW−Caches

Ein weiteres Problem bei der Verteilung von Datenbeständen über WWW ist die Zwischenspeicherung der übertragenen Daten in sog. *Cache*−Speichern der Proxy−Server und WWW−Browser. Die Zwischenspeicherung in Caches erfolgt, um bei einem späteren Zugriff auf die selbe URL evtl. schneller die Daten aus dem Cache an die Anwendung zu liefern, falls die Daten noch nicht veraltet sind. Dies birgt jedoch ein ernstzunehmendes Sicherheitsrisiko: Wird beispielsweise das Ergebnis eines WWW−Zugriffs auf einem öffentlich zugänglichen Cache−Proxy zwischengelagert, so ist die Vertraulichkeit nicht gewährleistet. Auch kann der Cache eines WWW−Browsers zum Vertraulichkeitsproblem werden, falls die Cache−Datenbestände mehreren Personen zugänglich sind. Die ausgelieferten HTML−Texte enthalten mit hoher Wahrscheinlichkeit auch wieder zur Authentifizierung benutzte Kennwörter, da CGI−BIN−Programme Zustände auf Seite des Clients halten müssen, was oft genug mit versteckten Eingabefeldern

[2] Z.B. nutzen dies manche Suchdienste für ihre Werbekunden aus, indem sie ein zentrales Werbebanner als CGI−BIN−Programm über GET aufrufen lassen. Der Werbekunde hat dann über die Referer−URL auch gleich die Suchbegriffe. Verknüpft man dies geeignet mit zentral von einem Werbering verwalteten Cookies, so kann man sich ein Interessenprofil eines Cookie−Besitzers erstellen.

in HTML−Formularen geschieht (auch *ldap−client−cgi.py* macht davon regen Gebrauch, siehe Anschnitt 7.4.1).

Eine solche Zwischenspeicherung in Cache−Speichern kann durch folgende Maßnahmen verhindert bzw. abgemildert werden:

- Ein CGI−BIN−Programm sollte im HTTP−Kopf einen der beiden Einträge *pragma: no−cache* oder *expire: [Zulu−Zeitangabe]* setzen, um eine Zwischenspeicherung ganz zu verbieten bzw. eine allzu lange Zwischenspeicherung zu vermeiden. Man kann sich allerdings nicht darauf verlassen, daß die beteiligte Software diese HTTP−Kopfeinträge auch auswertet und richtig beachtet. Auch sind diese Kopfeinträge für den End−Benutzer unkomfortabel, da die Zurück−Funktion der WWW−Browser dann mit einer leeren Seite beantwortet wird.
- Der Zugriff auf das CGI−BIN−Programm sollte am besten über eine SSL−gesicherte Verbindung geschehen, da eine solche Verbindung von Proxy−Servern mit der HTTP−Methode CONNECT transparent durchgereicht wird, es gibt eventuelle Vertraulichkeitsprobleme nur noch seitens des WWW−Browsers.

2.4.5 Das Modul cgiforms.py

Fast alle im Rahmen der Arbeit entstandenen CGI−BIN−Programme basieren auf dem selbst entwickelten Python−Modul *cgiforms.py*, welches neben einer erleichterten Ausgabe von HTML−Formularen und Eingabeinhalten vor allem auch die Prüfung der Eingabeparameter zur Aufgabe hat. Im Prinzip deklariert man zu Beginn in einem CGI−BIN−Programm alle zu erwartenden Parameter mit ihrem gültigem Format als regulärem Ausdruck. Das Einlesen, die Aufbereitung und Überprüfung der Parameter wird von cgiforms.py übernommen. Dabei werden POST− und GET−Zugriffe für das benutzende CGI−BIN−Programm transparent von *cgiforms.py* behandelt. Man kann also sein CGI−BIN−Programm bequem mit Aufrufen mittels GET−Methode testen und setzt danach einfach im das CGI−BIN−Programm aufrufenden Formular die Methode auf POST. Folgende Überprüfungen werden von *cgiforms.py* durchgeführt:

- Ist die Gesamtlänge der Eingabedaten zulässig?
- Sind alle gesendeten Parameternamen erwünscht?
- Ist die Anzahl der Parameter mit gleichem Parameternamen zulässig?
- Entsprechen alle Eingabefelder vollständig den vorher deklarierten regulären Ausdrücken?

Beispiel:
Das folgende Python−Skript deklariert ein Eingabeformular mit zwei Eingabefeldern für den Namen einer Person und die E−Mail−Adresse der Person.

```
#!/usr/bin/python
import cgiforms.py

form=cgiforms.formClass()
form.add(cgiforms.formInputClass('param1','Name',30,'[a-zA-Z. -]*'))
form.add(cgiforms.formInputClass('param2','E-Mail',30,'[a-zA-Z.-]*@[a-zA-Z.-]*'))

try:
  form.getparams()
except:
  # Hier folgt Fehlerbehandlung der einzelnen Ausnahmen
```

Ruft man das Skript auf, so werden die Parameter über *cgiforms.py* eingelesen und überprüft. Tritt bei der Parameterprüfung ein Fehler auf, so wird eine entsprechende Ausnahme generiert, welche jeweils einzeln vom CGI−BIN−Programm abgefangen werden sollte. Eine detailliertere Beschreibung des Moduls *cgiforms.py* findet sich in [Strö98a]. Schon in diesem Beispiel läßt sich leicht erkennen, daß eine zweckmäßige Überprüfung der Eingaben immer noch vom Autor des CGI−BIN−Programms abhängt, da er selbst die maximalen Eingabelängen und regulären Ausdrücke für jeden Parameter deklarieren muß. Nichts hindert ihn daran, sehr große Eingabelängen vorzusehen und '.*' für das Eingabemuster zu verwenden. Im o.g. Beispiel könnte man beispielsweise das Muster für die E−Mail−Adresse noch restriktiver verfeinern, wogegen bei der Angabe des Namens noch nicht einmal länderspezifische Sonderzeichen akzeptiert werden. Das Modul *cgiforms.py* ist also nur ein hilfreiches Rahmenwerk, es garantiert selbst aber noch keine sicheren CGI−BIN−Programme.

2.5 Zusammenfassung

Es wurden in diesem Kapitel beispielhaft diverse Sicherheitsprobleme bei vernetzten Rechnern, insbesondere bei der Nutzung der Internet−Dienste, dargestellt. Vor allem durch die vielfältigen Wechselwirkungen verschiedener Protokolle und Mechanismen sind die Risiken bei der Vernetzung von Rechnern schier unüberschaubar: Die möglichen Kombinationen aus fehlerhafter Software, Software mit beabsichtigten Seiteneffekten, fehlerhaften Konfigurationen und unzureichend gesicherten Protokollen lassen es letztlich unmöglich erscheinen, vernetzte Rechner gegen alle Angriffe zu sichern.
Konventionelle Sicherheitsmaßnahmen, vor allem für die Internet−Anbindung (Firewalls), sollten mit hinreichender Sorgfalt eingerichtet und betrieben werden. Für Hochsicherheitsbereiche (z.B. Lohnbuchhaltung, wichtige kryptographische Schlüsseldaten etc.) gilt ganz schlicht und einfach die Empfehlung, die dabei beteiligten Komponenten nicht zu vernetzen.
Viele der Probleme unzureichender Host−Authentifizierung und mangelnder Vertraulichkeit lassen sich durch den Einsatz kryptographischer Techniken lösen, deren Grundlagen und Verwendung im nun folgenden Kapitel 3 eingeführt werden.

3 Kryptographische Techniken

Zwar lassen sich einige der im Abschnitt 2.2 beschriebenen Sicherheitsrisiken durch geeignete »konventionelle« Sicherungsmaßnahmen (Abschnitt 2.3) vermindern, es bleibt aber ein vergleichsweise hohes Restrisiko, da die Möglichkeiten für eine Ende−zu−Ende−Sicherung der Authentizität und Autorisierung beschränkt bzw. sehr aufwendig sind. Unter der Ende−zu−Ende−Sicherung versteht man dabei die in sich geschlossene Sicherung von einem Client−Prozeß zu einem Server−Prozeß, d.h. die Sicherungsmechanismen des Protokolls zwischen den End−Prozessen sind nicht von Sicherungsmechanismen oder der Authentizität von Informationen darunterliegender Protokollschichten abhängig. Ein solche Ende−zu−Ende−Sicherung verspricht der Einsatz kryptographischer Techniken für die Authentifizierung und die Autorisierung.
In diesem Kapitel wird zuerst eine sehr kurze Einführung in das Gebiet der Kryptologie gegeben. Danach werden sukzessive weitergehende Konzepte und Verfahrensweisen zur Anwendung der Kryptographie bis hin zu konkreten Protokollen beschrieben.

3.1 Grundbegriffe

Die *Kryptologie* ist ein Teilgebiet der angewandten Mathematik und unterteilt sich in die *Kryptographie* und die *Kryptanalyse*. *Die Kryptographie* versucht, mit geeigneten Codierungen (*Verschlüsselung*) bzw. Prüfdaten Nachrichten gegen unbefugte Kenntnisnahme und Verfälschung zu sichern. Die *Kryptanalyse* versucht eine solche Verschlüsselung zu brechen. Eine weitaus ausführlichere Einführung findet man z.B. in [Schn96].

3.1.1 Kryptographie

Die Anforderungen an die moderne Kryptographie sind je nach Anwendung:

- *Vertraulichkeit*
  Der Inhalt von Nachrichten soll nur autorisierten Personen zur Kenntnis gelangen.
- *Integrität*
  Der Inhalt einer Nachricht soll nicht unbemerkt verändert werden können.
- *Authentizität*
  Der Urheber einer Nachricht soll zweifelsfrei eindeutig erkennbar sein.
- *Verbindlichkeit*
  Der Urheber einer Nachricht soll die Urheberschaft nicht abstreiten können.
- *Anonymität*
  U.U. ist auch die Vertraulichkeit der Nachrichtenübermittlung an sich zu gewährleisten.

Bei der Kryptographie müssen sich Sender und Empfänger auf ein bestimmtes Verfahren, den *Algorithmus* einigen. Dabei kann der Algorithmus selbst das Geheimnis sein, was aber den Nachteil hat,

daß sich Algorithmen meist nur sehr schwer geheimhalten lassen. Zudem kann der Algorithmus fehlerhaft oder nur unzureichend sicher sein, was aber nur durch eine möglichst breite wissenschaftliche Überprüfung mit kryptanalytischen Methoden hinreichend überprüft werden kann. Deshalb verwenden kryptographische Algorithmen in der Regel einen variablen *Schlüssel* als einziges zwischen Sender und Empfänger vereinbartes Geheimnis. Die Menge aller zu einem bestimmten Algorithmus kompatiblen Schlüssel nennt man den *Schlüsselraum* des Verfahrens. Als *Kryptosystem* bezeichnet man den Algorithmus, den Schlüsselraum sowie alle möglichen Klar− und Chiffretexte.

Aus der Tatsache, daß die Algorithmen möglichen Angreifern mit einer sehr hohen Wahrscheinlichkeit bekannt sind, folgt die Maxime von Kerckhoffs ([Schn96]):

> *Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus, sondern ausschließlich von der Geheimhaltung des Schlüssels abhängen.*

3.1.2 Kryptanalyse

Die Kryptanalyse verfolgt eines der folgenden Ziele:

- *Vollständiges Aufbrechen*
  Das Geheimnis, meist der geheime Schlüssel, wird ermittelt, so daß jede Nachricht mit dieser Algorithmus−/Schlüssel−Kombination lesbar wird.
- *Globale Deduktion*
  Es wird ein Verfahren gefunden, ohne das Geheimnis zu jedem Chiffretext den Klartext zu ermitteln.
- *Lokale Deduktion*
  Es wird ohne das Geheimnis der Klartext zu einem bestimmten Chiffretext gefunden.
- *Informationsdeduktion*
  Informationen über den Schlüssel oder den Klartext werden ermittelt.

Es gibt es mehrere verschiedene Klassen von Angriffen auf ein Kryptosystem:

- *Ciphertext Only*
  Es steht nur eine begrenzte Menge an Chiffretext für die Kryptanalyse zur Verfügung.
- *Known Plaintext*
  Es stehen Chiffretext und Klartext einer begrenzten Menge von Nachrichten für die Kryptanalyse zur Verfügung.
- *Chosen Plaintext*
  Es können ausgewählte Klartexte mit dem zu analysierenden Algorithmus und Schlüssel chiffriert werden.
- *Chosen Ciphertext*
  In diesem Fall werden bestimmte Chiffretexte ausgewählt, zu denen auch der Klartext vorliegt, um den Schlüssel zu ermitteln.

In der Regel bedarf es Jahre der kryptanalytischen Überprüfung, bevor ein Algorithmus als vertrauenswürdig angesehen werden kann. Meist ist es kein formaler mathematischer Beweis, welcher die Sicherheit eines Algorithmus bestätigt, sondern nur die Tatsache, daß kein kryptanalytischer Weg

gefunden wurde, den Algorithmus mit einem geringeren Aufwand zu brechen als dem Aufwand für das vollständige Durchsuchen des Schlüsselraums.
Daraus folgt auch, daß man es tunlichst vermeiden sollte, selbst ad hoc kryptographische Algorithmen zu erfinden. Oftmals sind solche Verfahren mit einem normalen Arbeitsplatzrechner in Millisekunden vollständig brechbar [Pesch98]. Dies gilt insbesondere für die "Verschlüsselung" in gängigen Büro−Applikationen (wie z.B. MS−Office) oder handelsüblichen Datenkompressionsprogrammen (z.B. ZIP). Leider verführt das Vorhandensein solcher Pseudo−Schutzmaßnahmen so manchen Anwender dazu, sich sicher zu fühlen und damit verschlüsselte Dateien als de facto ungesicherte Nachrichten zu versenden.

3.2 Kryptographische Verfahren

In diesem Abschnitt werden die grundlegenden Verfahren der Kryptographie kurz dargelegt. Weitergehende Beschreibungen der Verfahren finden sich in [Schn96].

3.2.1 Konzelationssysteme

Kryptosysteme, welche vor unbefugtem Mitlesen schützen sollen, also das Ziel der Vertraulichkeit verfolgen, nennt man *Konzelationssysteme* (lat. *celare*: verbergen, verheimlichen).

3.2.1.1 Notation

Das Kryptosystem soll beschrieben werden durch den Klartext p, das Chiffrat c, den Algorithmus A und den Schlüssel k. Die Umkehrung von Algorithmus A wird mit A^{-1} und der Schlüssel zum Dechiffrieren mit k^{-1} bezeichnet. Es gilt für den Vorgang des Chiffrierens $c = A(p,k)$ und $p = A^{-1}(c,k^{-1})$ für das Dechiffrieren.

3.2.1.2 Symmetrische Algorithmen

Bei *symmetrischen Algorithmen* einigen sich Sender und Empfänger auf ein gemeinsames Geheimnis, meist also einen gemeinsamen Schlüssel k für den gemeinsamen Algorithmus A. Es wird der selbe Schlüssel zum Verschlüsseln und Entschlüsseln der Nachricht verwendet (siehe Abbildung 5).

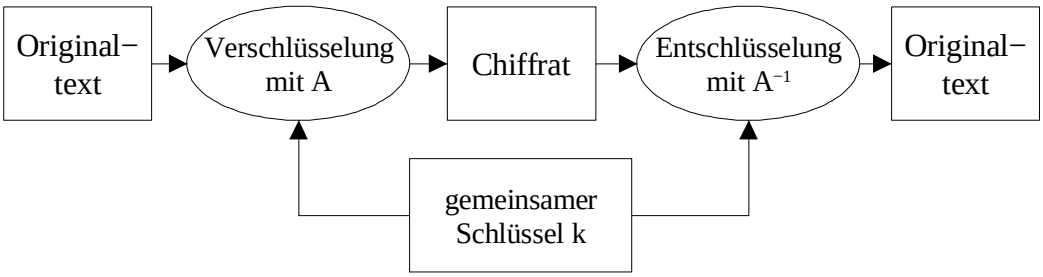


Abbildung 5: Symmetrische Verschlüsselung

Es gilt also: $k = k^{-1}$. Für diese Art der Kryptographie müssen sich Sender und Empfänger vor dem Kommunikationsvorgang über einen sicheren Kanal über das gemeinsame Geheimnis verständigen. Ein weiterer Nachteil solcher Verfahren ist der Schlüsselaufwand: Für die Kommunikation von n Teilnehmern untereinander muß man $(n^2-n)/2$ Schlüssel erzeugen und verwalten. Natürlich kann man auch gemeinsame Schlüssel für mehrere Kommunikationsbeziehungen verwenden. In diesem Fall geht aber die Möglichkeit verloren, genau zu bestimmen, welcher einzelne Empfänger welche Nachricht lesen darf.

3.2.1.3 Asymmetrische Algorithmen

Bei *asymmetrischen Algorithmen* erzeugt jeder Teilnehmer ein Schlüsselpaar (k_{pub},k_{priv}), bestehend aus einem öffentlichen und einem privaten Schlüssel. Verschlüsselt wird mit dem öffentlichen Schlüssel k_{pub}. Das Chiffrat kann nur mit dem privaten Schlüssel k_{priv} wieder entschlüsselt werden. Es gilt also: $c = A(p,k_{pub})$ und $p = A^{-1}(c,k_{priv})$ (Abbildung 6).

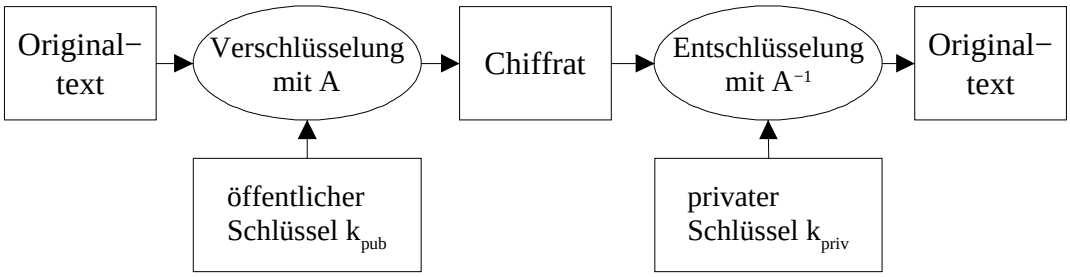


Abbildung 6: Asymmetrische Verschlüsselung

Der Vorteil gegenüber symmetrischen Algorithmen ist, daß der Schlüssel zum Verschlüsseln nicht geheim gehalten werden muß, sondern über offene Kommunikationskanäle verbreitet werden kann. Außerdem müssen für die Kommunikation von n Teilnehmern untereinander nur n Schlüsselpaare (2n Schlüssel) erzeugt und verwaltet werden.

Es besteht aber auch bei asymmetrischer Kryptographie ein weiteres Problem: Die Zuordnung von einem öffentlichen Schlüssel zu einem bestimmten Objekt muß gesichert werden, da sonst ein Angreifer ein eigenes Schlüsselpaar erzeugen könnte und sich in einer Kommunikationsbeziehung den Teilnehmern gegenüber als der jeweilige andere Teilnehmer ausgeben könnte. Diese Angriffsart bezeichnet man als *Man−in−the−middle−attack*. Der öffentliche Schlüssel muß also zweifelsfrei einem Teilnehmer zuzuordnen sein. Man spricht auch davon, daß Vertraulichkeit nicht ohne sichere Authentifizierung möglich ist. Dies wird im Abschnitt 3.2.2.4 näher beleuchtet.

Ein weiterer Nachteil von asymmetrischen Algorithmen ist der gegenüber symmetrischen Algorithmen sehr viel höhere Rechenaufwand, bedingt durch die zum Erreichen gleicher Sicherheit benötigten längeren Schlüssel und aufwendigeren Rechenoperationen. (Ein Schlüssel der Länge 1024 Bit bei asymmetrischen Kryptosystemen entspricht in der Stärke etwa einem Schlüssel der Länge 128 Bit bei symmetrischen Kryptosystemen).

3.2.1.4 Hybridsysteme

Um die jeweiligen Nachteile von symmetrischen und asymmetrischen Algorithmen zu vermeiden, werden in der praktischen Anwendung *Hybridsysteme* verwendet. Dabei wird für jeden Kommunikationsvorgang

ein zufälliger *Sitzungsschlüssel* k_{sym} (*session key*) für einen symmetrischen Algorithmus A_{sym} erzeugt, welcher mithilfe eines asymmetrischen Algorithmus A_{asym} mit Schlüssel k_{asym} verschlüsselt an den Empfänger übermittelt wird. Die eigentliche Nachricht, in der Regel eine sehr viel größere Datenmenge als der Sitzungsschlüssel, wird mit diesem Sitzungsschlüssel und einem symmetrischen Algorithmus verschlüsselt übermittelt.
Ein nicht zu vernachlässigendes Problem bei Hybridsystemen ist das wirklich zufällige Auswählen eines Sitzungsschlüssels. Gute Zufallszahlengeneratoren zu implementieren ist nicht trivial.

3.2.2 Authentifizierungssysteme

Kryptosysteme, welche als Ziel die Integrität, Authentizität und Verbindlichkeit von Nachrichten sicherstellen, nennt man *Authentifizierungssysteme*.

3.2.2.1 Kryptographische Hash−Funktionen

Eine *kryptographische Hash−Funktion oder Einwegfunktion* berechnet aus z.B. einer Nachricht einen vergleichsweise kleinen Datenblock, den sogenannten *Hashwert*. Kryptographische Einwegfunktionen haben die Eigenschaft, daß es unmöglich ist, ohne Kenntnis der Nachricht die zu einem kryptographischen Hashwert gehörige Nachricht zu finden, und daß es sehr unwahrscheinlich ist, daß zwei unterschiedliche Nachrichten den selben kryptographischen Hashwert haben. Man kann den kryptographischen Hashwert einer Nachricht auch als *kryptographische Prüfsumme* bezeichnen.

3.2.2.2 Fingerprint

Hat man zu einer Nachricht m mithilfe einer Einwegfunktion H den Hashwert h ermittelt, so kann man ihn mit einem Schlüssel k und einem Algorithmus A verschlüsseln und erhält den sog. *Fingerprint*. Für den Fingerprint gilt: $f = A(h,k)$ mit $h = H(m)$.
Wird der Fingerprint über einen sicheren Kanal übermittelt, so läßt sich überprüfen, ob ein Inhaber des Schlüssels k die Nachricht verfaßt hat. Es läßt sich mit dem Fingerprint also die Integrität und Authentizität einer Nachricht bezogen auf die Menge der Inhaber des Schlüssels k erreichen. Der Algorithmus A kann hierbei sowohl symmetrisch als auch asymmetrisch sein.

3.2.2.3 Asymmetrische Authentifizierungssysteme

Verwendet man speziell einen asymmetrischen Algorithmus A_{asym} mit dem Schlüsselpaar (k_{pub},k_{priv}) zur Berechnung des Fingerprints der Nachricht m, also $f = A_{asym}(h,k_{priv})$ mit $h = H(m)$, so läßt sich die Integrität und Authentizität der Nachricht m auch bei Übermittlung des Fingerprints über offene Kanäle mithilfe des Schlüssels k_{pub} überprüfen. Stellt man sicher, daß k_{priv} nur einem bestimmten Individuum bekannt ist, so hat man zusätzlich auch das Ziel der Verbindlichkeit erreicht. Man spricht in diesem speziellem Fall oft auch von einer *digitalen Signatur*.
Abbildung 7 stellt schematisch die Erzeugung einer digitalen Signatur beim Sender und die Überprüfung der Signatur beim Empfänger dar. Aus den Originaldaten wird mittels einer Hashfunktion ein Hashwert erzeugt, der mit dem privaten Schlüssel des Senders verschlüsselt wird. Die Signatur wird zusammen mit den Originaldaten übertragen. Der Empfänger berechnet ebenfalls den Hashwert mit der gleichen Hashfunktion und entschlüsselt die Signatur mit dem öffentlichen Schlüssel des Senders, um den vom Sender berechneten Hashwert zu ermitteln. Sind beide Hashwerte gleich, so kann man davon ausgehen, daß der Besitzer des privaten Schlüssels die Nachricht erstellt hat und die Nachricht nicht verfälscht wurde.

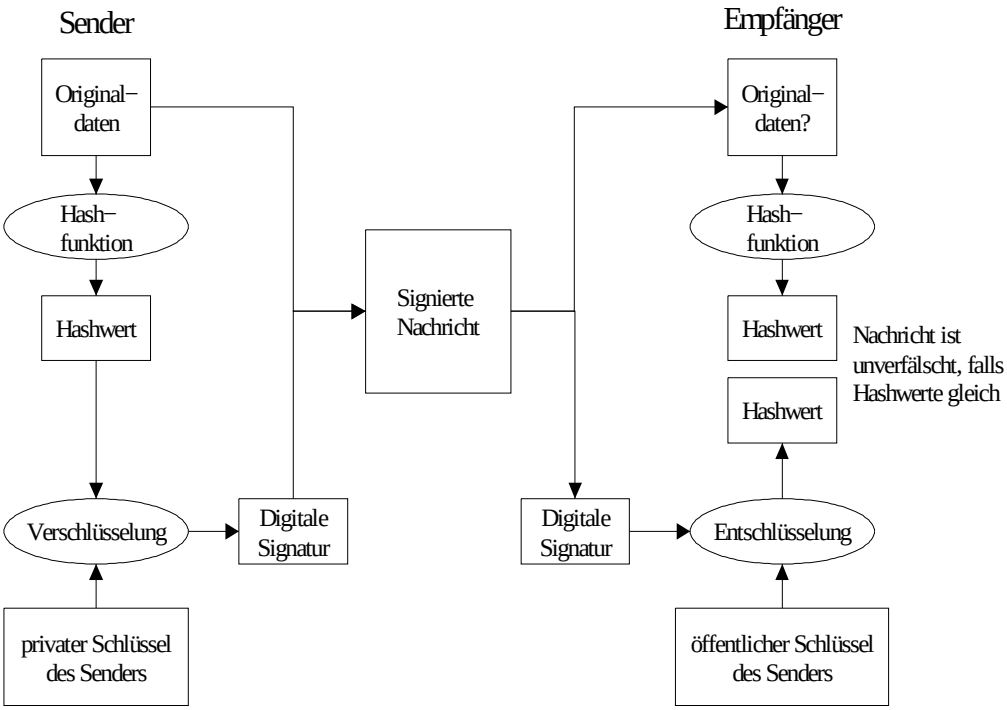


Abbildung 7: Schematische Darstellung der Erzeugung und Überprüfung einer digitalen Signatur

3.2.2.4 Schlüsselaustausch

Möchte man asymmetrische Verschlüsselungsmechanismen einsetzen, so hängt die sichere Benutzung der Verfahren davon ab, ob man sicher weiß, daß ein öffentlicher Schlüssel wirklich zu dem benannten Objekt gehört. Dabei kann ein Objekt eine Person, eine Organisation oder auch ein Server im Internet sein. Die beim Schlüsselaustausch auftretenden Schwierigkeiten sollen anhand von ein paar Beispielen veranschaulicht werden.

3.2.2.5 Persönlicher Kontakt

Am einfachsten läßt sich ein sicherer Schlüsselaustausch bei einem persönlichen Kontakt von sich bereits bekannten, vertrauten Partnern bewerkstelligen. Die Kommunikationspartner tauschen einfach ihre öffentlichen Schlüssel mithilfe geeigneter Medien (z.B. Disketten) aus. Unter der Annahme, daß die Kommunikationspartner ebenfalls an einer vertraulichen Kommunikationsbeziehung interessiert sind, können alle Beteiligten davon ausgehen, daß der öffentliche Schlüssel zur jeweiligen Person gehört. Der Schlüsselaustausch findet auf direktem Wege statt, der jeweilige Kommunikationspartner sorgt in seinem eigenen Interesse dafür, daß niemand in den Besitz seines privaten Schlüssels gelangen kann.

Das Beispiel mit persönlichem Kontakt wird schon etwas komplizierter, wenn sich die beteiligten Parteien nicht alle persönlich kennen, sich also nicht vertraut sind. Üblicherweise stellt man in solchen Situationen die Identität des Gegenübers anhand folgender Indizien fest:

- Man kann sich Personalausweis, Reisepaß o.ä. zeigen lassen.
- Durch weitere Ausweise kann auch z.B. die Zugehörigkeit zu einer bestimmten Organisation überprüft werden.
- Der bisher unbekannte Kommunikationspartner wird durch eine bereits persönlich bekannte Vertrauensperson vorgestellt.
- Die Identität ist durch den konkreten Kontext des Treffens (z.B. Geschäftsbesuch in Firmenräumlichkeiten) klar.
- Der bisher unbekannte Kommunikationspartner weiß bestimmte Informationen, welche seine Identität plausibel machen.

Zu beachten ist, daß man sich schon an dieser Stelle meist unbewußt auf Informationen bezüglich der Identität des Gegenübers verläßt, welche von Dritten bereitgestellt werden bzw. sich implizit aus der Situation ergeben − Identität ergibt sich also aus einem sozialen Kontext. Das Vertrauen in diesen Kontext ist subjektiv und kann getäuscht werden: Ein Reisepaß kann gefälscht sein, die Person benutzt nicht autorisiert bestimmte Räumlichkeiten etc.

3.2.2.6 Elektronischer Schlüsselaustausch

Da ein persönliches Treffen zum Schlüsselaustausch oftmals sehr aufwendig ist, muß man den eigenen Schlüssel oft auch ausschließlich auf elektronischem Wege verbreiten. Es bieten sich vor allem die Internet−Dienste wie WWW und E−Mail als Medium an. Es ist beispielsweise üblich, den öffentlichen PGP−Schlüssel (zu PGP, siehe Abschnitt 3.3.8.2) auf einer WWW−Seite zum Herunterladen anzubieten oder per E−Mail zu verschicken. Es ist auch üblich, PGP−Schlüssel auf zentralen Schlüssel−Servern abzulegen. Solche Schlüssel−Server bestätigen aber, im Gegensatz zu den im folgenden beschriebenen Zertifizierungsstellen, nicht die Echtheit eines Schlüssels bzw. die Identität des Besitzers, sondern erleichtern als zentrale Datenbank lediglich das Auffinden eines Schlüssels.

Publiziert man die öffentlichen Schlüssel ausschließlich mithilfe von Datenkommunikation, so kann man beim ersten Schlüsselaustausch nicht sichergehen, daß nicht ein Dritter den Schlüsselaustausch abfängt und sich gegenüber den Teilnehmern jeweils als der gewünschte Kommunikationspartner ausgibt (*Man−in−the−middle−attack*).
Die Vertrauenswürdigkeit eines öffentlichen Schlüssels kann erhöht werden, wenn man sich auf möglichst viele verschiedene Arten den öffentlichen Schlüssel beschaffen kann. Das Maß der anzuwendenden Sorgfalt hängt davon ab, wie gesichert die individuelle Kommunikationsbeziehung sein soll. Der Anwender entscheidet letzten Endes selbst durch seine eigene Sorgfalt über das Maß an Sicherheit.

Wünschenswert wäre es, zweifelsfrei die korrekte Zugehörigkeit des empfangenen öffentlichen Schlüssels zum jeweiligen Objekt überprüfen zu können. Dies kann mithilfe von zertifizierten Schlüsseln geschehen.

3.2.2.7 Zertifizierte Schlüssel

Bei zertifizierten Schlüsseln ist der jeweilige öffentliche Schlüssel des Kommunikationspartners bereits von einer schon bekannten, vertrauenswürdigen Instanz zertifiziert. Dies läßt sich bewerkstelligen, indem eine solche Instanz − Zertifizierungsstelle genannt[3] − den öffentlichen Schlüssel eines Teilnehmers mit

[3] Eine Zertifizierungsstelle wird in der einschlägigen Literatur auch oft mit CA (vom englischen Begriff »Certification Authority«) abgekürzt.

ihrem eigenen, privaten Schlüssel unterschreibt (signiert). Dies entspricht etwa einem Ausweis, welcher die Identität oder Funktion einer Person bestätigen soll. Bei der Überprüfung eines Ausweises muß Kenntnis über das Aussehen des Ausweises bzw. Vorhandensein von bestimmten Merkmalen (z.B. Stempel) auf dem Ausweis vorliegen. Ebenso muß bei der Überprüfung eines Schlüsselzertifikates der öffentliche Schlüssel der Zertifizierungsstelle bei der Überprüfung bereits vorliegen und dessen Echtheit zweifelsfrei erwiesen sein.

Zunächst erscheint also die Zertifizierung eines Schlüssels durch eine Zertifizierungsstelle nicht viele Vorteile zu haben, da auch hier erst die Identität der Zertifizierungsstelle zweifelsfrei überprüft werden muß. Der entscheidende Nutzen ergibt sich aber daraus, daß, wenn man erst einmal einer Zertifizierungsstelle wirklich vertrauen kann, alle weiteren von dieser Instanz zertifizierten Schlüssel sehr leicht überprüft werden können.
Anzumerken ist noch, daß natürlich auch schon bekannte, vertrauenswürdige Dritte die Echtheit eines öffentlichen Schlüssels bestätigen könnten, ohne als »ordentliche« Zertifizierungsstelle zu gelten. Dieses Prinzip findet z.B. bei PGP seine Anwendung, wo die einzelnen PGP−Nutzer sich gegenseitig »zertifizieren« und ein sogenanntes Netz des Vertrauens (»Web Of Trust«) bilden (siehe auch Abschnitt 3.3.8.2).

3.3 Zertifizierungsinfrastruktur (PKI)

3.3.1 Grundbegriffe

Es sollen erst einige grundlegende Begriffe benannt und erklärt werden[4]. Als *Zertifizierungsinfrastruktur* (*Public Key Infrastructure*, kurz *PKI*) bezeichnet man ein System für die *Zertifizierung* (certification), *Speicherung*, *Bereitstellung* und *Überprüfung* (validation) von kryptographischen Schlüsseln. Obwohl die englische Bezeichnung Public Key Infrastructure für Zertifizierungsinfrastruktur nur die Handhabung von öffentlichen Schlüsseln suggeriert und dies in der Literatur auch oft so eingeschränkt beschrieben wird, ist die Erzeugung und Speicherung von privaten Schlüsselteilen ebenfalls ein Charakteristikum einer Zertifizierungsinfrastruktur.
Ein *Zertifikat* (certificate) ist eine Ansammlung von Informationen, welche vom *Herausgeber* (Issuer) des Zertifikates kryptographisch unterschrieben wurde. Ein *Zertifikatbenutzer* (certificate user) verläßt sich auf die Korrektheit der in einem Zertifikat untergebrachten Information, vertraut also dem Herausgeber des Zertifikates. Der Zertifikatherausgeber wird meist als *Zertifizierungsstelle* (certification authority) bezeichnet.
Die Zertifikate können nach Art der enthaltenen Informationen unterschieden werden:

- Ein *Identitätszertifikat* (identity certificate) bestätigt die Identität eines Objektes, z.B. in Form des Namens oder der E−Mail−Adresse einer Person, und enthält den öffentlichen Schlüssel des Objektes.
- Ein *Attributzertifikat* (attribute certificate oder auch credential certificate) macht eine Aussage, wie z.B.: "Benutzer A hat Zugang zu Rechner R" oder "Kunde B hat einen Kreditrahmen von DM 2000.−" etc.

[4] Da die meiste Literatur auf englisch ist bzw. die meisten Diskussionen zu dem Thema auf englisch geführt werden, ist es oft schwierig, äquivalente, aber auch prägnante deutsche Begriffe zu benutzen. Um den Zusammenhang zu in der Literatur verwendeten Begriffen herzustellen, sind ebenfalls die derzeit üblichen englischen Begriffe in Klammern angegeben. Wird eine Abkürzung verwendet, so liegt immer der englische Begriff zugrunde, um nicht unnötig deutsche Akronyme zu definieren, welche sonst nicht gebräuchlich sind.

Grundlegende Operationen einer PKI sind:

- *Zertifizierung:*
  Prozedur der eindeutigen, unverfälschbaren Zuordnung eines öffentlichen Schlüssels zu einem Objekt oder einer Information.
- *Speicherung:*
  Die sichere Speicherung der Zertifikate und vielleicht auch privater Schlüsselteile.
- *Bereitstellung:*
  Verteilen der Zertifikate an Zertifikatbenutzer.
- *Überprüfung:*
  Prozedur zur Prüfung, ob ein Zertifikat (noch) gültig ist.

Die *Authentifizierung* eines Objektes ist das eigentliche Ziel einer PKI. Wird die Authentifizierung ausschließlich mithilfe der PKI−eigenen Mechanismen erreicht, so spricht man von *in−band−authentication*. Dagegen wird die Authentifizierung mittels nicht PKI−eigener Mechanismen, z.B. über Telefon, als *out−of−band−authentication* bezeichnet. Ein wichtiges Entwurfsziel für PKI−Architekturen ist die Minimierung der benötigten out−of−band−authentication. Eine PKI, welche eine wirklich ernstzunehmende Authentifizierung bieten soll, kommt nie ganz ohne PKI−fremde Mechanismen aus, da während der PKI−Initialisierung immer irgendwelche Informationen über einen sicheren Kanal transportiert werden müssen (für nähere Betrachtungen hierzu siehe auch Abschnitt 3.2.2.4).

3.3.2 Zertifizierungsstellen

Eine *Zertifizierungsstelle* (*certification authority*, kurz *CA*) stellt Zertifikate aus, d.h. sie bestätigt durch das Signieren mit ihrem privaten Schlüssel die Zuordnung eines öffentlichen Schlüssels zu einem Objekt (Identitätszertifikat) oder die Richtigkeit der im Zertifikat gemachten Aussage (Attributzertifikat). Zum Erreichen dieses Ziels muß die Zertifizierungsstelle geeignete Verfahren etablieren, um die Wahrhaftigkeit der Zertifikataussagen hinreichend zu überprüfen. Das Maß an nötiger Sorgfalt bei der Überprüfung der Zertifikataussagen hängt dabei stark vom Verwendungszweck der ausgestellten Zertifikate ab.
Für Identitätszertifikate braucht die *Zertifizierungsstelle* nur den öffentlichen Schlüssel eines Teilnehmers, der private Schlüssel ist bei der Benutzung ausschließlich dem Teilnehmer selbst bekannt. Einen weitaus detaillierteren Überblick über Aufgaben, Probleme und Verfahren beim Betrieb einer CA gibt Abschnitt 3.4, die Beschreibung einer im Rahmen der Arbeit implementierten Testzertifizierungsstelle findet sich in Kapitel 6.

3.3.3 Schlüsselvergabestellen (Trust Center)

Eine *Schlüsselvergabestelle* (*Trust Center*) übernimmt neben einer Überprüfung der Zuordnung von Identität eines zertifizierten Objekts zu einem öffentlichen Schlüssel auch die Erzeugung der Schlüsselpaare an sich und speichert den privaten Schlüsselanteil zur Sicherung gegen Datenverlust durch Verlust des privaten Schlüssels. Eine Schlüsselvergabestelle, welche private Schlüssel speichert, muß sehr große technische und organisatorische Anstrengungen unternehmen, um einen Mißbrauch der privaten Schlüssel zu verhindern.
Die Benutzung eines Trust Centers in der Zertifizierungsinfrastruktur ist sehr umstritten, da zum einen besonders staatliche Behörden immer wieder mit Hinweis auf die Strafverfolgung Zugang zu privaten

Schlüsseln fordern, die Mißbrauchsmöglichkeit durch den Staat und Dritte aber enorm hoch ist. So manche PKI−Literatur bezeichnet eine Zertifizierungsinfrastruktur, welche Trust−Center enthält, auch als *broken−by−design*[5].

3.3.4 Topologie der Zertifizierungsstellen

Offensichtlich ist es unpraktisch, nur eine Zertifizierungsstelle für eine globale Zertifizierungsinfrastruktur zu haben. Daher sehen die meisten PKI−Entwürfe vor, daß eine Zertifizierungsstelle weitere Zertifizierungsstellen zertifiziert, d.h. eine Zertifizierungsstelle weist ihre Benutzer an, den von der anderen Zertifizierungsstelle ausgestellten Zertifikaten zu vertrauen. Oft werden dabei auch *Zertifizierungsstellen−Zertifikate* (CA certificate) und *Benutzer−Zertifikate* (user certificate) unterschieden.

Möchte ein Benutzer A der Zertifizierungsstelle P mit dem Benutzer B der Zertifizierungsstelle Q sicher kommunizieren, so muß er sich entweder über einen sicheren Kanal das selbstsignierte Zertifikat Z_Q von Q besorgen und damit das Benutzer−Zertifikat Z_{QB} von B direkt prüfen, oder die Zertifizierungsstelle P stellt ein sog. Cross−Zertifikat Z_{PQ} für Q aus. Im zweiten Fall muß der Benutzer A vor der Kommunikation mit B wie gehabt nur das Zertifikat Z_P seiner eigenen Zertifizierungsstelle P kennen. Die Überprüfung des Benutzer−Zertifikats Z_{QB} von B erfolgt dann, indem das von P für Q ausgestellte Zertifikat Z_{PQ} mit dem CA−Zertifikat Z_P von P geprüft wird und dann das Benutzer−Zertifikat Z_{QB} von B mit dem Zertifikat Z_Q der Zertifizierungsstelle Q geprüft wird. Der entscheidende Vorteil ist, daß alle Benutzer von P nur durch das zusätzliche Zertifikat Z_{PQ} von P für Q alle Benutzerzertifikate von Q prüfen können (siehe Abbildung 8).

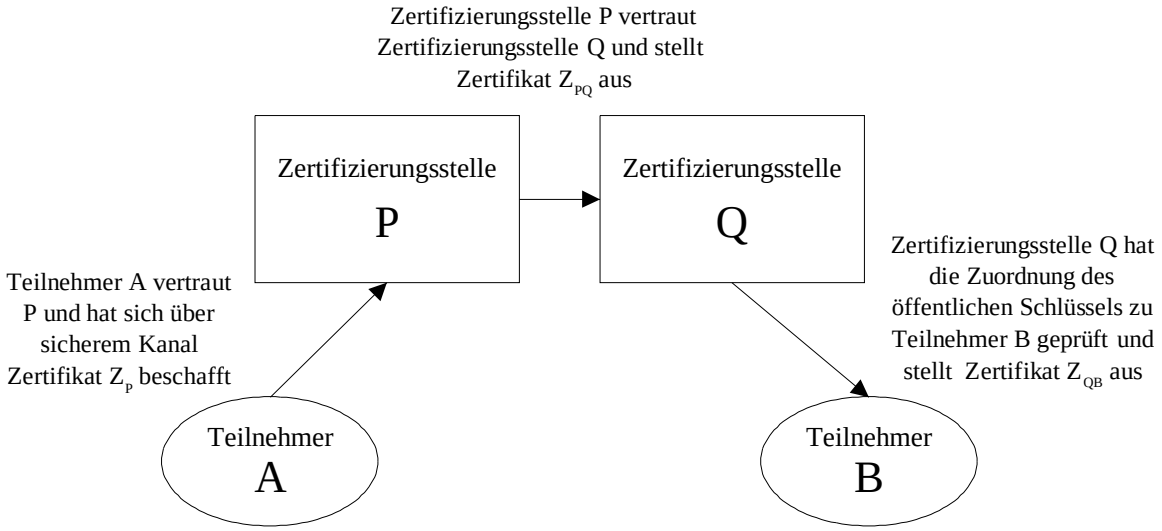


Abbildung 8: Zertifikatpfad mit zwei cross−zertifizierten Zertifizierungsstellen

Eine solche Aneinanderreihung von Zertifikaten bei der Überprüfung nennt man *Zertifizierungspfad* (certification path). Im allgemeinen können solche Zertifizierungspfade über beliebig viele Zertifizierungsstellen führen, wobei die Zertifikate sämtlicher Zertifizierungsstellen überprüft werden müssen. Eine solche Überprüfung nennt man *Überprüfung des Zertifizierungspfads* (certification path

[5] broken−by−design: Fehlerhafte Konstruktion

validation). Die *Zertifizierungspfadlänge* ist die Anzahl der zu überprüfenden Zertifizierungsstellen entlang des Zertifizierungspfades. Die Form der Verkettung mehrerer Zertifizierungsstellen einer PKI wird als *Topologie* (CA arrangement) bezeichnet. Manche Zertifizierungsinfrastrukturen benutzen streng hierarchische, baumförmige Topologien. Andere plazieren alle Zertifizierungsstellen auf einer gemeinsamen Ebene, wiederum andere geben überhaupt keine klar strukturierte Topologie vor. Sämtliche Mischformen von Topologien sind denkbar, aber technisch nicht leicht umzusetzen.

Je nach Topologie der PKI variiert die typische bzw. maximal mögliche Pfadlänge. Je länger die Zertifizierungspfadlänge ist, desto höher ist das Sicherheitsrisiko bei der Überprüfung eines Zertifikates. Ein weiteres Bewertungsmaß für eine PKI−Topologie ist die mögliche Skalierbarkeit. Eine PKI, in welcher sich ausschließlich einzelne Benutzer in Punkt−zu−Punkt−Kommunikation gegenseitig zertifizieren, ist global gesehen nur schlecht skalierbar. Problematisch ist auch die erforderliche Transitivität einer Vertrauensrelation zwischen Kommunikationspartnern bzw. Zertifizierungsstellen. Wenn A dem Benutzer B vertraut und dieser dem Benutzer C, so heißt das noch lange nicht, daß A wirklich dem Benutzer C vertrauen kann. Vor allem fehlt auch eine allgemein gültige Definition des Begriffs Vertrauen an sich (siehe auch [Ger98b]).
In einer streng hierarchischen PKI vertrauen alle Benutzer einer Wurzel−CA (root CA oder top−level CA) in jeder Hinsicht, was auf keinen Fall den verschiedenen Erfordernissen in einem globalen Kommunikationssystem gerecht werden kann.

3.3.5 Beziehung zwischen Entitäten

Man kann die Rollen an der PKI beteiligten Entitäten einteilen in:

- Zertifizierungsstellen (CA), Herausgeber (Unterzeichner) von Zertifikaten
- Zertifizierte Objekte (subjects), z.B. Personen, Server etc.
- PKI−Benutzer (users)

Eine PKI kann z.B. alle Rollen auf verschiedene Entitäten aufteilen, oder eine Entität ist CA, Objekt und Benutzer zugleich. Das wirft die Frage auf, wie stark die Vertrauensbeziehung zwischen verschiedenen Entitäten für das Funktionieren einer bestimmten PKI sein muß. Man könnte z.B. fordern, daß eine CA immer der Arbeitgeber einer zertifizierten Person ist, oder auch eine öffentliche Meldestelle, wie das Einwohnermeldeamt in Deutschland. Oder Personen (zertifizierte Objekte und Benutzer) haben sich mindestens einmal persönlich getroffen. Auch könnte dem Benutzer eine bestimmte CA vorgeschrieben werden, der er für alle Anwendungen vertrauen muß. Oder er hat die freie Wahl zwischen vielen Zertifizierungsstellen und wählt diejenige aus, welche seine eigenen Erfordernisse am besten erfüllt. Manche Zertifikatinfrastrukturen sehen auch eine Verfeinerung der Vertrauensrelationen vor, z.B. kann man wählen, daß von einer bestimmten CA nur Zertifikate für E−Mail akzeptiert werden, die einer bestimmten Adressen−Domäne angehören.

Schließlich stellt man fest, daß eine Vertrauensrelation nicht ohne einen bestimmten technischen, sozialen und juristischen Zweck und Kontext definiert werden kann. Schon allein die Identität eines Objektes ergibt sich nur aus einem sozialen Zusammenhang. Mögliche Verbindlichkeiten aus Kommunikationsbeziehungen (z.B. eine Online−Bestellung, geschäftliche Absprache etc.) ergeben sich auch nur in einem definierten juristischen Rahmen. Die technische Definition einer PKI ist alleine für die Erfüllung des Zwecks einer PKI nicht ausreichend. Und schließlich entscheidet immer der Benutzer einer PKI in Abhängigkeit des Anwendungsfalles, wie weit er Vertrauen in eine CA oder ein Zertifikat hat.

3.3.6 Zertifikatprüfung (certificate validation)

Eine wesentliche Operation in einer PKI ist die *Zertifikatprüfung* (certificate validation). Das Ziel der Zertifikatprüfung ist es herauszufinden, ob ein Zertifikat zu einem bestimmten Zeitpunkt benutzt werden darf oder durfte.

Es gibt im wesentlichen zwei Arten der Überprüfung:

- *Online−Prüfung* (online validation): Der Benutzer fragt jedesmal die CA direkt, ob ein von ihr ausgestelltes Zertifikat noch gültig ist.
- *Offline−Prüfung* (offline validation): Im Zertifikat selbst oder auf zusätzlich ausgegebenen Listenzertifikaten werden Anfang und Ende der Gültigkeitsdauer des benutzten Zertifikates vermerkt. Außerhalb dieses Gültigkeitszeitraums darf das Zertifikat nicht benutzt werden.

Eine PKI kann beide Formen der Zertifikatprüfung anbieten, und beide Formen haben ihre Vor− und Nachteile. Die Online−Prüfung ergibt sofort ein aktuelles Prüfungsergebnis, erzeugt aber sehr viel Netz− und Rechner−Last bei der CA. Eine Variante der Online−Prüfung ist ein Vorschlag für die Verwendung bei Online−Bezahlung (Carl Ellison, Cybercash): Es werden nur Zertifikate mit sehr kurzer Gültigkeitsdauer im Minutenbereich ausgestellt, welche nur für einzelne oder wenige Transaktionen verwendet werden können. Dies erfordert einen Kommunikationsvorgang mit der CA bei jeder Transaktion, kann aber in bestimmten Situationen weniger Last erzeugen als die Online−Überprüfung von Zertifikaten mit längerer Laufzeit. Bei der Online−Prüfung kommt als weiterer Nachteil hinzu, daß mit einer Verkehrsanalyse der Prüfungsvorgänge sehr leicht Kommunikationsprofile der Teilnehmer erstellt werden können. Die Offline−Prüfung erzeugt die meiste Last auf Seite des Benutzers, ist jedoch u.U. fehlerhaft, da keine wirklich aktuellen Informationen über den Zustand eines Zertifikates vorliegen.

3.3.6.1 Zertifikatwiderruf

Beim *Zertifikatwiderruf* (certificate revocation) wird die Gültigkeit der Zertifikataussage zurückgenommen. Mehrere Umstände können einen Zertifikatwiderruf notwendig machen:

- *Zertifikataussage nicht mehr gültig*
  Ein Zertifikat muß ungültig werden, wenn sich die Identität des zertifizierten Objektes oder die Aussage des Zertifikates in irgend einer Weise ändert und davon die Zertifizierungsregeln verletzt werden. Wenn z.B. ein Mitarbeiter eine Firma verläßt, so ist das Zertifikat zu widerrufen, welches ihn als Angehörigen dieser Firma ausweist. In der Regel wird in diesem Fall nicht der Zertifikatsinhaber das eigene Zertifikat widerrufen, sondern ein Repräsentant der betreffenden Organisation.
- *Verlust des privaten Schlüssels oder Zertifikat kompromittiert*
  Es könnte sein, daß der Anwender seinen privaten Schlüssel nicht sorgfältig genug gesichert hat und beispielsweise durch Datenverlust oder ein vergessenes Kennwort seine privaten Schlüsseldaten nicht mehr verfügbar sind. Ebenso könnte durch einen Angriff auf die privaten Schlüsseldaten das Zertifikat kompromittiert worden sein. Für diesen Fall muß der Zertifikatinhaber sein eigenes Zertifikat widerrufen können.

- *CA−Zertifikat kompromittiert*
  Auch das Zertifikat der Zertifizierungsstelle könnte kompromittiert worden sein. Dieser Fall stellt ein besonderes Problem dar, da die CA natürlich nicht mehr selbst einen Widerruf unterzeichnen kann. Dies kann nur eine evtl. übergeordnete CA, oder es ist ein kompletter Austausch der CA−Zertifikate bei allen Anwendern vonnöten.

Wenn die Zertifikatprüfung online erfolgt, so ist der Zertifikatwiderruf trivial, da die CA einfach bei der Zertifikatprüfung ’Zertifikat ungültig’ als Ergebnis liefern muß. Verzögerungszeiten zwischen eigentlichem Zertifikatwiderruf und Kenntnisnahme des Widerrufs beim Zertifikatbenutzer reduzieren sich auf Paketlaufzeiten des Transportsystems, sind also zu vernachlässigen.

3.3.6.2 Zertifikatwiderruflisten

Der am häufigsten verwendete Mechanismus bei der Offline−Prüfung ist die Ausgabe einer sogenannten *Zertifikatwiderrufliste* (*Certificate Revocation List*, kurz *CRL*) durch die Zertifizierungsstelle. Eine CRL ist eine Liste mit den bei einer CA widerrufenen Zertifikaten, meist auch mit dem Zeitpunkt des Zertifikatwiderrufs, welche von der CA herausgegeben und signiert wird. Meist enthalten CRL auch noch eine Gültigkeitsdauer und eine Adresse für das Abholen einer aktuelleren CRL. Der PKI−Benutzer muß bei der Zertifikatprüfung möglichst die aktuellste CRL besitzen und prüfen, ob das Zertifikat in der CRL enthalten ist.

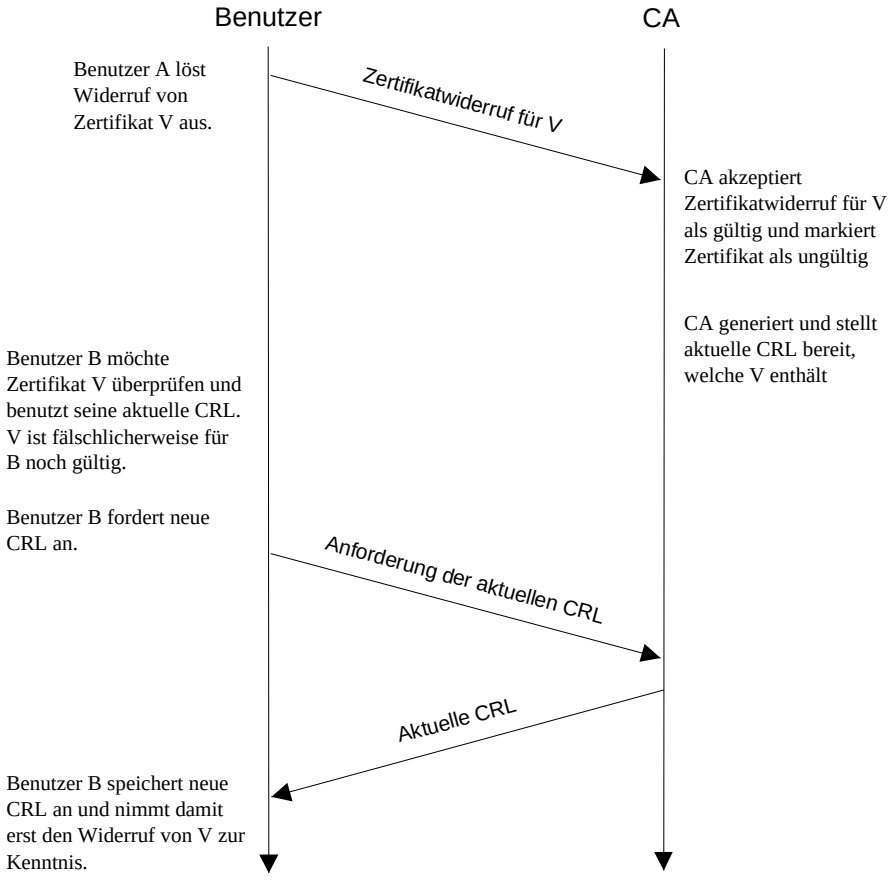


Abbildung 9: Zeitlicher Ablauf des Zertifikatwiderrufs

Im Gegensatz zur Online−Prüfung ergibt sich bei der Verwendung von Zertifikatwiderruflisten eine zeitliche Verzögerung zwischen dem Zeitpunkt, zu dem das Zertifikat bei der CA widerrufen wurde, und der Kenntnisnahme des Zertifikatwiderrufs beim PKI−Benutzer (CRL time granularity problem). Der zeitliche Ablauf beim Zertifikatwiderruf und der Zertifikatprüfung ist in Abbildung 9 dargestellt.
Die beim Zertifikatwiderruf auftretende Verzögerungszeit teilt sich in zwei Zeitspannen auf:

- Die Zeitspanne von der Kenntnisnahme der CA über den Zertifikatwiderruf bis zur Bereitstellung der nächsten CRL.
- Die Zeitspanne von der Bereitstellung der aktuellen CRL bis zur Übermittlung der CRL an den Benutzer.

Der Benutzer müßte sich, um jegliche Inkonsistenz zu vermeiden, vor jeder Zertifikatbenutzung die aktuelle CRL beschaffen, um diese Zeitverzögerung zu eliminieren, was wieder einer Online−Prüfung gleich käme.
Ein weiterer wichtiger Aspekt ist die Größe der CRL. Möglicherweise stellt eine CA sehr viele Zertifikate aus, wobei je nach Widerrufsrate die CRL schnell sehr lang werden kann. Sehr große Zertifikatwiderruflisten sind aber u.U. für den Benutzer schwierig zu handhaben, sei es wegen nur unzureichender Netzübertragungsleistung, mangelndem Speicherplatz oder mangelnder Rechenleistung.
Die o.g. Schwierigkeiten mit Zertifikatwiderruflisten führten zu weiteren Verfeinerungen des CRL−Konzepts.

3.3.6.2.1 Partitionierung von Zertifikatwiderruflisten

Eine CA könnte verschiedene Zertifikatwiderruflisten für verschiedene Gründe eines Zertifikatwiderrufs herausgeben. Z.B. ist das Ausscheiden eines Mitarbeiters aus einer Organisation und der damit verbundene Zertifikatwiderruf u.U. weniger zeitkritisch als echte Sicherheitsprobleme wie z.B. ein kompromittierter privater Schlüssel einer weiteren CA. Eine CA kann auch mehrere verschiedene Zertifikattypen ausstellen und dementsprechend für jeden Zertifikattyp eine eigene CRL herausgeben. Verwendet ein Benutzer nur Zertifikate des Typs A, so interessiert ihn die CRL der ebenfalls von der selben CA herausgegebenen Zertifikate des Typs B gar nicht. Die Gesamtgröße aller CRLs bleibt natürlich gleich, aber in beiden Fällen muß der Benutzer nur kleinere Zertifikatwiderruflisten behandeln, um das gerade benutzte Zertifikat zu prüfen.

3.3.6.2.2 Differenz−Zertifikatwiderruflisten (Delta−CRL)

Anstatt jedesmal die ganze CRL zu übermitteln, kann eine CA auch immer nur die Differenzen zwischen verschiedenen Versionen der Zertifikatwiderruflisten herausgeben. Diese Differenzen können öfter übertragen werden, die zeitliche Verzögerung zwischen eigentlichem Zertifikatwiderruf und Kenntnisnahme beim Benutzer wird kleiner. Natürlich setzt dies voraus, daß bei der ersten Benutzung die komplette, zu diesem Zeitpunkt aktuelle CRL zum Benutzer übermittelt wurde und der Benutzer auf zuverlässige Art und Weise eine komplette CRL auf seiner Seite pflegt.

3.3.7 Grundsätzliche Probleme

Angenommen, es gäbe eine technisch optimale PKI, so bleiben doch noch weitere Probleme, welche sich nicht technisch innerhalb einer PKI lösen lassen (vergleiche auch [Ger98],[Gar97]).

3.3.7.1 Umgang mit privaten Schlüsseln

Es ist klar, daß ein privater Schlüssel an sich keine Identität ist. Mit einem privaten Schlüssel erzeugte digitale Signaturen sind nur ein Hilfsmittel, eine Identität zu beweisen, dessen Zuverlässigkeit aber auf der absolut sicheren Handhabung des privaten Schlüssels beruht. Bei einer CA mögen die CA−Betreuer als Experten sich über die Bedeutung ja noch im klaren sein und entsprechende Maßnahmen zur Sicherung der privaten Schlüsseldaten ergreifen. Aber davon auszugehen, daß jeder an der PKI beteiligte Anwender überhaupt ausreichendes Wissen besitzt, sich vor den Gefahren des Mißbrauchs seines privaten Schlüssels zu schützen, ist bereits illusorisch. Die Teilnehmerendsysteme, heutzutage handelsübliche Personal Computer mit oftmals gar nicht oder nur unzureichend gesicherten Betriebssystemen, bieten sehr vielfältige Möglichkeiten des Angriffs auf gespeicherte Dateien und Tastatureingaben (siehe Abschnitt 2.2.5). Niemand hat auf diese Frage der Sicherheit der Endsysteme eine schlüssige Antwort. VeriSign sagt z.B. hierzu in den selbst vergebenen Zertifizierungsregeln [Ver98a]:

> *Each certificate applicant shall securely generate his, her or its own private key, using a trustworthy system, and take necessary precautions to prevent its compromise, loss, disclosure, modification, or unauthorized use.*

Diese Aussage ist technisch gesehen wenig hilfreich, sondern dient lediglich dazu, alle Haftung für den Mißbrauch von privaten Schlüsseln auszuschließen und alle Verantwortung dem Benutzer aufzuerlegen, was man aus Sicht einer CA auch verstehen kann.

Desweiteren ist es oftmals schwer, die zentrale Bedeutung des privaten Schlüssels dem Anwender verständlich zu machen. Allzu oft kommt es vor, daß zwei Leute sich eine Mail−Adresse teilen und die Nutzung gemeinsamer privater Schlüssel vereinbaren. Diese Form des "Mißbrauchs" kann man nicht kontrollieren. Der in der praktischen Anwendung von Verschlüsselung auftretenden Ignoranz mancher Anwender bezüglich Sicherheit läßt sich nur mit ausreichender Schulung und ggf. strikten Verfahrensanweisungen begegnen, wobei es sich oft als hilfreich erweist, Beispiele aus dem täglichen Leben (z.B. Verwendung herkömmlicher mechanischer Schlüssel in Schließsystemen etc.) auf die Kryptographie abzubilden. Das Sicherheitsempfinden der Menschen ist sehr subjektiv und vor allem von eigenen Erfahrungen und Erlebnissen im näheren Umfeld geprägt. Beim Einsatz von Kryptographie liegen den meisten Menschen schlicht und einfach solche Erfahrungen noch nicht vor.

3.3.7.2 Namen sind keine Personen

Ein Name bestimmt nicht hinreichend eine Identität. Zum einen muß der Name eindeutig werden. Bei sehr häufig vorkommenden Namen (z.B. in Deutschland "Peter Schmidt") kann dies innerhalb einer Organisation schon schwierig werden. Behelfen kann man sich an dieser Stelle mit dem Hinzufügen von zusätzlichen, eindeutigen Namensteilen, wie z.B. einer definitionsgemäß eindeutigen E−Mail−Adresse (vergleiche [RFC2377]).
Aber auch ein global eindeutig definierter Name kann nicht eine Identität bescheinigen. Die Identität ergibt sich nur aus dem sozialen Kontext. Selbst wenn der Mensch von Geburt an eine eindeutige Seriennummer hätte (z.B. genetischer Fingerabdruck als eindeutiges Identifikationsmerkmal), so müßte man, um eine Identität zu definieren, einer solchen Kennung erst eine Semantik zuweisen, welche aber wiederum kontextabhängig ist.
Eine CA muß sich also idealerweise Regeln geben, die definieren, was ein von ihr in einem Zertifikat vergebener Name bedeutet, und sicherstellen, daß beim Ausstellen von Zertifikaten die Richtigkeit dieser

Bedeutung immer überprüft wird. Der PKI−Benutzer muß die Zertifizierungsregeln zur Kenntnis nehmen, was ihm auch wieder sehr viel Verantwortung aufbürdet. Aber selbst das Definieren solcher selbstauferlegter Regeln ist alles andere als trivial: Z.B. ist bei multinationalen Organisationen und Personen mit hoher Mobilität eine für den Zertifikatverwendungszweck sinnvolle Ortsangabe nicht immer eindeutig zu bestimmen. Die Definition des X.509−Attributes C (Land, Staat − countryname) in [RFC2256] besagt:

> *5.7. c*
> *This attribute contains a two−letter ISO 3166 country code (countryName).*

Bei Personen ergeben sich schon mehrere Möglichkeiten: Das Geburtsland, das Land in welchem eine Staatsangehörigkeit besteht[6], das Land des gemeldeten Wohnorts oder des häufigsten Aufenthaltsortes? Ein anderes Beispiel ist die Semantik des X.509−Attributes CN (der allgemeine Name − Common Name). Die auf LDAP bezogene Definition in [RFC2256] besagt:

> *5.4. cn*
> *This is the X.500 commonName attribute, which contains a name of an object. If the object corresponds to a person, it is typically the person's full name.*

Ist mit Personenname der Name in der Geburtsurkunde, der üblicherweise verwendete Vorname und Nachname, alle Vornamen und Nachname oder ein wohlbekannter Name (z.B. Künstlername) gemeint? Und welcher Name ist bei einem zertifizierten Server einzutragen?

Ein weiteres Problem sind Namen, welche eine Rolle und kein Individuum beschreiben. Z.B. gelang es Kevin McCurley, sich von VeriSign Zertifikate für die E−Mail−Adressen *root@localhost*, *postmaster@localhost*, *daemon@localhost* und *bin@localhost* ausstellen zu lassen. Ein unbedarfter Benutzer eines UNIX−Systems würde annehmen, daß Nachrichten, welche mit einem solchen Zertifikat signiert sind, tatsächlich von einem System−Administrator kommen und z.B. Anfragen nach dem eigenen Kennwort beantworten. Nun könnte man argumentieren, daß man einfach definitionsgemäß keine Zertifikate für solche E−Mail−Adressen ausgeben sollte. Allerdings wäre es für einen Benutzer vielleicht hilfreich, eine verschlüsselte Mail an *root@localhost* schreiben zu können, weil ihm der persönliche Name des Administrators unbekannt ist.

3.3.7.3 Gruppenkommunikation

Auch kann sich hinter jeder Mail−Adresse eine ganze Gruppe von Leuten verbergen. Es gibt mehrere Möglichkeiten, für eine Gruppe Nachrichten zu verschlüsseln:

- Der Sender beschafft sich die Liste der Gruppenmitglieder und von jedem Mitglied der Gruppe das Zertifikat und schickt an jedes Mitglied eine getrennt verschlüsselte Mail. Dieser Fall ist sicher, aber für den Teilnehmer sehr aufwendig, und u.U. sollen die Mitglieder der Gruppe aus Datenschutzgründen gar nicht öffentlich bekannt sein. Z.B. sollte ein Kunde durchaus eine verschlüsselte Nachricht an die Mail−Adresse *support@firma.my* schicken können. Wer die Mitarbeiter in der Support−Abteilung wirklich sind, möchte man aber vielleicht nicht nach außen dringen lassen, um einen Angriffspunkt für »Social Hacking« zu vermeiden.

[6] Es gibt doppelte Staatsbürgerschaften!

- Ein Zertifikat für einen Mail−Verteiler wird ausgestellt, und ein Teilnehmer schickt an die Gruppe eine mit dem Gruppenschlüssel verschlüsselte Nachricht. Alle Mitglieder der Gruppe haben den privaten Schlüssel und können die Nachricht lesen, sie dürfen den privaten Gruppenschlüssel aber nicht zum Signieren verwenden. Die Gruppe wird aber Zu− und Abgänge bei den Mitgliedern haben, was jedesmal eine neue Erzeugung des privaten Schlüssels nach sich ziehen würde, um wirklich sicherzustellen, daß nur die Mitglieder der Gruppe die Nachricht lesen können.
- Ein Zertifikat für einen Mail−Verteiler wird ausgestellt, und ein Teilnehmer schickt an die Gruppe eine mit dem Gruppenschlüssel verschlüsselte Nachricht. Der Mail−Server kennt die einzelnen Mitglieder der Gruppe und deren Zertifikate. Auf dem Mail−Server wird die Mail entschlüsselt und für jedes Gruppenmitglied getrennt wieder verschlüsselt. Das Problem ist hier die automatische Handhabung eines privaten Schlüssels ohne menschliche Interaktion (wie z.B. Eingabe einer Passphrase). Eine solche Vorgehensweise erfordert sehr viel Aufwand bei der Sicherung des Systems, und dem Administrator des Mail−Servers muß voll vertraut werden.

3.3.7.4 Datenschutz

Es stellt sich immer wieder die Frage, welche Informationen in einem Zertifikat untergebracht werden sollen. Z.B. sind X.509−Zertifikate eine Art Mini−Dossier, in welchem u.U. Informationen stehen, welche nur für eine bestimmte Anwendung benötigt werden, aber immer komplett an alle übermittelt werden, die nach dem Zertifikat fragen. Zwei einfache Beispiele sollen diese Problematik illustrieren:

- In den meisten Zertifikaten ist die E−Mail−Adresse enthalten, welche aber z.B. für den Zugriff auf einen WWW−Server in der Regel nicht benötigt wird. Je nach Konfiguration sendet aber z.B. der Netscape Navigator das Benutzer−Zertifikat ohne Rückfrage an jeden WWW−Server, der danach fragt (siehe auch Abschnitt 5.1.1). Eine willkommene und durch den angewählten WWW−Inhalt zielgruppenorientierte Datenquelle für Versender von Werbenachrichten (spam mails).
- Möchte z.B. eine Versicherung mit ihren Kunden eine Kommunikationsbeziehung einrichten, so wäre es für diese Anwendung nützlich, die Versichertennummer mit in dem Zertifikat unterzubringen. Meist gilt diese Versichertennummer bei Telefonaten mit der Versicherung bereits als eine Art Authentifizierung des Kunden, sollte also nicht jedem Betreiber eines obskuren WWW−Servers zugänglich gemacht werden.

Daraus folgt, daß der Benutzer für jede spezielle Kommunikationsbeziehung und jede Anwendung ein eigenes Zertifikat besitzen muß. Die weltweit operierende Zertifizierungsstelle *Thawte* hat z.B. unter dem griffigen Namen »Strong Extranet« einen entsprechenden Entwurf veröffentlicht [Thaw98a]. Dies setzt aber wieder eine a−priori Abstimmung der Teilnehmer über die Zertifikatattribute voraus (siehe auch Abschnitt 3.3.8.1.1), ist also für spontane, sichere Kommunikation ohne Vorbereitung unpraktikabel. Außerdem ist es fraglich, ob der Anwender nicht mit mehr Zertifikaten auch mehr Bedienerfehler macht.

Für Zugriffe auf Server könnte man auch völlig anonymisierte Zertifikate ausstellen mit Benutzer−IDs, deren Bedeutung nur dem Zertifikatherausgeber bekannt ist. Präsentiert ein PKI−Teilnehmer ein Zertifikat, so fragt der PKI−Benutzer eine Datenbank mithilfe der enthaltenen Benutzer−ID ab, um weitere Attribute zu bekommen. Für geschlossene Anwendungen mit Möglichkeiten der Absprache vor der Kommunikation ist dies der vertraulichste Weg, aber vergleichsweise aufwendig und inflexibel, da auch die Datenbankabfrage wieder hinreichend gesichert werden muß.

3.3.8 Beispiele

Es werden nun verschiedene PKI−Ansätze zum Vergleich kurz vorgestellt. Eine detaillierte Übersicht über verschiedene PKI und ein Ansatz zum systematischen Vergleich von PKI findet sich in [Bra97]. Folgende PKI sind derzeit von praktischem Interesse:

- X.509 als ein System mit strengen strukturellen Vorgaben, welches zentrale Zertifizierungsstellen fördert.
- Web−Of−Trust (PGP) als stark auf den einzelnen Benutzer konzentriertes System, in welchem keinerlei strukturelle Vorgaben gemacht werden.
- Secure DNS als ein Beispiel, wie ein schon bestehender Internet−Dienst mit Erweiterungen versehen werden kann, um eine PKI aufzubauen.
- Royal−Holloway (Trusted Third Parties) als Beispiel für eine PKI mit Speicherung privater Schlüsselteile bei einer Schlüsselvergabestelle.

Weitere Ansätze existieren, haben aber derzeit nur geringe praktische Bedeutung.

3.3.8.1 X.509

X.509 ist das Authentifizierungskonzept für den ISO−Verzeichnisdienst X.500 und ist wie dieser hierarchisch verzeichnisorientiert. Es gibt zwei Authentifizierungsarten:

- *Simple Authentication* (einfache Authentifizierung) mittels Kennwörtern und ohne Verschlüsselung: Dieser Fall interessiert nur beim schwach geschützten Zugriff, z.B. dem Lesezugriff auf Verzeichnisdienste.
- *Strong Authentication* (starke Authentifizierung) mittels kryptographischer Authentifizierung, also mithilfe von Schlüsselzertifikaten.

Die Benutzung von kryptographischen Zertifikaten nach dem X.509−Standard ist in vielen kryptographischen Protokollen verankert, z.B. SSL (Abschnitt 3.5.3) und S/MIME (Abschnitt 3.5.4). X.509 liegt inzwischen in der Version X.509v3 vor, welche in Abschnitt 3.3.8.1.1 näher beschrieben wird.
X.500 definiert für beliebige Objekte (Länder, Organisationen, Orte, Personen, Gruppen, etc.) hierarchisch strukturierte Verzeichniseinträge, welche ebenfalls auf hierarchisch strukturierten Verzeichnisdienst−Servern abgelegt sind (näheres zu X.500 bzw. dem leichtgewichtigeren Derivat LDAP siehe [Wäch98], [IBM4986]). Dazu bekommt jedes Objekt einen global eindeutigen, hierarchisch strukturierten Namen (Distinguished Name − kurz DN), welcher relativ zur Position im Verzeichnisbaum festgelegt wird.
Ebenso definierte X.509 ursprünglich für die weltweite Zertifizierungsinfrastruktur eine ausschließlich baumartige Hierarchie von Zertifizierungsstellen. Desweiteren muß auch in X.509−Zertifikaten jedem Objekt ein global eindeutiger Name zugewiesen werden. An der Wurzel des Baumes steht eine (weltweite) Root−CA, die untergeordnete Zertifizierungsstellen zertifiziert, welche ihrerseits wiederum kleinere Teilbereiche von Objekten (Regional CA, Personen, Server, etc.) zertifizieren (Abbildung 10).

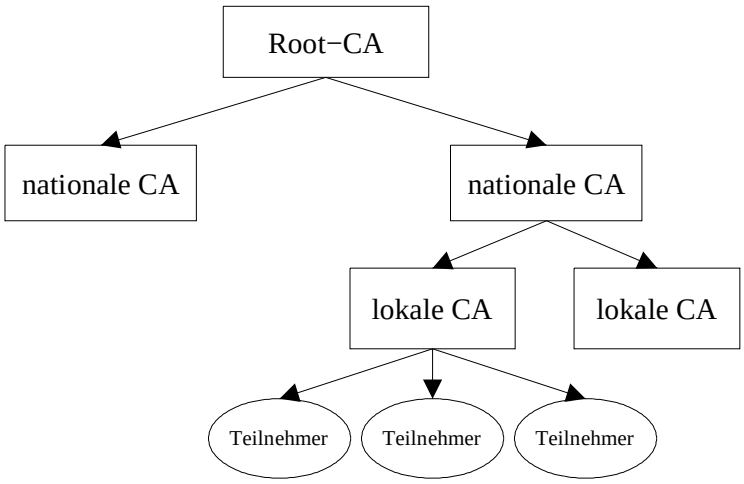


Abbildung 10: Beispiel einer X.509−Zertifizierungshierachie

Diese hierarchische Topologie wurde auch bereits im PKI−Vorschlag zu PEM festgeschrieben (siehe Abschnitt 3.3.8.1.3 und [RFC1422]). Ein X.509−Zertifikat besteht im wesentlichen aus folgenden Feldern (siehe auch [Glö96]):

- Versionsnummer (Version)
  X.509−Version, zu der das Zertifikat konform ist, derzeit also 1, 2 oder 3. Je nach Version sind bestimmte Zusatzfelder erlaubt bzw. erforderlich.
- Seriennummer (Serial Number)
  Eine bezogen auf die das Zertifikat herausgebende CA eindeutige Seriennummer (ganze Zahl). Nähere Bestimmungen zu der Seriennummer gibt es nicht.
- Algorithmen (Algorithm Identifier)
  Angaben zu den verwendeten kryptographischen Algorithmen.
- Herausgeber (Issuer Name)
  Der global eindeutige Name des Zertifikatherausgebers, also der DN der Zertifizierungsstelle.
- Gültigkeitsdauer (Period Of Validity)
  Anfang und Ende der Gültigkeitsdauer bezogen auf die Zeitzone GMT.
- Name des Objekts (Subject Name)
  Von der Zertifizierungsstelle dem zertifizierten Objekt zugewiesener, global eindeutiger Name.
- Öffentlicher Schlüssel (Subject Public Key Info)
  Der öffentliche Schlüssel des zertifizierten Objektes mit Angaben zu dem bei der Signatur verwendeten Kryptosystem.
- Digitale Signatur des Herausgebers (Signature) mit Angabe des verwendeten Algorithmus.

Zusätzlich können in X.509v2−Zertifikaten folgende Felder enthalten sein:

- Issuer Unique Identifier
  Bitstring, welcher den X.500−Namen der das Zertifikat herausgebenden CA eindeutig machen soll.

- Subject Unique Identifier
  Bitstring, welcher den X.500−Namen des zertifizierten Objektes eindeutig machen soll.

Diese Unique Identifier waren für den Fall vorgesehen, daß ein X.500−Name, der bereits einmal einem Objekt zugewiesen worden war, wieder verwendet werden soll. Z.B. könnte ein Mitarbeiter namens ’Peter Schmidt’ ein Zertifikat besitzen und die Firma verlassen. Später kommt ein neuer Mitarbeiter mit Namen ’Peter Schmidt’, welcher von dem ersten unterschieden werden können muß.

Ein selbst erzeugtes (fiktives) X.509−Zertifikat sieht beispielsweise in der üblichen Textdarstellung so aus (kryptographische Angaben sind gekürzt):

```
Certificate:
  Data:
    Version: 3 (0x2)
      Serial Number: 3 (0x3)
      Signature Algorithm: md5WithRSAEncryption
      Issuer: C=DE, L=CA Ort, O=Vertrau mir GmbH, OU=CA, CN=Premium CA
      Validity
          Not Before: Jul  7 14:42:16 1998 GMT
          Not After : Aug  4 14:42:16 1998 GMT
      Subject: C=DE, CN=Michael Stroeder/Email=nospam@nowhere.my
      Subject Public Key Info:
          Public Key Algorithm: rsaEncryption
          RSA Public Key: (1024 bit)
              Modulus (1024 bit):
                  00:cb:41:02:03:45:ad:d1:a2:84:f8:c5:dc:6c:0a:
                  [..]
                  5b:e3:37:fc:0c:16:4c:07:19
              Exponent: 65537 (0x10001)
  Signature Algorithm: md5WithRSAEncryption
      7c:5e:9b:e6:6d:52:c0:aa:b4:f9:3a:68:18:05:b8:84:f6:44:
      [..]
      28:3c:49:3d
```

Alle im Zertifikat gemachten Angaben müssen von jedem Benutzer verstanden werden, was eine eindeutige Definition aller Bezeichner und deren Bedeutung voraussetzt. Zugrunde liegt allen Bezeichnern eine hierarchisch vergebene, global eindeutige Objektidentifikationsnummer (Object ID, kurz OID), welche im Prinzip eine durch Punkte getrennte Aneinanderreihung von Ganzzahlen ist. Auch diese OID kommt aus der X.500−Definition, wo die Semantik eines Attributs eines Verzeichniseintrags durch zentrale Vergabe einer global eindeutigen OID festgeschrieben wird.

Diese streng hierarchisch strukturierte Vorgabe bei X.509 mit zwingend erforderlicher eindeutiger Namensgebung ist Vor− und Nachteil zugleich. Die Vorzüge der hierarchischen Aufteilung sind:

- Die Hierarchie entspricht meist ganz gut der Vorstellung einer hierarchischen Unterteilung von Unternehmen.
- Das Zertifizieren von Schlüsseln kann gut an Spezialisten delegiert werden. Auch dies entspricht der Vorstellung von einer spezialisierten CA−Abteilung innerhalb eines Unternehmens.
- Die eindeutige Namensgebung macht X.509−Zertifikate (theoretisch) global benutzbar.
- Zertifikate können transitiv überprüft werden, d.h. man überprüft ein Zertifikat, indem man das Zertifikat der ausgebenden CA und dann weiter die Zertifikate der Zertifizierungsstellen bis zur Wurzel−CA (oder der nächsten vertrauenswürdigen CA) überprüft.

Die Probleme bei der X.509−PKI sind:

- Eine eindeutige und sinnvolle CA−Hierarchie kann meist nicht gefunden werden. Beispielsweise könnte man als oberste Hierarchieebene Staaten nehmen, was aber schon bei einem multinationalen Unternehmen erste Probleme aufwirft.

- Eine hierarchische, global eindeutige Namensgebung ist nicht möglich, aber für eine PKI gemäß X.509 von essentieller Bedeutung.
- Die transitive Überprüfung von Zertifikaten setzt voraus, daß alle Zertifizierungsstellen in der CA−Kette beim Benutzer die gleiche Vertrauenswürdigkeit genießen. Vorgaben für Zertifizierungsverfahren, also die Voraussetzung zu einer standardisierten Vertrauenswürdigkeit, sind aber nicht Bestandteil der X.509−Spezifikation, sondern obliegen der Zertifizierungsstelle selbst. Die Vertrauensrelation entlang einer Zertifizierungskette ist also im allgemeinen nicht transitiv.
- Wird der private Schlüssel der Root−CA kompromittiert, so sind alle untergeordneten Zertifizierungsstellen kompromittiert.
- Es gibt derzeit keine klaren Vorstellungen darüber, wer die Root−CA betreiben soll und welche konkrete Verantwortung bzw. Verbindlichkeit sich daraus ergibt.

Tatsächlich entstehen derzeit kleinere, unabhängige Zertifizierungshierarchien, welche lokal z.B. auf eine Organisation beschränkt durchaus ihren Zweck erfüllen können.

3.3.8.1.1 X.509v3

Die derzeit neueste Version X.509v3 des X.509−Standards versucht einige der o.g. Probleme zu vermeiden bzw. abzumildern, indem das Format der X.509v3−Zertifikate und −Zertifikatwiderruflisten (CRL) individuell erweiterbar gemacht wurde. Als Basisattribute sind weiterhin die in Abschnitt 3.3.8.1 beschriebenen Felder enthalten. Die Erweiterungen sollen dazu dienen, zusätzliche Informationen im Zertifikat selbst unterzubringen:

- Zertifikatrichtlinien (Certificate Policies and Policy Mapping)
  Eine CA kann die Richtlinien, die bei der Erstellung des Zertifikats befolgt wurden, im Zertifikat unterbringen. Ein Zertifikatbenutzer soll anhand dieser Informationen leichter entscheiden können, ob das Zertifikat seinem Verwendungszweck entspricht. Beispielsweise könnte in dem Zertifikat enthalten sein, daß es nur zum Signieren von E−Mail gedacht und geeignet ist.
- Alternative Namen (Alternative Names)
  Weitere Namen sollen helfen, ein Zertifikat ohne Zugriff auf ein korrespondierendes X.500−Verzeichnis benutzen zu können. Z.B. können als weitere Namen Mail−Adressen oder eine URL angegeben sein.
- Weitere Verzeichnisattribute des zertifizierten Objektes (Subject Directory Attributes)
  Diese Erweiterung soll es erlauben, beliebige Attribute für spezielle Anwendungen mit in das Zertifikat aufzunehmen.
- Vorgaben für den Zertifizierungspfad (Certification Path Constraints)
  Mit dieser Erweiterung soll eine CA dem Zertifikatbenutzer Vorgaben machen können, welcher Zertifizierungspfad bei der Überprüfung des Zertifikates verfolgt werden muß. Insbesondere ermöglicht dies auch nicht hierarchische CA−Topologien. Vor allem kann dabei auch die Rolle des zertifizierten Objekts (z.B. CA oder Benutzer) explizit ausgewiesen werden.

Die Erweiterungen in X.509v3−CRLs sind:

- Nummer der Widerrufsliste und Widerrufsgrund (CRL Number and Reason Codes)
  Jede CRL für eine bestimmte CA und einen bestimmten Zertifikattyp bekommt eine streng monoton steigende Seriennummer, um es dem Benutzer zu ermöglichen festzustellen, ob er alle CRLs erhalten hat. Außerdem kann der Widerrufsgrund mit in der CRL untergebracht werden.

- Dienstzugangspunkt für die Bereitstellung der CRLs (CRL Distribution Points)
- Delta−CRL
  Um die Bereitstellung von Delta−CRLs (Abschnitt 3.3.6.2.2) zu ermöglichen, sind Erweiterungen für die dazu benötigten Informationen vorgesehen.
- Indirekte CRL

Zusätzlich gibt es erweiterte Attribute, welche auch selbst definiert werden können. Eine Organisation kann eine eigene OID beantragen und erhält im Prinzip einen ganzen OID−Raum als Zweig des global eindeutigen OID−Baums. Z.B. könnte die Organisation ’Vertrau mir GmbH’ eine CA betreiben und für sich die OID 34.133.545.223.67.32 (Beispiel völlig willkürlich gewählt) zugeteilt bekommen. Die ’Vertrau mir GmbH’ könnte nun beliebige Attribute mit einer speziellen Semantik definieren, indem sie weitere OIDs mit der eigenen OID als Prefix vergibt. Z.B. könnte das Feld mit Namen ’firmenwagen’ als Attribut 34.133.545.223.67.32.1.23 mit entsprechender Bedeutung vergeben werden. Dabei können weiter beliebige OID−Zweige mit dem registrierten OID−Prefix gebildet werden.
Das Problem hierbei ist die Bekanntmachung selbst vergebener OIDs, falls dies erwünscht ist. Ein Zertifikatbenutzer muß vor der Benutzung des Zertifikates die Bedeutung eines Attributes kennen. Außerdem ist die numerische Repräsentation von OIDs alles andere als benutzerfreundlich, eine textuelle Repräsentation wird aber nicht im Standard festgeschrieben und ist damit nicht mehr eindeutig. Auch könnte ein und das selbe Objekt mehrere OIDs haben. Ein Benutzer sieht dann ein ihm nur unter einer anderen OID bekanntes Objekt und meint, es nicht korrekt behandeln zu können, was z.B. bei der OID−Zuweisung zu kryptographischen Algorithmen sehr hinderlich sein kann.
Derzeit machen beispielsweise die beiden Software−Hersteller Netscape und Microsoft intensiven Gebrauch von der Vergabe eigener Objekt−IDs in ihren WWW− und LDAP−Produkten, was aber oft genug zu Mißverständnissen führt.

Der X.509v3−Standard selbst definiert bereits mehrere Attribute [X509]. An dieser Stelle sollen kurz mehrere dieser Attribute beschrieben werden, die bereits im Netscape Communicator Verwendung finden. Anwendungsspezifische Netscape−Attribute werden in Abschnitt 3.3.8.1.2 vorgestellt.

- *keyUsage*
  Dieses Attribut bestimmt den Verwendungszweck des Zertifikats (Tabelle 1).
- *basicConstraints*
  Dieses Attribut wird dazu verwendet, um CA−Zertifikate zu identifizieren und die Länge des Zertifikatpfades zu begrenzen. Das Attribut besteht wiederum aus mehreren Komponenten: Das Flag *cA* zeigt an, daß dieses Attribut ein CA−Zertifikat ist. Die Komponente *pathLenConstraint* legt die maximale Länge des Zertifikatpfades fest.
- *authorityKeyIdentifier*
  Existieren mehrere Zertifizierungsstellen mit dem gleichen Namen (*subject name*), so bestimmt dieses Attribut, welche CA das Zertifikat herausgegeben hat. Dieses Attribut besteht ebenfalls aus mehreren Komponenten: *keyIdentifier*, *authorityCertIssuer*, *authorityCertSerialNumber* und *authorityCertIssuer* machen weitere Angaben zur das Zertifikat herausgebenden CA.
- *subjectKeyIdentifier*
  Diese Erweiterung wird benutzt, um verschiedene Zertifikate gleichen Namens zu unterscheiden. Ist das Feld nicht vorhanden, so wird sein Wert mittels SHA−1−Hash aus dem öffentlichen Schlüssels des zertifizierten Objekts (Attribut *subjectPublicKeyInfo*) berechnet.

Der Verwendungszweck kann mithilfe des keyUsage−Attributes folgendermaßen festgelegt werden:

Verwendungszweck	*Inhalt des keyUsage−Attributs*
SSL Client	digitalSignature
SSL Server	keyEncipherment
S/MIME Signing	digitalSignature
S/MIME Encryption	keyEncipherment
Certificate Signing	keyCertSign
Object Signing	digitalSignature

Tabelle 1Definition des Verwendungszwecks im keyUsage−Attribut

3.3.8.1.2 Netscape−Attribute für X.509v3−Zertifikate

Von Netscape wurden anwendungsspezifische Attribute für X.509v3−Zertifikate definiert, welche speziell für den Einsatz der Zertifikate mit dem Netscape Communicator gedacht sind und alle mit dem OID 2.16.840.1.113730 beginnen [Nets97c][Nets97d]:

- *nsComment*
  enthält einen beliebigen textuellen Kommentar, welcher den Verwendungszweck des Zertifikats beschreibt.
- *nsBaseUrl*
  enthält eine Basis−URL−Adresse für WWW−Zugriffe auf CA−Daten
- *nsCaRevocationUrl*
  enthält eine relative URL−Adresse zum Abruf der aktuellsten CRL
- *nsRevocationUrl*
  enthält eine relative URL−Adresse zur Online−Prüfung von Zertifikaten (siehe auch Abschnitt 6.7.3.5)
- *nsRenewalUrl*
  enthält eine relative URL−Adresse für die Zertifikatverlängerung
- *nsCaPolicyUrl*
  enthält eine relative URL−Adresse für den HTML−Text der Zertifizierungsrichtlinien
- *nsCertType*
  ist eine Ganzzahl (ein Byte) und schränkt den Verwendungszweck des Zertifikats ein (siehe Tabelle 2)
- *nsSslServerName*
  Regulärer Ausdruck für den Host−Namen des Servers, welcher das Zertifikat benutzt.

Der Verwendungszweck eines Zertifikats wird durch folgende bitweise Zusammensetzung der in *nsCertType* angegebenen Ganzzahl eingeschränkt:

Bit	*Verwendungszweck*	*DER−kodiert in ssleay.cnf*
0	SSL Client	0x80
1	SSL Server	0x40
2	S/MIME for client use	0x20
3	Object Signing (e.g. Java applets)	0x10
4	reserviert für zukünftige Erweiterungen	0x08
5	SSL CA	0x04
6	S/MIME CA	0x02
7	Object Signing CA	0x01

Tabelle 2Definition des Verwendungszwecks im nsCertType−Attribut

3.3.8.1.3 PKI gemäß Privacy Enhanced Mail (PEM)

Die ersten Bemühungen, einen auf X.509−Zertifikaten basierten, gesicherten Nachrichtenversand zu definieren, stellen die Entwürfe für *PEM* (Privacy Enhanced Mail) dar [RFC1421][RFC1422][RFC1423][RFC1424]. Eine grundlegende Beschreibung der erforderlichen PKI enthält [RFC1422]. In diesem Entwurf wird eine streng hierarchische CA−Topologie vorgeschlagen. Zu diesem Zweck sieht der Entwurf die Einrichtung einer einzelnen, globalen Root−CA vor, welche ihrerseits nationale oder organisationsweite Zertifizierungsstellen zertifiziert. Es wird nicht weiter benannt, wer diese zentrale Root−CA betreiben soll. Es wird auch ein Verfahren vorgestellt, welches die Eindeutigkeit der Namensgebung in Zertifikaten sichern soll. Die Probleme mit einer streng hierarchischen X.509−PKI wurden schon in Abschnitt 3.3.8.1 dargestellt.
Der PEM−Standard ist inzwischen überholt und hat auch nie eine weitere Verbreitung gefunden, die gemachten Vorschläge haben aber teilweise als Vorbild für spätere Standards wie PKCS (Abschnitt 3.5.1) und S/MIME (Abschnitt 3.5.4) gedient.

3.3.8.1.4 PKIX

PKIX (Public−Key Infrastructure X.509) [PKIX] definiert Protokolle für alle Aspekte des Betriebs und der Verwaltung einer PKI, ohne dabei Vorgaben bzw. Einschränkungen bezüglich der Zertifizierungsrichtlinien (z.B. Ort der Schlüsselerzeugung, CA−Topologie etc.) zu machen. Abbildung 11 gibt eine schematische Übersicht über die in PKIX definierten Entitäten [Ada98]. Auch PKIX basiert auf X.509v3 (Abschnitt 3.3.8.1), ohne jedoch die streng hierarchische CA−Struktur von X.509 zu übernehmen. Einige Formatvorgaben basieren auf den PKCS−Standards (Abschnitt 3.5.1).

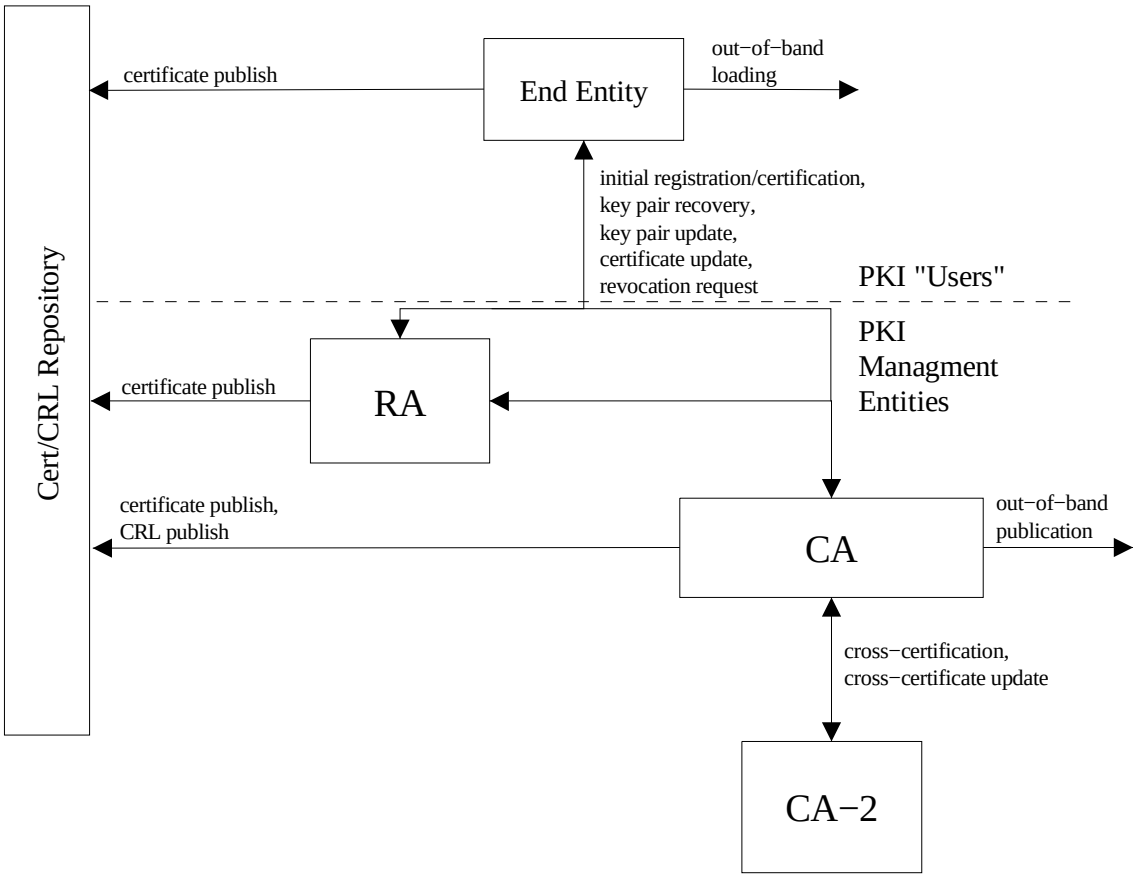


Abbildung 11: Übersicht der in PKIX definierten Entitäten

Folgende Aspekte sind im PKIX−Entwurf enthalten:

- Formate von X.509v3−Zertifikaten und X.509v2−Zertifikatwiderruflisten mit speziellen Zertifikat−Attributen für den Aufbau einer PKI
- Format von Zertifikatanforderungen (Off−Line, über HTTP/XML etc.)
- Verfahrensweisen für sämtliche Vorgänge beim Aufsetzen, der Verwaltung und Benutzung einer PKI
- Nachrichtenformate zwischen Entitäten
- Empfehlungen für das Erstellen von Zertifizierungsrichtlinien und den Zertifizierungsablauf
- Entwürfe für die Online−Überprüfung von Zertifikaten
- Bereitstellung von Zertifikaten und Zertifikatwiderruflisten über Mail, HTTP, FTP und LDAP
- Formate von Zeitstempel−Zertifikaten
- Empfehlungen für den Betrieb einer Schlüsselvergabestelle (vergleiche Abschnitt 3.3.8.4)

Reale Implementierungen der PKIX−Standards bzw. deren Integration in schon bestehende Produkte sind wohl erst im Laufe des Jahres 1999 zu erwarten, aber die umfassende und offene Normung aller Aspekte rund um den Aufbau einer PKI, lassen PKIX als den zukunftsweisenden, kommenden Standard erscheinen.

3.3.8.2 Web−Of−Trust (PGP)

Eines der ersten weitverbreiteten Verschlüsselungsprogramme mit asymmetrischen Algorithmen war PGP von Phil Zimmermann, erstmals auf dem Internet veröffentlicht im Juni 1991. PGP ist ein komplettes System zur Verschlüsselung von E−Mail und Dateien und enthält demnach auch Definitionen zur Standardisierung von Formaten verschlüsselter Nachrichten, Schlüsseln und digitalen Signaturen. PGP ist ein Hybridsystem, welches in der ursprünglichen Form als asymmetrischen Algorithmus RSA benutzte, um die Sitzungsschlüssel für den symmetrischen IDEA−Algorithmus zu schützen. Inzwischen sind weitere asymmetrische und symmetrische Algorithmen hinzugekommen.
Insbesondere legt PGP auch die Verfahrensweise für die Verbreitung von öffentlichen Schlüsseln fest, definiert also eine PKI. Im folgenden werden die PKI−Charakteristika von PGP kurz beschrieben, eine tiefergehende Einführung in PGP findet sich in [Gar95].

3.3.8.2.1 Zertifizierung von Schlüsseln

In PGP verwaltet jeder Benutzer je einen öffentlichen und einen privaten *Schlüsselbund* (public/private key ring). Im öffentlichen Schlüsselbund werden die öffentlichen Schlüssel anderer Teilnehmer und deren RFC822−konforme[7] Mail−Adressen gespeichert. Um eine Verfälschung der Schlüsselbunde zu vermeiden, sind diese mit dem privaten Schlüssel des Besitzers signiert. Wird ein öffentlicher Schlüssel oder ein ganzer öffentlicher Schlüsselbund eines anderen Teilnehmers zum Schlüsselbund hinzugefügt, so wird jeder neue öffentliche Schlüssel quasi durch den Schlüsselbundinhaber signiert. Der Benutzer ist also in der PKI Zertifizierungsstelle (CA), zertifiziertes Objekt (subject) und PKI−Benutzer (user) in einer Person. Der Grundgedanke bei PGP ist, daß sich so durch die beliebige Weitergabe öffentlicher Schlüssel ein unstrukturiertes Netz von Vertrauensrelationen (web−of−trust) ausbildet.

Beim Signieren öffentlicher Schlüssel muß der Schlüsselbundinhaber jedem öffentlichen Schlüssel eine von vier möglichen Vertrauensstufen zuweisen, welche ebenfalls im Schlüsselbund mit abgelegt werden[8]:

- *Completeley trusted*
  Jedem weiteren Schlüssel, welcher von einem als ’Completeley trusted’ markierten Schlüssel signiert ist, wird blind vertraut.
- *Marginally trusted*
  Ein Schlüssel, welcher von einem als ’Marginally trusted’ markierten Schlüssel signiert ist, muß noch von mindestens einem weiteren Schlüssel signiert sein.
- *Untrusted*
  Ein solcher Schlüssel wird überhaupt nicht verwendet, um die Vertrauenswürdigkeit anderer Schlüssel zu überprüfen.
- *Unknown*
  Eine Vertrauensstufe kann nicht ermittelt werden, in der Praxis gleich der Markierung als *Untrusted*.

Zu einer wirklich sicheren Nutzung muß bei PGP bei jeder Kommunikationsbeziehung ein Schlüsselaustausch über einen sicheren Kanal erfolgen. In diesem Fall erfüllt PGP den Zweck der sicheren Punkt−zu−Punkt−Verbindung voll und ganz. Man kann zwar Zertifikatketten bilden, deren Wert aber noch zweifelhafter ist als bei anderen PKI, da mehr oder minder sorgfältig vorgehende Benutzer die

[7] Beispiele RFC822−konformer E−Mail−Adressen, ebenfalls RFC822−konform durch Kommata getrennt: michael@nowhere.de, Michael Stroeder <michael@nowhere.de>, (michael@nowhere.de) Michael Stroeder

[8] Da die englische PGP−Version wahrscheinlich am weitesten verbreitet ist, werden hier gleich die englischen Bezeichnungen gewählt.

öffentlichen Schlüssel zertifizieren. Desweiteren gibt es keine PKI−Topologie bzw. Vorgaben für Zertifizierungspfade, was das Auffinden und die Überprüfung unbekannter Zertifikate erschwert.
Sogenannte *PGP−Schlüssel−Server* (PGP key server), auf denen PGP−Benutzer ihre öffentlichen Schlüssel ablegen können, mildern diese Umstände etwas ab und erleichtern zumindest das Auffinden öffentlicher Schlüssel zu einem bestimmten Namen. Die Vertrauenswürdigkeit dieses Servers ist aber auch zweifelhaft.

3.3.8.2.2 Widerruf

PGP−Schlüssel haben uneingeschränkte Gültigkeitsdauer. Dies ist im Hinblick auf evtl. kompromittierte Schlüssel und Fortschritte bei der für Angriffe verfügbaren Rechenleistung ein schweres Problem. Um einen öffentlichen Schlüssel zu widerrufen, muß der Schlüsselinhaber ein spezielles *Widerrufszertifikat* (key revocation certificate) ausstellen, welches er an alle seine Kommunikationspartner verschickt. Wird jemand dabei vergessen bzw. hat sich jemand den Schlüssel über Dritte beschafft, so ist die weitere Benutzung des kompromittierten Schlüssels nicht auszuschließen, und das auf immer und ewig. PGP bürdet also alle Verantwortung den Teilnehmern auf.

3.3.8.2.3 PGP 6

Die neueste Version PGP 6 erweitert PGP um einige PKI−Charakteristika, welche denen von X.509 ähneln und die o.g. Einschränkungen von PGP aufheben sollen [Luck98]:

- Ein Schlüssel kann als *Trusted Introducer* signiert werden, welchem dann wie einer Zertifizierungsstelle vertraut wird. Diese Einschätzung kann an Dritte weitergegeben werden (Zertifizierungspfad).
- Das einem Schlüssel entgegengebrachte Vertrauen kann auf bestimmte Internet−Domänen−Namen eingeschränkt werden.
- Zertifikaten kann ein Gültigkeitszeitraum zugewiesen werden.
- Ein Schlüsselinhaber kann im Voraus andere PGP−Teilnehmer dazu bevollmächtigen, ein Widerrufszertifikat auszustellen.
- Die Verwendung von Schlüssel−Servern (key server) wird standardmäßig unterstützt.
- Eine Organisation kann Richtlinien für die interne Verwendung von PGP festlegen, wie z.B. die erzwungene Verschlüsselung von Nachrichten mit einem zusätzlichen Wiederherstellungsschlüssel (ADK, Additional Decryption Key).

Damit gleichen sich X.509v3/PKIX und PGP bezüglich ihrer PKI−Charakteristika immer mehr an. PGP 6 verliert dabei aber seine Kompabilität zu PGP 2. Zu beachten ist auch, daß PGP 6 ein proprietärer, von nur einem Hersteller abhängiger Standard ist.

3.3.8.3 Secure DNS

Der Domänen−Namens−Dienst im Internet (Domain Name System, kurz: DNS) ist ein grundlegender Dienst, der von allen anderen Internet−Diensten genutzt wird. Das DNS stellt eine Architektur und Verteilungsmechanismen für die Zuordnung von symbolischen Namen zu IP−Adressen bereit. DNS selbst ist ein völlig ungesicherter Dienst, Angriffe auf die Integrität der Namens−Adreß−Zuordnung sind vergleichsweise einfach möglich [Mraz97].
Um den Sicherheitsproblemen von DNS zu begegnen, wurden inzwischen Erweiterungen vorgeschlagen, das System mit digitalen Signaturen für DNS−Einträge zu sichern [RFC2065]. Dabei ist auch eine

Verteilung öffentlicher Schlüssel für die verwendeten asymmetrischen kryptographischen Algorithmen, also im Prinzip eine PKI, vonnöten.

3.3.8.3.1 Grundbegriffe des DNS

Das DNS spannt einen hierarchischen Namensraum auf, wobei eine *Domäne* (domain) eine Ansammlung von Rechnern (hosts) ist, welche in irgendeiner Art und Weise zusammengehören. Domänen können wiederum in *Unterdomänen* (subdomains) aufgeteilt werden. Die Wurzel des Namensbaumes wird Wurzeldomäne (root domain) genannt und durch einen einzigen Punkt gekennzeichnet. Dieser Punkt wird angehängt, um zu zeigen, daß eine Namensangabe ein vollständig angegebener Domänenname (fully qualified domain name, kurz FQDN) ist. Ansonsten können Namen auch relativ zu einer Domäne angegeben werden.

Jede Domäne hat einen oder mehrere *Namens−Server* (name server), welche autorisiert sind, die Auskünfte über eine Domäne zu geben, welche wiederum auch die Adressen der autorisierten Namens−Server von Unterdomänen enthalten. Um zwischen einer Domäne und einem Bereich von Rechnern zu unterscheiden, für den ein Namens−Server voll autorisiert ist, wird der Begriff *Zone* verwendet. Neue DNS−Einträge für eine Zone werden auf den dafür voll autorisierten Namens−Servern erzeugt, andere Server verteilen diese DNS−Einträge über rekursive Abfragemechanismen weiter.
Wenn eine Anwendung die IP−Adresse zu einem Namen, z.B. 'www.uni−karlsruhe.de.', finden möchte, so fragt sie erst ihren lokalen Namens−Server. Hat dieser den Namen nicht in seiner lokalen Datenbank, so fragt er einen Root−DNS−Server an der Wurzel des DNS−Baumes, welcher die Antwort auch nicht weiß, aber einen Verweis auf den für die Domäne 'de.' zuständigen DNS−Server gibt, welcher wiederrum einen Verweis auf den für die Subdomäne 'uni−karlsruhe.de.' autorisierten DNS−Server gibt, welcher schließlich und endlich die IP−Adresse für 'www.uni−karlsruhe.de.' zurückgibt. Das Ganze kann noch komplizierter verschachtelt werden, je nachdem wieviel Hierarchieebenen durchlaufen werden müssen und wie die Konfiguration lokaler DNS−Server aussieht. Um die DNS−Abfragen effizienter zu gestalten, halten DNS−Server die Ergebnisse früherer Anfragen in einem temporären Zwischenspeicher (Cache) vor, um spätere Anfragen nach dem selben Hostnamen schneller beantworten zu können.
Neben den eigentlichen Adreß−Einträgen (A−records) kann im DNS auch andere Information vorgehalten werden. Im allgemeinen werden DNS−Einträge in sogenannte resource records (RR) geschrieben. Ein RR besteht aus dem DNS−Namen (owner name), der Klasse (class), dem Typ (type) und einem Datum (data). Ein DNS−Name ist immer eine durch Punkte getrennte Liste von Strings und kann eine Zone, einen Rechnernamen, einen Benutzer oder andere Informationen enthalten. Als Klasse gibt es derzeit nur die Klasse IN der DNS−Einträge für das Internet. Der Typ kann z.B. eine Adresse (Typ A), einen Mail−Server (Typ MX), einen weiteren Namens−Server (Typ NS) angeben. Z.B. ist der für die Domäne 'uni−karlsruhe.de' zuständige Mail−Server (oder auch mehrere) in den speziellen DNS−Einträgen (MX−records) festgeschrieben, welche z.B. von einem MTA[9] beim Versenden von Mail via SMTP[10] abgefragt werden:

```
uni−karlsruhe.de IN MX smtp.rz.uni−karlsruhe.de
```

3.3.8.3.2 Kryptographische DNS−Erweiterungen

Für die kryptographischen DNS−Erweiterungen im Entwurf zu SecureDNS wurden die RR−Typen KEY und SIG vorgeschlagen, um eine Verteilung öffentlicher Schlüssel und die Authentifizierung von DNS−

[9] MTA: Message Transfer Agent. Der Prozeß, welcher für die Weiterleitung von E−Mail zuständig ist. Als Beispiel sei hier *sendmail* als prominentester Vertreter genannt.
[10] SMTP: Simple Mail Transfer Protocol

Einträgen zu ermöglichen [RFC2065]. Ein KEY−Eintrag ermöglicht es, einen DNS−Namen mit einem öffentlichen Schlüssel zu verknüpfen, und kann Markierungen enthalten, welche den Verwendungszweck des Schlüssels beschreiben bzw. einschränken.
Ein SIG−Eintrag ist dafür vorgesehen, eine digitale Signatur, versehen mit einem Gültigkeitszeitraum, für eine Menge anderer RR zu enthalten. Die signierten RR und der SIG−Eintrag stellen zusammen also eine Art DNS−Zertifikat dar. Der Entwurf sieht vor, daß in den meisten Fällen für jede Zone ein privater Schlüssel zum Generieren der SIG−Einträge verwendet wird. Die Zonenverwaltung agiert also als eine Zertifizierungsstelle für die eigenen Zoneneinträge.
Das Datenfeld eines SIG−RR enthält:

- Digitale Signatur
- Zertifizierter RR−Typ
- Zeitpunkt der Signatur
- Verfallszeitpunkt (expire)
- Name des Signierenden (normalerweise die Zone, welche den RR enthält)

In jeder Zone wird auch ein Zertifikat für die Subdomänen und die darüber liegende Zone ausgestellt, deren öffentliche Schlüssel über sichere Kanäle beschafft werden müssen. Damit ist dann eine Benutzung der ganzen SecureDNS−PKI möglich. Die Überprüfung von Zertifikaten erfolgt online. Hat ein SecureDNS−Server den öffentlichen Schlüssel seiner Super−Zone, so kann er von dieser das Zertifikat für die Super−Super−Zone erhalten und sich so hochhangeln bis zum Zertifikat der root−Domäne, von der er wieder die Zertifikate gewünschter Sub−Domänen erhält, also analog zur in Abschnitt 3.3.8.3.1 beschriebenen Abfrage der IP−Adressen. Auch Zertifikate zwischen beliebigen Zonen untereinander sind möglich, um die Effizienz zu verbessern. Zertifikate werden widerrufen oder geändert, indem die Zonenverwaltung die betreffenden RRs und den SIG−RR löscht bzw. ändert.

Das Bestechende des Ansatzes von SecureDNS als PKI ist, daß ein schon bestehender und sowieso im Internet zwingend benötigter Dienst zu einer PKI erweitert wird.
Die Probleme dabei sind:

- Die Namensgebung im DNS ist starken Restriktionen unterworfen, und die begehrten 2nd Level−Domänen unterhalb der Top−Level−Domänen sind bereits fast ausgeschöpft.
- Es wird sich auch bei SecureDNS auf eine Wurzel−Zertifizierungsstelle verlassen. Wird dieses lohnende Angriffsziel kompromittiert, so ist die ganze PKI kompromittiert.
- Die Spezifikation erlaubt derzeit nur 254 verschiedene Verwendungszwecke eines Zertifikates.
- Der CA einer Zone muß voll vertraut werden, nur sie weist Schlüsseln einen eingeschränkten Verwendungszweck zu. Es gibt keine Möglichkeiten für eine Zone, den Zertifikaten einer anderen Zone nur eingeschränkt zu vertrauen.
- Es ist oft üblich, daß der Internet−Provider die DNS−Zonen seiner Kunden pflegt. D.h. die Internet−Provider werden automatisch zu einer Zonen−Zertifizierungsstelle.

Derzeit hat SecureDNS noch keine weitere Verbreitung gefunden, schon gar nicht in den heute bereits üblichen kryptographischen Kommunikationsprotokollen, so daß man SecureDNS noch nicht mit PGP's Web−Of−Trust oder X.509 vergleichen könnte.

3.3.8.4 Royal−Holloway (Trusted Third Parties)

Bisher wurden Zertifizierungsinfrastrukturen beschrieben, welche alle so konzipiert sind, daß niemand anderes im Besitz des privaten Schlüssels sein sollte als das zertifizierte Objekt selbst. Das birgt aber ein Problem bezüglich der Datensicherheit: Speichert ein Anwender Daten ausschließlich in verschlüsselter Form und verliert den privaten Schlüssel, so sind diese Daten nicht wieder herzustellen. Zu diesem Zweck sehen manche PKI−Entwürfe die Hinterlegung privater Schlüssel bei einem Trust Center (Abschnitt 3.3.3) vor. Ein solches PKI−Konzept stellt das *Royal−Holloway−Verfahren* dar [Mit96].
Das Royal−Holloway−Verfahren basiert auf vertrauenswürdigen Instanzen (Trusted Third Parties), welche Schlüssel erzeugen, ausgeben und zertifizieren. Jeder Teilnehmer wird dabei genau einer TTP−Stelle zugeordnet, von der er seine Schlüsseldaten bezieht (Abbildung 12).

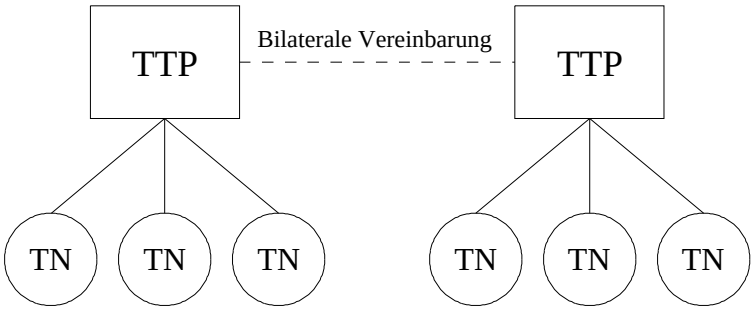


Abbildung 12: TTP−Infrastruktur für Verschlüsselungsschlüssel

Die TTP erzeugt für jeden der ihr zugeordneten Teilnehmer TN jeweils ein Schlüsselpaar zum Senden (KS_{pub},KS_{priv}) und ein Schlüsselpaar zum Empfangen (KE_{pub},KE_{priv}) von Nachrichten, welches jeweils aus einem privaten und dem dazugehörigen öffentlichen Schlüssel des Teilnehmers besteht. Innerhalb eines TTP−Bereiches errechnen die Teilnehmer damit einen festen symmetrischen Kommunikationsschlüssel: Der Sender berechnet seinen Kommunikationsschlüssel KK_{send} aus seinem privaten Sendeschlüssel KS_{priv} und dem öffentlichen Empfangsschlüssel des Empfängers KE_{pub}. Entsprechend verwendet der Empfänger zur Berechnung seines Kommunikationsschlüssels KK_{empf} den öffentlichen Sendeschlüssel KS_{priv} des Absenders und seinen privaten Empfangsschlüssel KE_{priv}.
Die so errechneten Kommunikationsschlüssel KK_{send} und KK_{empf} werden zur Verschlüsselung eines zufällig gewählten Sitzungsschlüssels benutzt, mit dem dann die eigentliche symmetrische Verschlüsselung des Dokuments durchgeführt wird (sog. Master−/Session−Key−Verfahren).
Möchten Teilnehmer unterschiedlicher TTPs miteinander kommunizieren, so müssen die beiden TTPs vorher einen gemeinsamen symmetrischen Schlüssel vereinbaren, welcher geheimzuhalten ist und nur für diese spezielle bilaterale TTP−Vereinbarung gilt. Der Empfangsschlüssel eines Teilnehmers einer fremden TTP wird mit Hilfe dieses bilateralen TTP−Schlüssels aus teilnehmerspezifischen offenen Daten berechnet. Anschließend ist es den zugehörigen Teilnehmern auch TTP−übergreifend möglich, verschlüsselt zu kommunizieren.
Beim Royal−Holloway−Verfahren kann jede TTP jeden Kommunikationsschlüssel der ihr direkt zugeordneten Teilnehmer berechnen, da sie vorher die Schlüsselpaare an die Teilnehmer ausgegeben und gespeichert hat. Kommunikationsschlüssel zwischen Teilnehmern anderer TTPs sind für die TTP nicht zugänglich. Dies ermöglicht bei einem Schlüsselverlust die Wiederherstellung von Daten, welche ansonsten verloren wären.

Das Royal−Holloway−Verfahren birgt mehrere echte Gefahren: Eine ernstzunehmende Geheimhaltung von bilateralen Kommunikationsschlüsseln zwischen TTPs ist kritisch bzw. als unrealistisch einzustufen. TTPs stellen einen guten zentralen Angriffspunkt auf private Schlüsseldaten dar. Zudem sollte man nicht die Bedeutung mangelnder Anwenderakzeptanz unterschätzen. U.U. benutzen die Anwender nur sehr ungerne Verschlüsselung, von der sie wissen, daß Dritte die Kommunikationsbeziehung mitlesen können. Eine weitergehende Diskussion über die Probleme dieses Verfahrens findet man in [Laur96].

Obwohl der Vorteil der Wiederherstellung von Dateien nicht zu unterschätzen ist, ist der Einsatz von TTPs eher nicht zu empfehlen. Benutzer können auch auf andere Weise Dateien fahrlässig löschen, d.h. eine Organisation sollte eher geeignete Richtlinien zur Speicherung von wichtigen Daten und entsprechende Datensicherungsmaßnahmen einführen, um möglichen Datenverlusten durch Schlüsselverluste vorzubeugen, denn eine Schlüsselvergabestelle.

3.4 Zertifizierungsstellen−Charakteristika

In folgenden soll kurz beschrieben werden, welche Voraussetzungen für den Aufbau einer Zertifizierungsstelle erfüllt werden müssen. Dieser Abschnitt ist sehr allgemein gehalten, und es soll von der eigentlichen Zertifizierungsinfrastruktur abstrahiert werden. Die konkrete Realisierung einer Zertifizierungsstelle im Testbetrieb für die Fa. Propack Data GmbH wird in Kapitel 6 dargestellt.

3.4.1 Wer zertifiziert wen?

Es gibt viele verschiedene Arten von Beziehungen zwischen einer Zertifizierungsstelle und den zertifizierten Objekten. Exemplarisch sollen hier ein paar Möglichkeiten genannt und die Vor− und Nachteile beleuchtet werden.

3.4.1.1 Interne CA

Eine Organisation kann eine eigene Zertifizierungsstelle betreiben, um z.B. speziell Angehörige der eigenen Organisation als solche zu bestätigen. Die Vorteile sind:

- Die Beteiligten (CA−Personal und Benutzer) sind sich oft wenigstens teilweise persönlich bekannt.
- Sichere Out−Of−Band−Authentifizierung ist sehr leicht zu bewerkstelligen, z.B. erkennt man die Stimme bekannter Personen am Telefon meist sehr gut.
- Das Vertrauensverhältnis von Mitarbeitern zur CA ist groß.
- Das CA−Personal kann die Benutzer speziell schulen, speziell im Hinblick auf die Eigenverantwortung der Benutzer bei der Handhabung privater Schlüssel und die detaillierte Bedeutung der Zertifikate.
- Strukturelle Änderungen im Rahmen der PKI, z.B. neue Zertifikattypen für einen neuen Anwendungsfall, sind leicht durchzuführen.
- Die Organisation selbst hat die volle Kontrolle über Bezeichnungen für Identitäten und Attribute.
- In einer Firma mit festen Angestellten ergibt sich z.B. schon aus dem Arbeitsrecht auch ein definierter juristischer Rahmen für die Bedeutung der Zertifikate, d. h. in Form einer Arbeitsanweisung definierte Verfahrensweisen müssen von den Mitarbeitern eingehalten werden.

Als Nachteile können angesehen werden:

- Der zeitliche und finanzielle Aufwand ist sehr hoch.
- Geeignetes Personal für die CA, welches über genügend Sachverstand und Verantwortungsbewußtsein verfügt, ist nur schwer zu finden.
- Eine Einbeziehung externer Benutzer gestaltet sich u.U. schwierig, der Aufwand zur sicheren Verbreitung des eigenen CA−Zertifikates steigt, je loser die Beziehung zu den (externen) Benutzern ist.

Ob sich die Einrichtung einer eigenen CA lohnt, hängt von der Größe einer Organisation und der konkreten Anwendung der ausgestellten Zertifikate ab. Je größer die Organisation und je höher die Anforderungen an eine klar definierte Verbindlichkeit der Zertifikate sind, desto lohnender ist der Aufbau einer eigenen CA.

3.4.1.2 Externe, beauftragte CA

Eine Organisation kann auch eine andere Firma oder Organisation mit dem Ausstellen von Zertifikaten betrauen (Outsourcing).
Die Vorteile sind:

- Der Zertifizierungsbetrieb kann an (hoffentlich) sachkundige Experten delegiert werden, was in aller Regel kostengünstiger ist (z.B. Personal− und Hardwarekosten).
- Je nach Verbreitungsgrad der beauftragten CA können externe Benutzer besser in die PKI mit einbezogen werden.

Nachteile sind:

- Die konkrete Bedeutung eines Zertifikates kann meist nicht sehr fein granuliert definiert werden, auch kleine strukturelle Änderungen ziehen meist einen kostenpflichtigen Auftrag an die CA−Betreiber nach sich.
- Die sorgfältige Ausarbeitung eines Outsourcing−Vertrages ist sehr aufwendig.
- Outsourcing ist meist gar nicht oder nur mit sehr großen Anstrengungen rückgängig zu machen.
- Es muß teilweise eine den Betrieb und die Sicherheit einschränkende Gesetzgebung hingenommen werden, wenn sich die CA in Ländern mit gesetzlichen Restriktionen bezüglich der Anwendung von Kryptographie befindet.
- Eine echte juristische Verbindlichkeit wird es aller Wahrscheinlichkeit nicht geben, da die beauftragte CA versuchen wird, eine Haftung möglichst weitgehend auszuschließen. Das gelingt ihr auch gut, da sie sich selbst ihre eigenen Zertifizierungsrichtlinien vergibt. Abhilfe soll hier das Signaturgesetz schaffen, aber viele Details im deutschen Signaturgesetz sind unklar [Kop98] und auf EU−Ebene befindet sich die Gesetzgebung derzeit noch in der Entwurfsphase.

3.4.1.3 Staatliche CA

Denkbar ist auch, daß staatliche Stellen die Aufgabe übernehmen, für alle Bürger kryptographische Zertifikate auszustellen. Im Prinzip betreiben die meisten Staaten bereits solche Zertifizierungsstellen in Form von Einwohnermeldeämtern, Polizeibehörden etc., mit Reisepässen oder Personalausweisen (identity card) als nicht kryptographischen Personen−Zertifikaten − die Infrastruktur ist also bereits vorhanden. Dies hätte den Vorteil, daß die Überprüfung der Identität eines Objektes für ein kryptographisches Zertifikat genauso sicher durchgeführt werden könnte wie die Überprüfung für

beispielsweise einen Reisepaß. Auch die juristische Verbindlichkeit wäre besser gewährleistet, wenn die Gesetzgebung entsprechend angepaßt würde. Allerdings wird von staatlichen Stellen meist nur die Identität von Personen überprüft. Die Identität z.B. eines WWW−Servers kann eine Polizeibehörde nur schwerlich mit den herkömmlichen Mitteln (Geburtsurkunde etc.) prüfen. Auch ist das Ausstellen von Attributzertifikaten nur schwer möglich bzw. aus Gründen des Datenschutzes auch meist gar nicht erwünscht.

3.4.2 Zertifikattypen

Es ist sinnvoll und oft auch zwingend notwendig, verschiedene Zertifikattypen für bestimmte Anwendungen auszustellen. Die Zertifikate können dann auf den Anwendungszweck abgestimmte Attribute enthalten. U.U. wird von der Software schon die Einschränkung der Zertifikatverwendung unterstützt, oder bei bestimmten Attributen wird eine bestimmte Semantik vorausgesetzt. Dabei können die verschiedenen Zertifikate von einer gemeinsamen CA ausgestellt werden, oder für jeden Zertifikattyp wird eine eigene CA mit eigenem privaten Schlüssel eingerichtet. Ist eine mehrstufige CA−Hierarchie akzeptabel, so kann eine übergeordnete CA jeweils eine Sub−CA für jeden Zertifikattyp zertifizieren, um den Initialaufwand bei der sicheren Übermittlung der CA−Zertifikate an die Teilnehmer zu verringern.

3.4.3 Richtlinien (Policy)

Eine Zertifizierungsstelle muß ihre eigenen Daten und *Richtlinien* (Policy*)* veröffentlichen. In diesem Dokument beschreibt die Zertifizierungsstelle möglichst vollständig und detailliert die für den Zertifizierungsvorgang relevanten Details und selbst auferlegten Regelungen, in der Literatur oft auch *Certification Service Rules* (CSR) genannt. Die Veröffentlichung dieser Richtlinien soll den Benutzer in die Lage versetzen zu beurteilen, wie weit er den Zertifikaten einer bestimmten Zertifizierungsstelle für einen bestimmten Verwendungszweck vertrauen kann.
Dabei ist zu beachten, daß die Policy ein recht langlebiges Dokument ist, welches sich nach Erstellen gar nicht oder nur noch unwesentlich ändern sollte. Grundlegende Änderungen an der Policy kommen einer neuen, anderen Zertifizierungsstelle gleich. Eine Zertifizierungsstelle kann mehrere solcher Policy−Dokumente für verschiedene Sicherheitsstufen anbieten. Eine solche Selbstdarstellung sollte folgende Details enthalten (siehe auch [RFC1422], [DFN97]):

3.4.3.1 Eigene Identität

Zur Identität der Zertifizierungsstelle gehören alle zur Kontaktaufnahme nützlichen Daten:

- Eindeutiger Name (DN = Distinguished Name) in einem für elektronische Kommunikation geeigneten Adreßraum
- postalische Adresse
- Name der Organisation
- Geschäftsführer und Ansprechpartner
- Telefon− und Fax−Nummer
- E−Mail−Adressen
- Adressen der elektronischen Informationsdienste (WWW, FTP etc.)

3.4.3.2 Zuständigkeitsbereich

Der Zuständigkeitsbereich legt fest, welche Objekte von der CA zertifiziert werden. Eine CA kann beispielsweise ihre Aktivitäten auf die Zertifizierung anderer Zertifizierungsstellen einschränken oder nur Objekte einer bestimmten Organisation zertifizieren, etc. Zu zertifizierende Objekte können sowohl Personen als auch z.B. Server sein, welche dann durch zuständige Personen repräsentiert werden sollten.

3.4.3.3 Sicherheit und Datenschutz

Es muß beschrieben werden, welche technischen und prozeduralen Voraussetzungen erfüllt sein müssen, um das geforderte Maß an Sicherheit zu gewährleisten. Ein solche Darstellung muß enthalten:

- Verwendete Hardware und Software
- Verfahrensweise beim Datentransport von Schlüsseldaten
- Erzeugung von Schlüsselpaaren
- Speicherung der Schlüsseldaten
- Mindestlänge von Schlüsseln
- Maßnahmen zum Datenschutz
- Dokumentation
- Technische und prozedurale Anforderungen an die zu zertifizierenden Objekte

3.4.3.4 Zertifizierungsregeln

Die Zertifizierungsregeln beschreiben technische und organisatorische Richtlinien und Prozeduren für den Zertifizierungsvorgang sowie letztlich die eigentliche Aussage und den Verwendungszweck verschiedener Zertifikattypen. Dies umfaßt:

- Die zugrundeliegende bzw. unterstützte Zertifizierungsinfrastruktur (PKI − Abschnitt 3.3).
- Aufstellung von Zertifikattypen und deren Semantik und u.U. auch deren Verwendungszweck
- Regeln zur eindeutigen Namensgebung für zu zertifizierende Objekte, insbesondere auch Verfahren zur Auflösung von Namenskonflikten
- den Namen zugewiesene Semantik
- Verfahren zur Schlüsselerzeugung
- Verfahren zur Überprüfung der Identität zu zertifizierender Objekte
- Bereitstellung der ausgestellten Zertifikate an das zertifizierte Objekt
- maximale Gültigkeitsdauer von Zertifikaten
- Verschlüsselungsverfahren, Protokolle und Datenstruktur der Zertifikate

3.4.3.5 Zertifikatmanagment

Das Zertifikatmanagment umfaßt die Verteilung der ausgestellten Zertifikate und Verfahren des Widerrufs von abgelaufenen bzw. zurückgenommenen Zertifikaten. Dementsprechend muß die Policy folgende Informationen enthalten:

- Öffentliche Bereitstellung der Zertifikate (z.B. auf WWW−Servern, via Mail−Server bzw. Verzeichnisdiensten)
- Datenschutzrechtliche Maßnahmen
- Regeln für den Widerruf eines Zertifikates durch die Zertifizierungsstelle
- Verfahren zum Widerruf eines Zertifikates durch den Eigentümer
- Bereitstellung von Zertifikatwiderruflisten (Certificate Revocation Lists, kurz CRLs)
- Häufigkeit der Erzeugung und Bereitstellung neuer Zertifikatwiderruflisten (CRLs)

3.4.4 Verbreitung von CA−Schlüsseln

Um die eigene Vertrauenswürdigkeit zu stärken, müssen Zertifizierungsstellen ihren öffentlichen, selbst unterschriebenen Schlüssel auf möglichst viele Arten breit gestreut verteilen. Selbstverständlich kann eine CA sich von einer übergeordneten CA zertifizieren lassen. Dies ist auch in den einschlägigen Entwürfen so vorgesehen (vergleiche Abschnitt 3.3.8.1). Es verlagert sich damit aber nur das Problem einer initialen Schlüsselübermittlung.

3.4.4.1 Bereitstellung über Internet−Dienste

Man kann öffentliche Schlüssel via WWW, FTP, Mail, News oder Verzeichnisdienste bereitstellen. Dies ist technisch einfach und kostengünstig zu realisieren, birgt aber mit das höchste Risiko gegen einen Angriff der Art *Man−in−the−middle−attack.*
Für den Anwender ist diese Methode sehr einfach zu handhaben. Netscape hat bereits einen speziellen MIME−Typ `application/x-x509-ca-cert` für das selbstsignierte Zertifikat einer Zertifizierungsstelle im X509.v3−Format definiert (siehe [Nets97e]). Greift der Netscape Navigator via HTTP auf eine entsprechende Datei zu, so erscheint sofort ein Dialog, welcher den Benutzer fragt, ob und wie weit er der im Zertifikat angegebenen Zertifizierungsstelle vertrauen möchte.

3.4.4.2 Datenträger

Vertreibt eine Organisation sowieso schon Daten, Informationen oder Programme auf Datenträgern, so bietet es sich an, darauf auch zusätzlich kryptographische Schlüsseldaten abzulegen. Vorzugsweise sollte der Datenträger nur lesbar sein (z.B. CD−ROM) und die Herstellung sorgfältig nachgeprüft werden. Gelangt der Datenträger auf gut kontrollierten Vertriebswegen zum Anwender, so kann dieser relativ sicher gehen, der Echtheit des Schlüssels vertrauen zu können. Natürlich kann auch ein Datenträger sehr leicht gefälscht werden und ein offizielles Aussehen imitiert werden.

3.4.4.3 Als Bestandteil der Client−Software

Man kann den öffentlichen Schlüssel als festen Bestandteil eines Benutzer−Programms bereitstellen. Z.B. sind im Netscape Navigator und dem Internet Explorer von Microsoft bereits die öffentlichen Schlüssel mehrerer wohlbekannter Zertifizierungsstellen enthalten, um die Startpunkte für eine globale PKI zu verbreiten. Hierbei gibt es aber auch Probleme: Wird die Client−Software im Quell−Code verbreitet, so ist eine feste Zertifikatspeicherung sehr leicht zu manipulieren. Selbst in einer Binärdatei eines ausführbaren Programms lassen sich die öffentlichen Schlüssel finden und manipulieren. Ein gewissenhafter Benutzer sollte seine Client−Software also nur aus autorisierten Vertriebswegen beziehen und die Zertifikate aller enthaltenen Zertifizierungsstellen nochmals mittels weiterer Informationsquellen überprüfen.

3.4.4.4 Publikation in Printmedien

Eine Zertifizierungsstelle kann ihren öffentlichen Schlüssel bzw. dessen Fingerprint in gedruckter Form veröffentlichen. Es bieten sich hierfür mehrere Möglichkeiten, den öffentlichen Schlüssel abzudrucken:

- Visitenkarten/Ausweise
- Briefpapier
- Anzeigen in Tageszeitungen und Zeitschriften
- Prospekte oder sonstiges Werbematerial
- Firmenstempel

Die Verteilung von Printmedien ist sehr viel aufwendiger und kostspieliger als beispielsweise eine Verbreitung über Internet−Dienste, was aber meist als Nachteil nicht so schwer wiegt, da bei einer Firma sowieso Informationen in o.g. Form publiziert werden. Man kann also einfach kryptographische Daten zu bestehenden Drucksachen hinzufügen.
Die Handhabung der Daten in gedruckter Form ist für den Benutzer vergleichsweise umständlich. Deswegen sollte immer nur der Fingerprint des Zertifizierungsschlüssels in Printmedien veröffentlicht werden, um dem Benutzer die Überprüfung des auf elektronischem Wege bezogenen Schlüssels zu ermöglichen.
Diese Verbreitungsform erscheint zwar vergleichsweise sicher, da Fälschungen aufwendig sind und die Fälschung durch gesetzliche Bestimmungen unter Strafe gestellt ist, aber natürlich ist auch hier keine absolute Sicherheit gegeben. Es herrschen dieselben Risiken vor wie bei gedruckten Dokumenten im allgemeinen.

Ein Benutzer, welcher also die Identität einer Zertifizierungsstelle zweifelsfrei überprüfen möchte, sollte sich den öffentlichen Schlüssel auf möglichst viele Arten beschaffen. Dabei hängt die beim Schlüsselaustausch einzuhaltende Sorgfalt stark davon ab, wie sensitiv die betreffende Kommunikationsbeziehung ist. Gerade beim Akzeptieren eines selbstsignierten Zertifikates einer Zertifizierungsstelle sollten aber alle Beteiligten der zuverlässigen Übermittlung höchste Aufmerksamkeit zukommen lassen, da allen von dieser Zertifizierungsstelle ausgestellten Zertifikaten mehr oder minder automatisch vertraut wird.
Im Prinzip könnte auch jeder Endanwender seine eigenen Schlüssel auf die oben beschriebene Weise verteilen. Der Aufwand ist aber meist für die beteiligten Kommunikationspartner viel zu hoch, so daß besser eine Zertifizierungsstelle als Stellvertreter in Anspruch genommen wird.

3.4.5 Zertifizierungsaufgaben

Nimmt der Anwender die Dienste einer Zertifizierungsstelle in Anspruch, so stellt sich die Frage, wie der Ablauf des Zertifizierungsverfahren aussehen soll. Das Zertifizierungsverfahren umfaßt:

- Schlüsselerzeugung für das zu zertifizierende Objekt
- Übermittlung des Schlüssels an die Zertifizierungsstelle
- Überprüfen der Identität des zu zertifizierenden Objekts durch die Zertifizierungsstelle
- Bereitstellung des Zertifikats an das zertifizierte Objekt

- Widerruf von Zertifikaten durch die Zertifizierungsstelle oder das zertifizierte Objekt
- Geeignete Dokumentation aller Vorgänge

3.4.5.1 Schlüsselerzeugung

Naheliegend ist erst einmal, daß der spätere Zertifikatinhaber den privaten Schlüssel erzeugt und dann von der Zertifizierungsstelle unterschreiben läßt. Der öffentliche Schlüssel und weitere personenbezogene Daten des Anwenders können dann mithilfe des öffentlichen CA−Schlüssels verschlüsselt und an die Zertifizierungsstelle übermittelt werden.
Ebenso kann aber auch die Zertifizierungsstelle selbst Schlüsselpaare für den Anwender erzeugen. Die Schlüsselpaare können dann auf geeigneten Datenträgern dem Anwender übermittelt werden. Insbesondere die Ausgabe von SmartCards o.ä. ist oft nur durch die Zertifizierungsstelle möglich, weil beim Anwender zumeist ausreichendes Wissen und geeignete Hardware fehlen.
Erfolgt die Erzeugung privater Schlüssel seitens der Zertifizierungsstelle, so sollte diese die erzeugten privaten Schlüssel nach der Ausgabe an den Benutzer wieder zuverlässig löschen, um einen späteren Mißbrauch der privaten Schlüsseldaten zu vermeiden.
Man könnte sich auch vorstellen, daß die Zertifizierungsstelle im Einverständnis mit dem zu zertifizierenden Objekt Kopien der privaten Schlüssel zur Datensicherung vorhält, was aber dem Verwendungszweck, der unmißverständlichen Authentifizierung von Objekten, zuwider läuft. Auch sind natürlich größere Ansammlungen von privaten Schlüsseln für einen möglichen Angreifer sehr interessant. Man muß die Sorgfalt im Umgang mit dem privaten Schlüssel also einzig und allein dem Benutzer auferlegen.

3.4.5.2 Überprüfen der Identität

Speziell wenn das Schlüsselpaar auf Benutzerseite erzeugt wird, kann man den übermittelten Daten erst einmal nicht vertrauen. In den gängigen Formaten für Zertifikatsanforderungen sind Datenfelder zur Beschreibung der Identität des Schlüsselinhabers enthalten, deren Wahrheitsgehalt von der Zertifizierungsstelle geprüft werden muß.
Möglichkeiten zur Identitätsprüfung sind:

1. Persönliches Treffen des zu zertifizierenden Objekts (bzw. dessen Repräsentanten) und der Zertifizierungsinstanz (bzw. deren Repräsentanten) zur Übergabe des öffentlichen Schlüsselteils, evtl. mit Vorlage von Personalausweis, Firmenausweis o. ä.
2. Übermittlung der Zertifikatsanforderung durch vertrauenswürdige Dritte, z.B. andere Zertifizierungsinstanzen oder regionale Autoritäten (RA)
3. Post−Ident−Verfahren: Die zu zertifizierende Person muß auf einem Postamt ihren Personalausweis vorlegen, die Identitätsangaben werden per Einschreiben an einen Empfänger übermittelt.
4. Rückruf über Telefon
5. Zusätzliche Einsendung des Fingerprints durch das zu zertifizierende Objekt (bzw. dessen Repräsentanten) auf anderem Weg (out−of−band, z.B. Fax, Briefpost o.ä.)
6. Überprüfung durch (automatisierten) Mail−Dialog
7. Überprüfung durch Herkunftsmerkmale (z.B. IP−Quelladressen, Mailabsender etc.)

Die Sorgfalt bei der Identitätsprüfung hängt natürlich davon ab, wie weit später dem Zertifikat vertraut werden soll. Soll dem Zertifikat sehr weitgehend vertraut werden, so kommen eigentlich nur die ersten beiden Möglichkeiten in Betracht.

Eine Ausnahme sind die ebenfalls in [RFC1422] vorgeschlagenen PERSONA−Zertifikate. Eine Zertifizierungsstelle (mit entsprechender Zertifizierungsrichtlinie) stellt den zertifizierten Objekten lediglich ein anonymes Zertifikat für verschlüsselte Mail zur Verfügung, wobei die Identität explizit nicht geprüft wird. Solche Zertifikate können von einer Zertifizierungsstelle auch halbautomatisch ausgestellt werden, indem man lediglich die Funktionsfähigkeit der angegeben Mail−Adresse prüft (z.B. PERSONA−Zertifikate von Verisign). Solche Zertifikate sollten aber nicht zum autorisierten Zugang zu vertraulichen Daten oder zur Authentifizierung eines Mail−Absenders benutzt werden. Zur Verschlüsselung von Mails zum Schutz vor unautorisiertem Mitlesen sind sie aber besser als gar keine Sicherheitsmaßnahme.

3.4.5.3 Zertifikat ausstellen

Das eigentliche Zertifikat auszustellen, ist ein sicherheitskritischer Vorgang, da hier der private Schlüssel der Zertifizierungsstelle zum Einsatz kommt. Es ist also Sorge zu tragen, daß dieser Vorgang in einer sicheren Umgebung stattfindet (siehe Abschnitt 3.4.6).

3.4.5.4 Bereitstellung der Zertifikate und Widerruflisten

Die Schlüsselzertifikate und Widerrufslisten sollten für andere Benutzer leicht aufzufinden sein. Es bieten sich mehrere Wege der Bereitstellung über Internet an:

- Bereitstellung über WWW, entweder direkt als einzelne Dateien oder mittels Abfrage über ein CGI−BIN−Programm (evtl. mit Datenbank oder Zugriff auf einen anderen Dienst)
- Mail−Server mit automatischer Antwort auf bestimmte Abfragebefehle
- Komplette Zertifikat−/Widerrufslisten auf einem FTP−Server
- Öffentliche Verzeichnisdienste wie X.500 und LDAP

Unter Umständen muß der Zugriff auf solche Schlüsselzertifikate ebenfalls auf einen bestimmten Nutzerkreis eingeschränkt werden bzw. ein Zertifikatinhaber möchte seinen Schlüssel gar nicht veröffentlichen, da das Zertifikat auch personenbezogene Daten enthält.

Netscape hat eigene MIME−Typen für die Übermittlung von Zertifikaten im X.509v3−Format definiert, mithilfe derer Zertifikate sehr einfach in die Zertifikatdatenbank im Netscape Navigator eingebracht werden können (siehe auch [Nets97e]):

- CA−Zertifikate: *application/x−x509−ca−cert*
- Benutzer−Zertifikate: *application/x−x509−user−cert*
- Server−Zertifikate: *application/x−x509−server−cert*

Erhält man eine Mail mit einem Anhang bzw. lädt man eine Datei via HTTP mit einem dieser MIME−Typen, so erscheint ein entsprechender Dialog, welcher einen dazu auffordert, zu entscheiden, ob man dem angegebenen Zertifikat vertrauen möchte. Man kann dabei auch die Zertifikate gleich mit dem jeweiligen CA−Zertifikat überprüfen lassen (entsprechend den Vorschlägen in [RFC1422] und [RSA95]).

3.4.5.5 Widerruf von Zertifikaten

Ein Widerruf von Zertifikaten kann von der Zertifizierungsstelle, dem zertifizierten Objekt oder vertrauenswürdigen Dritten ausgehen. Es ist Aufgabe der Zertifizierungsstelle, abgelaufene Zertifikate regelmäßig auf einer Zertifikatwiderrufliste (siehe Abschnitt 6.7.3.1.2) zu veröffentlichen. Diese Zertifikatwiderruflisten sollten über längere Zeiträume archiviert werden, damit man auch nachträglich noch die Gültigkeit eines Zertifikats zu einem bestimmten Zeitpunkt prüfen kann.

3.4.5.6 Dokumentation

Gegebenenfalls sollte die Zertifizierungsstelle die Einhaltung der selbst auferlegten Regelungen auch von unabhängigen Gutachtern überprüfen (auditieren) lassen. Zu diesem Zweck müssen alle Vorgänge genau dokumentiert werden. Zum einen sollte die verwendete Software die Dokumentation der durchgeführten Schritte anhand von Protokolldateien unterstützen, und zum anderen sollten die CA−Mitarbeiter zusätzlich schriftliche Protokolle mit persönlicher Unterschrift anfertigen. Die schriftliche Protokollierung sollte mit entsprechenden Formularen für die verschiedenen Arbeitsschritte unterstützt bzw. formalisiert werden[11].

3.4.6 Handhabung des CA−Schlüssels

Wird der private Schlüssel der Zertifizierungsstelle kompromittiert, so sind alle ausgestellten Zertifikate wertlos. Die Handhabung des CA−Schlüssels ist also mit höchster Sorgfalt zu organisieren. Insbesondere kann die Benutzung des CA−Schlüssels auch nicht automatisiert erfolgen. Verschiedene Aspekte sind zu beachten:

- **Speicherung des CA−Schlüssels**
  Der private Schlüsselteil des CA−Schlüssels muß irgendwo gespeichert werden. Er darf nicht verloren gehen und nicht von Unbefugten kopiert werden können. Zum einen muß man also für ausreichende Datensicherung der Schlüsseldaten sorgen, zum anderen sind alle erstellten Kopien des CA−Schlüssels als streng vertraulich einzustufen und der Zugang dementsprechend zu sichern.
- **Benutzerkreis veränderlich**
  Personen, welche mit dem Ausstellen von Zertifikaten betraut sind, dann aber die Organisation der Zertifizierungsstelle verlassen, könnten sich eine Kopie des CA−Schlüssel erstellt haben.
- **Benutzung des CA−Schlüssels**
  Zum Ausstellen der Zertifikate muß der CA−Schlüssel benutzt werden. So selbstverständlich sich dies auch anhört: Dieser Vorgang ist kritisch, da der CA−Schlüssel im Moment der Benutzung im Klartext vorliegen muß.

In folgenden Unterabschnitten werden mögliche Sicherheitsvorkehrungen beim Umgang mit dem CA−Schlüssel beschrieben.

3.4.6.1 Paßwortschutz

Die einfachste Sicherung für den CA−Schlüssel ist die Verschlüsselung mit einem möglichst langen Kennwort. Der Schlüssel liegt nur im Moment der Benutzung im Klartext vor, alle gespeicherten Kopien

[11] Die Protokollierung der Arbeitsschritte ist insbesondere auch für eine Einbindung der Verfahren in ein Qualitätsmanagment und die Zertifizierung nach ISO 9000 erforderlich.

sind verschlüsselt. Das Anfertigen von Sicherheitskopien ist einfach, jedoch sollten diese Sicherheitskopien auch geschützt aufbewahrt werden (z.B. in einem Tresor, Bankschließfach o.ä.).
Schon diese einfache Maßnahme unterbindet jegliche automatisierte Benutzung des CA−Schlüssels, da das Kennwort niemals in irgendeiner Datei gespeichert werden darf.
Der CA−Schlüssel liegt während der Benutzung im Klartext vor, meist im Arbeitsspeicher des Rechners, auf welchem das Ausstellen des Zertifikates erfolgt. Es ist daher dafür Sorge zu tragen, daß diese Speicherbereiche streng gesichert sind. Ist die verschlüsselte Speicherung nur von einem Kennwort abhängig, so kann sich eine Person mit Kenntnis des Kennworts unbemerkt Kopien des privaten CA−Schlüssels anfertigen und veruntreuen.

3.4.6.2 Shared Secrets

Möchte man vermeiden, daß eine mit dem Ausstellen von Zertifikaten betraute Person sich Kopien des privaten CA−Schlüssels anfertigt und veruntreut, so kann man das Prinzip der Shared Secrets verwenden [Schn96]. Dabei besteht das Geheimnis zum Entschlüsseln oder zur Benutzung des CA−Schlüssels aus mehreren, teilweise redundanten Teilgeheimnissen, welche zur selben Zeit, aber nicht unbedingt am selben Ort, zusammenkommen müssen. Beispielsweise haben mehrere Personen jeweils ein Kennwort oder eine spezielle Hardware und müssen zusammen den privaten CA−Schlüssel entschlüsseln.

3.4.6.3 Anforderungen an die Hardware

Da der private CA−Schlüssel bei der Benutzung im Klartext vorliegt, sollte auf diesen Speicherungsort kein unautorisierter Zugriff möglich sein. Verschiedene Grundvoraussetzungen sollten bei der verwendeten Hardware erfüllt sein:

- Kein physikalischer Zugang für nicht autorisierte Personen, damit keine unerlaubten Umbauten an der Hardware (z.B. Tastaturlauscher, Sender etc.) vorgenommen werden können. Man kann beispielsweise einen getrennten, entsprechend gesicherten Raum vorsehen, zu dem nur diejenigen Personen Zutritt haben, welche mit dem Ausstellen der Zertifikate betraut sind. Oder man lagert die Hardware in einem Tresor und entnimmt sie nur für die Zeitpunkte der Benutzung.
- Es sollten keine vernetzten Rechner verwendet werden, da diese nicht wirklich zuverlässig zu sichern sind.
- Die Software sollte nach der Installation immer wieder mit kryptographischen Mitteln überprüft werden, so daß Unbefugte keine Veränderungen an der installierten Software vornehmen können.

3.4.6.4 Spezielle kryptographische Hardware

Um die recht vielfältigen Probleme bei der Sicherung von üblichen Hardware−/Software−Installationen zu vermeiden, gibt es spezielle kryptographische Hardware. Das CA−Schlüsselpaar wird in dieser einbruchssicheren Hardware erzeugt, und jegliche Benutzung des privaten CA−Schlüssels erfolgt innerhalb einer solchen Krypto−Box, d.h. der private CA−Schlüssel verläßt diese Hardware zu keinem Zeitpunkt. Die Kosten für solche Hardware und der Aufwand für eine ausfallsichere Lösung sind allerdings erheblich. Inzwischen gibt es solche Krypto−Hardware auch in Form von Chip−Karten, welche natürlich nicht einbruchssicher, dafür aber leichter zu handhaben und billiger sind.

3.5 Kryptographische Protokolle

Als *kryptographisches Protokoll* wird ein Kommunikationsprotokoll bezeichnet, welches als Mechanismen die Repräsentation der Daten, verwendete kryptographische Algorithmen oder Verfahren

für den Schlüsselaustausch spezifiziert. In diesem Abschnitt werden verschiedene kryptographische Protokolle vorgestellt, welche auch im weiteren Verlauf der Arbeit zum Einsatz kommen:

- *PKCS* enthält Definitionen von Datenformaten zum Versenden und Speichern kryptographischer Daten.
- *IPSec* dient der Sicherung von IP−Paketen, demnach also ein kryptographisches Protokoll der Netzwerk−Schicht.
- *SSL* ist für die Sicherung von Client−/Server−Zugriffen gedacht und ist ein kryptographisches Protokoll der Transport−Schicht.
- *S/MIME* definiert einen Standard für verschlüsselte und signierte E−Mail−Nachrichten, ein Beispiel für ein kryptographisches Protokoll der Anwendungs−Schicht.

3.5.1 PKCS

PKCS definiert mehrere Standards für unterschiedliche Aufgaben bei der Nutzung von Public−Key−Kryptographie. Es werden z.B. Standards für die Speicherung von Zertifikatdaten, Formate für signierte Nachrichten und die Nutzung von Krypto−Hardware definiert [PKCSov].
Derzeit von praktischer Bedeutung für den Anwender sind folgende Teile des PKCS−Standards:

- *PKCS#7 − Cryptographic Message Syntax Standard* [PKCS7]
  beschreibt eine allgemeine Syntax für Daten, auf welche kryptographische Verfahren angewandt wurden. Dies können digitale Signaturen (digital signatures) oder auch verschlüsselte Daten (digital envelopes) sein. PKCS#7−Daten können rekursiv geschachtelt werden, d.h. eine digitale Signatur kann wieder verschlüsselt werden, und es entsteht immer wieder eine gültige PKCS#7−Datenstruktur. Des weiteren werden zusätzliche Attribute, wie z.B. der Zeitpunkt des Signierens, mit dem Inhalt einer Nachricht festgehalten und authentifiziert. Zusätzlich können in PKCS#7−Daten Zertifikate und Zertifikatwiderruflisten enthalten sein.
- *PKCS#10 − Certification Request Syntax Standard* [PKCS10]
  beschreibt eine Syntax für Zertifikatanforderungen, zur Übermittlung eines vom Objekt selbst signierten Datensatzes an eine Zertifizierungsstelle. Ein PKCS#10−Datensatz besteht aus einem eindeutigen Namen, zusätzlichen Attributen und einem öffentlichen Schlüssel. [PKCS9] beschreibt mögliche Zusatzattribute, es können aber beliebige eigene Attribute definiert werden.
- *PKCS#11 − Cryptograhic Token Interface Standard* [PKCS11] (alias Cryptoki−API)
  definiert eine einheitliche Zugriffsmethode auf kryptographische Einheiten (cryptographic token). Solche kryptographischen Einheiten können sowohl in Software oder auch als getrennte Hardware ausgeführt sein und verschiedenste Aufgaben erfüllen, wie z.B. die Speicherung von kryptographischen Daten oder die Ausführung kryptographischer Algorithmen. PKCS#11 stellt also eine generische Abstraktionsschicht zwischen der kryptographischen Anwendung und den einzelnen kryptographischen Einheiten dar.
- *PKCS#12 − Personal Information Exchange Syntax Standard* [PKCS12]
  definiert eine Syntax für die zugriffsgesicherte Speicherung von Zertifikaten inkl. dem privaten Schlüssel. Zur Zugriffssicherung wird vom Zertifikat− und Schlüsselinhaber ein Kennwort (Passphrase) vergeben, welches als Schlüssel zur Verschlüsselung des PKCS#12−Datensatzes dient.

Die Syntax von PKCS−Daten wird in ASN.1 definiert, die Daten werden als BER−kodierte Oktett−Strings dargestellt, welche für die verlustfreie Übertragung in den üblichen E−Mail−Systemen noch BASE64−kodiert werden müssen [Kal93c].

3.5.2 Virtual Private Networks mit IPSec

IPSec [IPSec] ist ein IETF−Entwurf zur Sicherung der Vertraulichkeit, Authentizität und Integrität auf IP−Ebene. Dies kann dazu benutzt werden, sog. virtuelle private Netze (VPN − Virtual Private Networks) aufzubauen, bei denen nicht die Endsysteme kryptographische Techniken direkt benutzen, sondern unsichere Teilstrecken zwischen Teilnetzen mit IPSec gesichert werden. Damit können beispielsweise interne Netze mehrerer Firmenstandorte über öffentliche Leitungen zu einem Netzwerk zusammengeschlossen werden, ohne daß die Vertraulichkeit, Authentizität und Integrität der übertragenen Daten gefährdet wird. Abbildung 13 soll dies verdeutlichen.

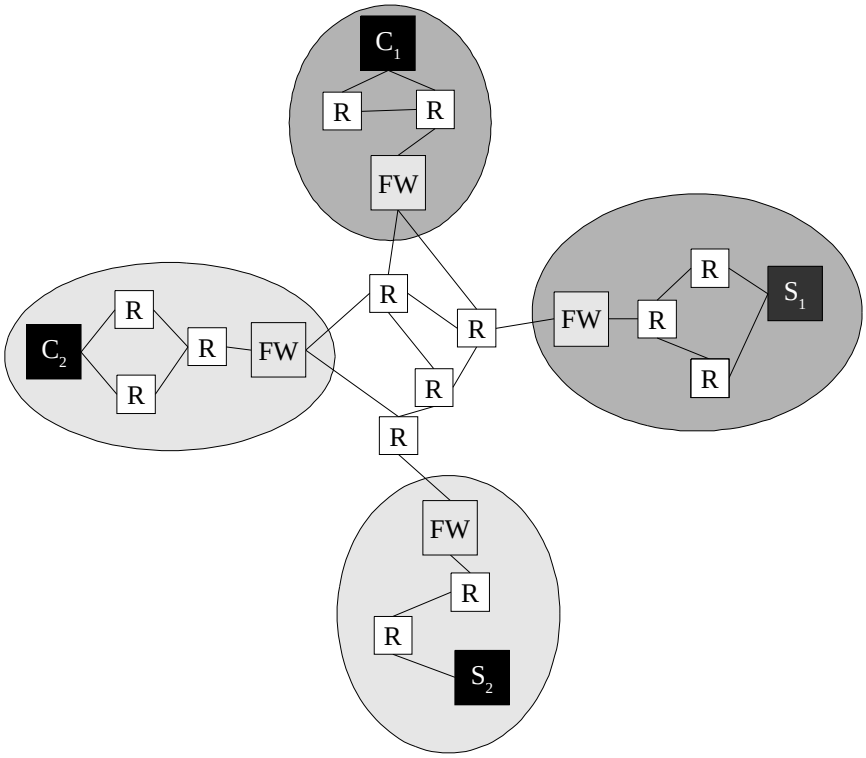


Abbildung 13: Beispiel einer Nutzung von Virtual Private Networks

In diesem Beispiel kommunizieren Client C_1 mit Server S_1 und Client C_2 mit Server S_2 derart, als ob sie in jeweils einem gemeinsamen Netzwerk liegen würden. Die Verbindung kann zwischen den "Firewalls" FW mittels IPSec gegen Abhören und Verfälschen gesichert werden. Die "Firewall"−Systeme können und müssen ferner sicherstellen, daß z.B. nicht Client C_1 auf Server S_2 zugreift, da sie unterschiedlichen virtuellen Netzen angehören.

Da IPSec als verbindungsloses Protokoll der Netzwerkschicht operiert, ist die Integrität und Authentizität immer nur paketweise gesichert (im Gegensatz z.B. zu SSL, siehe Abschnitt 3.5.3).

IPSec definiert folgende Protokolle:

- *IP Authentication Header* (AH) sichert durch einen zusätzlichen Authentifizierungskopf die Datenintegrität, Senderauthentizität auf IP−Ebene und bietet einen Schutz gegen Replay−Attacken.
- *IP Encapsulating Security Payload* (ESP) sichert durch Verschlüsselung und einen zusätzlichen Authentifizierungskopf die Vertraulichkeit, Datenintegrität, Authentizität der Sender−IP−Adresse und bietet einen Schutz gegen Replay−Attacken.
- *Internet Security Association and Key Management Protocol* (ISAKMP) stellt Methoden für das Verbindungs− und Schlüsselmanagement bereit.

Die beiden Protokollarten AH und ESP können in jeweils zwei verschiedenen Modi betrieben werden:

- *Transport−Mode*
  Bei dieser Betriebsart wird ein bestehendes IP−Paket verändert. Dieser Modus ist primär zur direkten Verwendung durch Endsysteme gedacht.
- *Tunnel−Mode*
  Bei dieser Betriebsart wird ein bestehendes IP−Paket als IP−Datenteil gemäß IP−IP−Tunneling in ein neues IP−Paket gepackt. Dieser Modus ist für die Paketweiterleitung in Zwischensystemen gedacht, also z.B. zwischen zwei Firewall−Systemen, welche eine VPN−Verbindung realisieren.

Es ist dabei möglich, daß z.B. ein Client und ein Server ihre Verbindung mittels Transport−Mode sichern, aber gleichzeitig diese Verbindung von zwei Zwischensystemen in eine Tunnel−Mode−Verbindung eingebettet wird.

IPSec ist sehr nützlich, um unsichere Teilstrecken gegen Abhören und Manipulationen zu schützen, vor allem in Situationen, in denen die End−Systeme bzw. Anwendungen keine Verschlüsselung beherrschen. Bei sorgfältiger Planung und Installation bietet diese Technik die Möglichkeit der Authentifizierung von IP−Adressen (Sicherung gegen IP−Spoofing), welche wiederum eine Voraussetzung für den sicheren Einsatz von IP−Paketfilterregeln ist (vergleiche Abschnitt 2.3.2). Zudem können andere Protokolle in IPSec eingebettet werden, wie z.B. IPX in IP, welches wiederum mit IPSec gesichert wird. Diese Möglichkeiten sind vor allem bei räumlich weit verteilten Netzen reizvoll, da durch die Nutzung des Internet zur LAN−LAN−Koppelung die Kommunikationskosten dramatisch gesenkt werden können.
Protokolle auf dieser Ebene können natürlich keine Authentifizierung von Benutzern bereitstellen. Auch sollte man beim Einsatz von IPSec beachten, daß die IPSec−Router von einem VPN durchaus ein lohnendes Angriffsziel darstellen.

3.5.3 SSL

SSL ist als eine transparente, flexible Protokollschicht zwischen den Protokollen der Anwendungsschicht und einem gesicherten, verbindungsorientierten Protokoll gedacht [Nets98a]. Es dient der Sicherung von Online−Verbindungen zwischen einem Client und einem Server. Meist wird es als kryptographische Schicht zwischen einem Internet−Dienst−Protokoll (z.B. HTTP, POP3, NNTP, LDAP etc.) und dem TCP−Protokoll verwendet (Abbildung 14).

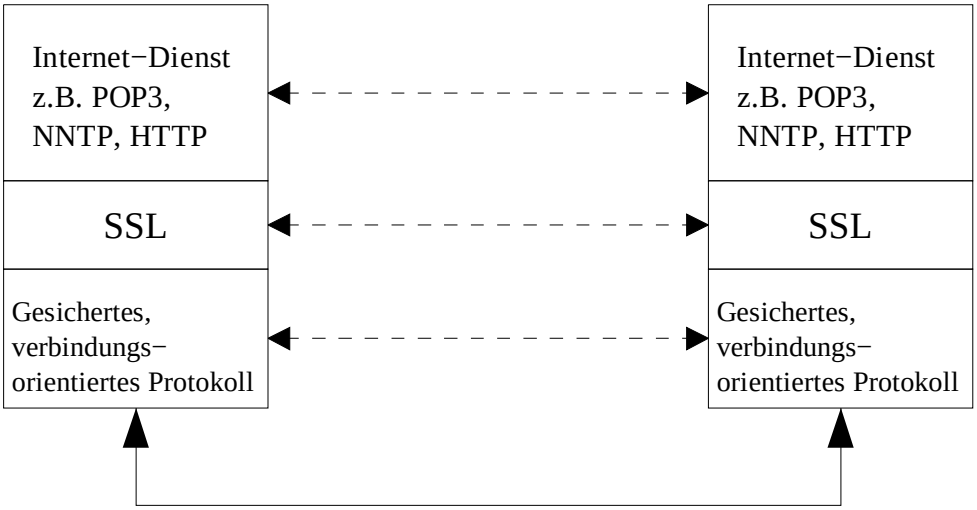


Abbildung 14: Das SSL−Protokoll als Schicht zwischen Anwendungs− und Transportschicht

SSL−Verbindungen werden 1:1 auf Verbindungen des darunterliegenden Protokolls abgebildet, da SSL keinerlei eigene Protokollmechanismen zur Sicherung einer Verbindung gegen Paketverluste enthält. Daher kann SSL nicht auf ein verbindungsloses Datagramm−Protokoll wie z.B. IP oder UDP aufgesetzt werden. Inzwischen liegt die dritte Version SSLv3 vor, welche als Vorbild für das von der *IETF* (Internet Engineering Task Force) spezifizierte Protokoll *TLS* (Transport Layer Security) diente [TLS]. Es werden an dieser Stelle kurz die für den praktischen Einsatz relevanten SSL−Eigenschaften vorgestellt. Eine detailliertere Beschreibung von SSL findet man in der Protokollspezifikation von Netscape [Nets97b].

Verwendete kryptographische Algorithmen

Beide Kommunikationspartner müssen in einer SSL−Verbindung jeweils die selben kryptographischen Algorithmen beherrschen. SSL selbst schreibt die verwendeten Algorithmen nicht explizit vor, sondern es werden beim SSL−Verbindungsaufbau, dem sog. *SSL Hello*, von beiden SSL−Endsystemen die gemeinsam verwendeten Algorithmen ausgehandelt. Dadurch ist es möglich, auch neu entwickelte Algorithmen zu verwenden, ohne das SSL−Protokoll selbst zu verändern. Der Server kann die Menge der möglicherweise verwendeten Algorithmen einschränken, was ermöglicht, auf Serverseite bestimmte Algorithmen und Mindestschlüssellängen für die SSL−Verbindung vorzugeben.

Sicherung der Authentizität und Integrität

Die Authentizität und Integrität wird mithilfe von X.509v3−Zertifikaten gesichert, d.h. SSL sichert eine Verbindung bei korrekter Anwendung gegen *Man−in−the−Middle−* und *Replay−*Angriffe. Es kann sich sowohl der Server als auch der Client mithilfe von Zertifikaten authentifizieren. Ein Server kann während des SSL−Verbindungsaufbaus vom Client ein Zertifikat anfordern. Durch die Serverkonfiguration kann festgelegt werden, ob ein Client−Zertifikat für eine SSL−Verbindung zwingend notwendig ist.

Separation der kryptographischen Aufgaben

SSL kann für die Verschlüsselung, Authentifizierung und Integritätssicherung jeweils verschiedene Algorithmen mit verschiedenen Schlüsseln verwenden. Dies ermöglicht den Einsatz in Umgebungen, in denen die geltende Gesetzeslage den Einsatz von Kryptographie reglementiert. Bespielsweise ist in Frankreich Verschlüsselung in öffentlichen Datennetzen verboten. SSL ermöglicht für diesen Fall eine unverschlüsselte Verbindung, bei der aber die Integrität der übertragenen Daten durch kryptographische Prüfsummen gesichert ist.

Effizienz
Speziell asymmetrische Verschlüsselung ist ein rechenintensiver Vorgang. Anstatt den gesamten SSL−Verbindungsaufbau mit Austausch symmetrischer Schlüssel bei jeder SSL−Verbindung neu durchzuführen, kann bei der ersten SSL−Verbindung zwischen einem Client und einem Server ein sog. *Master Secret* zwischengespeichert werden (SSL Cache), welches bei der nächsten SSL−Verbindung zwischen diesem Client−/Server−Paar verwendet wird und dann eine unmittelbare Nutzung von symmetrischer Verschlüsselung ohne vorherige asymmetrische Verschlüsselungsoperationen ermöglicht.
Protokollunabhängigkeit
SSL kann auf jedem verbindungsorientierten, gesicherten Übertragungsprotokoll aufsetzen. Meist wird dabei TCP verwendet, es kann aber auch X.25 oder ein OSI−Protokoll zum Einsatz kommen.
Datenkompression
SSL unterstützt das Komprimieren von Daten vor der Verschlüsselung, da Kompression bereits gut verschlüsselter Daten keinen Sinn macht. Derzeit gibt es aber keine SSL−Implementierung, welche von dieser Möglichkeit Gebrauch macht.

3.5.4 S/MIME

S/MIME soll die Vertraulichkeit, Integrität und Authenizität von E−Mail−Nachrichten sicherstellen. S/MIME basiert auf *MIME* (Multi−Purpose Internet Mail Extensions), dem Standard für mehrteilige Nachrichten [RFC2045][RFC2046][RFC2047][RFC2048][RFC2049], und erweitert diesen um Elemente zum Verschlüsseln und Signieren von Nachrichten [RFC2311][RFC2312]. Alle verschlüsselten Daten liegen bei S/MIME im PKCS#7−Format vor [RSA95]. Derzeit existieren einige Implementierungen für S/MIMEv2 (siehe auch Kapitel 5). Die neuere Version S/MIMEv3 hat derzeit noch Gegenstand eines Internet−Draft [SMIME].
Nachrichten im MIME−Format können mehrere Teile (Parts) enthalten, welche jeweils mit einem bestimmten MIME−Typ versehen sind. Am jeweiligen MIME−Typ kann ein MUA (Message User Agent − das Mail−Programm des Benutzers) feststellen, von welchem Datentyp ein Teil der Nachricht ist. Mehrteilige MIME−Nachrichten können auch wieder geschachtelt in einem MIME−Part untergebracht werden. S/MIME definiert weitere MIME−Typen für das Verschlüsseln und Signieren von Nachrichten:

- *application/x−pkcs7−mime*
  ist eine verschlüsselte Nachricht
- *application/x−pkcs7−signature*
  enthält die Signatur einer Nachricht

Unterzeichnet man eine Nachricht, so wird die Signatur als ein weiterer MIME−Part des Typs *application/x−pkcs7−signature* zu der Nachricht hinzugefügt. Möchte man eine verschlüsselte Nachricht versenden, so wird diese mit allen Teilen (inkl. Signatur) verschlüsselt und die verschlüsselten Daten im PKCS#7−Format in einen einzigen MIME−Part des Typs *application/x−pkcs7−mime* verpackt. Ein MUA auf Empfängerseite muß auf einer ersten Ebene die verschlüsselte Nachricht entschlüsseln, um an die weiteren MIME−Parts und insbesondere die Signatur zu kommen. Danach kann der MUA die Signatur prüfen. Das PKCS#7−Format enthält insbesondere auch die Informationen über die vor der Übertragung angewendeten kryptographischen Algorithmen und den mithilfe des asymmetrischen Algorithmus verschlüsselten Sitzungsschlüssel für den symmetrischen Algorithmus, mit dem die eigentliche Nachricht verschlüsselt wurde.

3.6 Zusammenfassung

In diesem Kapitel wurden die wichtigsten Grundbegriffe kryptographischer Techniken eingeführt, was das Verständnis der folgenden Kapitel erleichtern soll. Ferner wurden auch verschiedenartige Probleme diskutiert, die für einen erfolgreichen Einsatz kryptographischer Techniken noch allgemein oder jeweils im Einzelfall zu lösen sind.
Anzumerken ist, daß kryptographische Techniken keineswegs den Einsatz konventioneller Sicherheitsmechanismen überflüssig machen können. Vielmehr muß bei der Konfiguration auf das sinnvolle Ineinandergreifen der herkömmlichen und der kryptographischen Sicherheitsmechanismen geachtet werden. So nützt es wenig, wenn vertrauliche Daten zwar verschlüsselt übertragen werden, dann aber nur unzureichend oder gar nicht gesichert auf dem Zielsystem gespeichert werden.

4 Lage bei der Propack Data GmbH

In diesem Kapitel werden als erstes im Abschnitt 4.1 die als extern einzustufenden Kommunikationsbeziehungen erläutert und danach im Abschnitt 4.1.2 kurz die gewünschten Anwendungen dargestellt und die dafür erforderlichen Sicherheitsmechanismen genannt. Abschnitt 4.2 gibt einen Überblick über die bereits Software im Client− und Server−Bereich eingesetzte Software.

4.1 Externe Kommunikationsbeziehungen

Die zunehmend räumlich verteilten Aktivitäten bei der Propack Data GmbH machen vermehrt externe Kommunikation erforderlich, wobei die Benutzer möglichst transparent auf alle extern und intern verfügbaren Internet−Dienste, wie z.B. E−Mail, WWW, Datei−Bestände und Datenbanken zugreifen möchten. Eine interne Bestandsaufnahme ergab verschiedene Klassen externer Kommunikationsbeziehungen im Sinne der in Abschnitt 2.1.1 gemachten Definitionen. Abbildung 15 zeigt verschiedene mögliche externe Zugriffe.

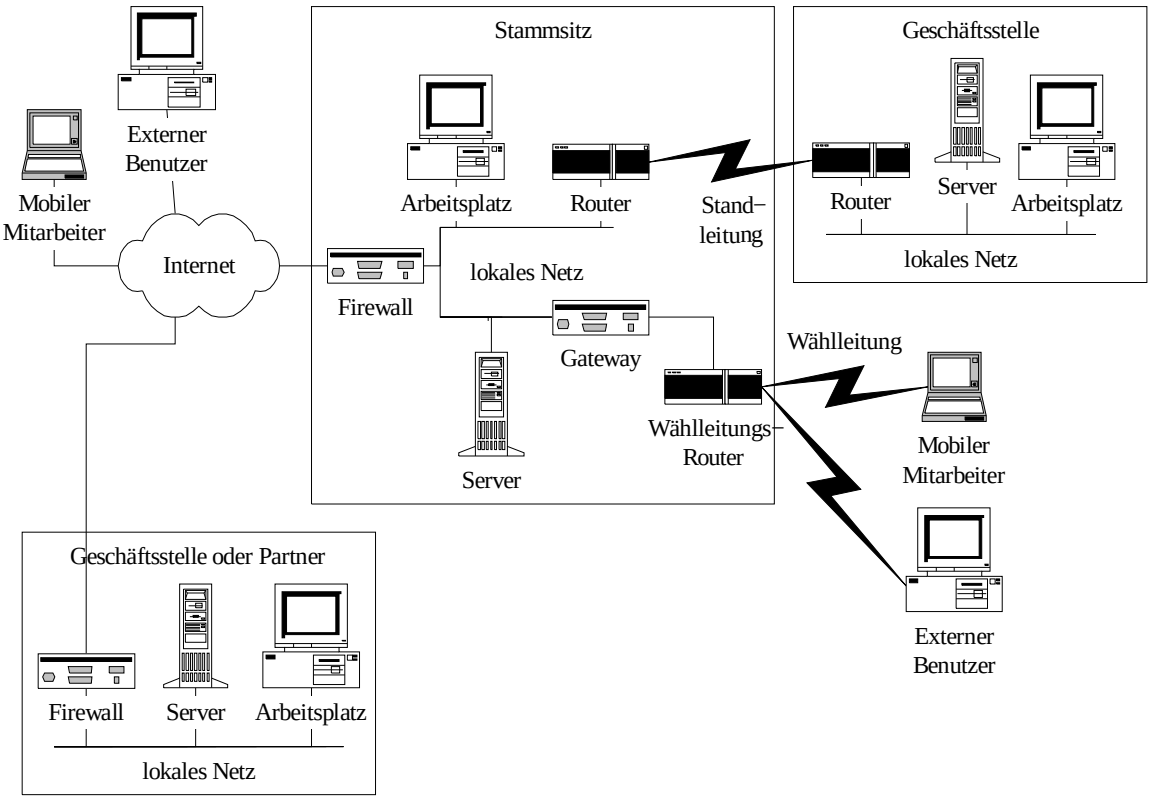


Abbildung 15: Übersicht der externen Kommunikationsbeziehungen

4.1.1 Benutzerklassen

Es erscheint sinnvoll, erst einmal die verschiedenen Benutzer zu klassifizieren, da sie sich hinsichtlich der räumlichen Verteilung, technischen Voraussetzungen, administrativen Zuständigkeiten und sicherheitstechnischen Kenntnissen stark unterscheiden.

4.1.1.1 Geschäftsstellen

Die Mitarbeiter in den *Geschäftsstellen* haben eine sehr enge Anbindung an das Stammhaus. Im Prinzip ist die Anbindung der Geschäftsstellen eine Erweiterung des internen Netzes. Da dies nicht ohne den Einsatz öffentlicher Leitungen zustande kommen kann, hat man bei solchen Leitungen ein Sicherheitsproblem bezüglich der Vertraulichkeit, Authentizität und Integrität der Daten. Angriffe auf Standleitungen sind zwar technisch vergleichsweise aufwendig. Da über solche Leitungen leicht interne Zugriffe erfolgen können, sind sie aber auch ein lohnendes Angriffsziel. Für die permanente Anbindung von Geschäftsstellen ist daher eine transparente Verschlüsselung auf Routing−Ebene (VPN − Virtual Private Network) vorgesehen.

4.1.1.2 Mobile Mitarbeiter

Speziell Angehörige der Vertriebs− und Support−Abteilungen sind sehr *mobile Mitarbeiter*, möchten aber trotzdem unterwegs wie gewohnt auf Informationen und Dienste im internen Netz zugreifen können. Als vordringlichster Wunsch wird dabei meist der Zugriff auf interne E−Mail genannt. Auch der Zugriff auf interne WWW−Inhalte und andere Server sind erwünscht, aber nicht für die tägliche Arbeit erforderlich.
Mobile Mitarbeiter sind meist mit firmeneigenen, tragbaren Rechnern (Notebook oder Laptop) unterwegs, so daß die Installation der für Sicherungsmaßnahmen benötigten Software wie bei internen Rechnern gehandhabt werden kann. Auch können die Mitarbeiter entsprechend geschult werden, und der Rechner befindet sich weitestgehend unter der persönlichen Aufsicht des Mitarbeiters.

4.1.1.3 Heimarbeitsplätze

Ähnliche Verhältnisse herrschen bei Mitarbeitern mit *Heimarbeitsplätzen*, wobei in diesem Fall der Wunsch nach engerer Anbindung und weitergehendem Zugriff noch größer als bei mobilen Mitarbeitern ist, da die tägliche Arbeit bei längerer Abwesenheit sonst nicht durchgeführt werden kann. Bei Heimarbeitsplätzen befindet sich ein ebenfalls von internen Administratoren eingerichteter Arbeitsplatz vor Ort bzw. die benötigte Software kann leicht auf dem Arbeitsplatzrechner installiert werden. Vor der Benutzung können und müssen die Mitarbeiter im Umgang mit der sicherheitsrelevanten Software entsprechend geschult werden.

4.1.1.4 Eigene Mitarbeiter in anderen Firmen

Zur Unterstützung des Aufbaus von komplexeren Installationen bei Kunden vor Ort sind einige Mitarbeiter der Propack Data GmbH oft monatelang in anderen Firmen untergebracht. Diese Mitarbeiter sollten genauso auf Informationen und Dienste im internen Netz zugreifen können wie interne Mitarbeiter. Als besonders vordringlich ist hier der konsequente Einsatz von kryptographischen Techniken für die Nutzung von E−Mail zu nennen.
Leider sind diese Mitarbeiter oft weitgehend auf eine von der Gastfirma betreute IT−Infrastruktur angewiesen und sind demnach noch nicht einmal frei in der Auswahl der Client−Software, da die

Gastfirmen oft Standard−Protokolle, wie z.B. POP3 zum Abholen von E−Mails, nicht anbieten. Auch der freie Zugriff auf Internet−Dienste ist oft nicht gegeben.
Gerade bei solchen Mitarbeitern ist auch nicht so gut die Möglichkeit gegeben, sie intensiv im Umgang mit bestimmten Kommunikationstechniken zu schulen, da sie meist nicht am Stammsitz anwesend sind, so daß selbst beim Einsatz kryptographischer Techniken ein vergleichsweise hohes Risiko durch unsachgemäße Anwendung bleibt.
Bei dieser Anwendergruppe ergibt sich aufgrund der o.g. Randbedingungen also die höchste Diskrepanz zwischen dem Anspruch der möglichst engen Anbindung und den sehr eingeschränkten technischen Möglichkeiten, effektive und einheitliche Sicherungsmaßnahmen einzuführen.

4.1.1.5 Kooperationspartner

Die Gruppe der *Kooperationspartner* umfaßt Firmen, welche entweder im Vertrieb oder bei der Software− und Projekt−Entwicklung mit der Propack Data GmbH eng kooperieren. Diese Zusammenarbeit erfordert, neben gesichertem Nachrichtenaustausch und Zugriffen auf interne WWW−Inhalte, teilweise Zugriffe bis auf Datei−Server mit Quelltexten, stellt also hohe Anforderungen an die verwendeten Sicherheitsmechanismen.
Die Probleme bezüglich der Software−Installation und Benutzerschulung bei den Kooperationspartnern sind ähnlich wie bei eigenen Mitarbeitern in anderen Firmen (Abschnitt 4.1.1.4). Erschwerend kommt aber hinzu, daß die Rechtevergabe ungleich komplizierter wird, da Kooperationspartner nicht Angehörige der Propack Data GmbH sind. Eine fein granulierte Zugriffskontrolle setzt aber eine entsprechende, ausgefeilte Strukturierung von Informationen voraus.

4.1.1.6 Kunden

U.U. möchte man *Kunden* eine Zugriffsmöglichkeit auf bestimmte interne WWW−Inhalte geben. Der Unterschied zu Kooperationspartnern ist der, daß die Informationen separat aufbereitet werden und die korrekte Zugriffskontrolle vergleichsweise einfach einzurichten ist, wenn eine zuverlässige Authentifizierung erfolgt. Allerdings läßt sich im realen Betrieb die Grenze zwischen Kooperationspartnern und Kunden nicht immer genau ziehen. Die Entwicklung in einer Geschäftsbeziehung ist an dieser Stelle oft sehr dynamisch. Desweiteren gelten die selben Probleme bezüglich der Verteilung und Installation der benötigten Software, wie bei Kooperationspartnern.

4.1.1.7 Anonyme Nutzer

Im Gegensatz zu allen vorherigen Benutzerklassen, die eine Authentifizierung der Benutzer erfordern, ist es nicht nötig, *anonyme Nutzer* zu erkennen, welche auf separat aufbereitete Informationen (z.B. Werbematerial via WWW, unverbindliche Anmeldung zu Workshops etc.) zugreifen.

4.1.2 Gewünschte Anwendungen

In diesem Abschnitt werden die bisher von den Benutzern gewünschten Klassen von Anwendungen dargestellt.

4.1.2.1 Dokumentenaustausch

Oft ist es erforderlich, Dokumente zur gemeinsamen Bearbeitung als Mail−Anhang (mail attachment) zu verschicken. Diese Dokumente enthalten häufig sensitive Informationen, welche selbstverständlich nicht

verfälscht und nur von autorisierten Personen eingesehen werden dürfen. Auch könnten Zugriffe auf firmeninterne Dokumente über extern zugängliche Datei−Server erwünscht sein.

4.1.2.2 Fehlerdatenbank

Es soll Kunden und Service−Personal vor Ort ermöglicht werden, über den WWW−Dienst auf eine Fehlerdatenbank zuzugreifen. Damit soll die Fehlererfassung und Informationsverteilung zur Fehlerbehebung verbessert werden. Diese Informationen sind teilweise kundenspezifisch, es muß also eine sichere Authentifizierung und eine Autorisierung erfolgen.

4.1.2.3 Software−Updates

Oftmals ließe sich viel Wartungsaufwand einsparen, wenn der Kunde gezielt Software−Updates von einem Server laden und nach Anleitung selbst installieren könnte. Diese Software−Updates sollten vor allem unverändert übertragen werden. Enthalten die Programmteile kundenspezifische Anpassungen, so sind sie außerdem u.U. auch gegen Mitlesen zu schützen.

4.1.2.4 Online−Bestellung

Es soll Kunden ermöglicht werden, Hardware oder Software−Lizenzen direkt via WWW−Zugriff zu bestellen. Die Zahlungsforderung erfolgt dann, wie im konventionellen Geschäftsablauf üblich, mit einer Rechnung. Auch diese Dienstleistung ist kundenspezifisch, d.h. der Kunde muß authentifiziert werden können, um eine Verbindlichkeit bei der Bestellung zu erreichen. Diese Authentifizierung ist weniger sicherheitskritisch, da neben der Authentifizierung beim Netzzugriff automatisch auch eine Plausibilitätsprüfung bei der Auftragsbearbeitung an sich erfolgt (in Analogie zu Bestellungen via Fax, Telefon o.ä.[12]). Um Verbindlichkeit zu erreichen, müssen die eigentlichen Bestellangaben vom Kunden signiert werden.

4.1.2.5 Integration in eigene Produkte

Mit der zunehmenden räumlichen Verteilung von Unternehmen wird der entfernte Zugriff auf Produktionsdaten immer interessanter, ja geradezu notwendig. Es bietet sich an, ebenfalls über WWW auf Informationen innerhalb der Produktionsplanungssysteme zugreifen zu können. Die Vorteile sind:

- Durch den Einsatz nicht proprietärer Protokolle und Technologien (Netzanbindung via TCP/IP, HTTP, Darstellung durch HTML, Einsatz von Java) kann der Datenfluß über Internet oder dedizierte Leitungen erfolgen, je nachdem welche Sicherheitsanforderungen gestellt werden, ohne daß etwas geändert werden müßte.
- Auf Client−Seite muß keine spezielle Software entwickelt, installiert und gewartet werden.
- Der Einsatz in heterogenen Umgebungen − im Sinne verschiedener Client−Betriebssysteme − ist einfacher.

Man kann sofort einsehen, daß solche Produktionsdaten z.T. eine sehr hohe Sensitivität haben, also ihre Integrität und Vertraulichkeit gewährleistet sein muß. Auch kann die kontrollierte Software−Verteilung mittels digitaler Signaturen gesichert werden.

[12] Auch im normalen geschäftlichen Ablauf verläßt sich der Lieferant auf Wissen und Erfahrungen, welche er mit einem bestimmten Kunden gemacht hat. Also besteht schon in einer normalen, schriftlichen Bestellung keine absolute Sicherheit für den Lieferanten, ob 1. die Bestellung gefälscht wurde oder 2. der Kunde auch wirklich bezahlen wird. Allerdings ist im Streitfalle die juristische Anerkennung z.B. einer schriftlichen Bestellung (mit Unterschrift) sicher, während es noch wenig juristische Erfahrungen mit der Anerkennung von z.B. Protokolldateien von Servern oder elektronischen Signaturen gibt.

4.2 Voraussetzungen

4.2.1 Bereits vorhandene Software

In diesem Abschnitt folgt eine kurze Bestandsaufnahme der bereits vor der Arbeit verwendeten Software bei der Propack Data GmbH. Diese Bestandsaufnahme motivierte die Auswahl der untersuchten Pakete in Kapitel 5.

4.2.1.1 Server−Software

Die Bereitstellung und Nutzung von Internet−Diensten wurde auch vor der Arbeit bereits über Linux−basierte Server abgewickelt. Dabei kam so weit wie möglich frei und als Quell−Code verfügbare Software zum Einsatz. Zusätzlich wurde vor der Arbeit auch *Lotus Notes* unter *Windows NT* in Betrieb genommen. Es besteht der Wunsch, zukünftig über Lotus Notes als integriertes Datenbank− und Messaging−System den organisatorischen Arbeitsablauf (Workflow) bei der Propack Data GmbH abzuwickeln.

4.2.1.1.1 Linux

Als Betriebssystem für sämtliche Server mit Internet−Diensten wurde das Unix−Derivat *Linux* eingesetzt, da handelsübliche Distributionen bereits alle Pakete mitbringen, die zur Nutzung sämtlicher Internet−Dienste notwendig sind. Inzwischen erfährt Linux auch immer mehr Unterstützung durch große Software−Hersteller (z.B. IBM/Lotus, Oracle, Netscape etc.), so daß man im nachhinein die Entscheidung für Linux als noch positiver bewerten muß, als es bei der Einführung abzusehen war.
Zudem liegt Linux (der Kernel selbst) bzw. ganze Distributionen als Quell−Code vor, und es findet über sicherheitsrelevante Aspekte eine sehr offene Diskussion statt. Insbesondere besteht also derzeit kein Grund, die Entscheidung für Linux als Server−Betriebssystem zu revidieren.

4.2.1.1.2 E−Mail

Als Software−Paket für den internen Mail−Server kam *sendmail* und zum lokalen Ausliefern *procmail* zum Einsatz. Die Mail wird nur intern erreichbar über POP3 und IMAP von den Benutzern abgeholt. Da sendmail sicherheitstechnisch nicht unkritisch ist, ist der E−Mail−Dienst nach außen über ein Mail−Relay mit vergleichsweise primitiver Relay−Software getrennt abgesichert.
Das Abholen von Mail via POP3 und IMAP ist als sicherheitskritisch einzustufen, da bei ungesicherter Übertragung Kennwörter im Klartext über LAN gehen. Vor allem war dies ein Ansatzpunkt für den Einsatz kryptographischer Sicherung (siehe Abschnitt 7.3.1).

4.2.1.1.3 WWW

Als WWW−Server wurde schon immer das Paket *Apache* eingesetzt, da dieser Server kontinuierlich weiterentwickelt wird, die Dokumentation gut ist und Sicherheits−Audits von den Entwicklern durchgeführt werden, die stets zu einer schnellen Benachrichtigung über Sicherheitsprobleme führen. Durch eine definierte Modulschnittstelle ist Apache erweiterbar, was immer mehr Drittherstellern ermöglicht, die Nutzung anderer Software−Pakete (z.B. Datenbanken, Skriptsprachen etc.) in Apache zu integrieren. Apache hat derzeit den höchsten Marktanteil bei WWW−Servern, auch hier gibt es keinen Grund, von Apache als WWW−Server abzurücken.
Da das Anbieten von Mehrwert−Diensten über WWW immer mehr in den Blickpunkt des Interesses rückt, ist vor allem hier ein Ansatz für den Einsatz kryptographischer Techniken gegeben (siehe Abschnitte 5.2.1, 7.3.2 und 7.4).

4.2.1.1.4 Lotus Notes unter Windows NT Server

Die Kombination aus Lotus Notes und Windows NT stellt ein besonderes Sicherheitsproblem dar. Beide Systeme sind komplex und haben ihre eigenen Authentifizierungs− und Autorisierungsmechanismen. Zudem findet man über die Sicherheitsaspekte und möglicherweise existierende Sicherheitsprobleme vergleichsweise wenig Informationen beim Hersteller. Zudem liegen natürlich beide System nicht als Quell−Code vor, absichtlich eingebaute Hintertüren für den Zugriff, vorbei an sämtlichen Autorisierungsmechanismen, sind nicht auszuschließen. Dies führt dazu, daß solche Systeme bei der Propack Data GmbH nicht mit direktem Internet−Zugriff betrieben werden.
Trotzdem ist gerade der Wunsch, Lotus Notes einzusetzen, besonders groß, da Notes eine universelle, gut skalierbare Plattform für Workflow−Anwendungen darstellt (z.B. mit Replikation auf mehrere Server zur Datensicherung und Leistungssteigerung und für den Off−Line−Betrieb von Notes−Clients). Aufgrund der unzulänglichen kryptographischen Eigenschaften von Notes (Abschnitt 5.2.2) mußten andere Wege gefunden werden, die Notes−Nutzung auch über Internet zu ermöglichen (Beispiel siehe Abschnitt 7.3.2).

4.2.1.2 Client−Software

Alle Arbeitsstationen arbeiteten unter dem Betriebssystem *Windows NT Workstation*, da aufgrund der momentanen Marktsituation auch die eigenen Produkte der Propack Data GmbH unter Windows NT entwickelt und angeboten werden. Als Client−Software wurde der *Netscape Communicator* (siehe auch Abschnitt 5.1.1) eingesetzt, sowohl für die Nutzung der E−Mail− und News−Dienste als auch als WWW−Client. Diese Client−Software wurde ausgewählt, da zum einen alle relevanten, im Internet gängigen Standards unterstützt werden und zum anderen die Benutzer mit der integrierten Benutzeroberfläche für die Nachrichten− und WWW−Dienste gut zurechtkommen. Zudem gilt für den Netscape Communicator eine recht preisgünstige Lizenzpolitik.

4.3 Geplante Sicherungsmaßnahmen

Aufgrund der bestehenden Wünsche und vielfältigen Kommunikationsbeziehungen werden in dieser Arbeit folgende Sicherungsmaßnahmen für externe Kommunikation bestimmt:

- Sicherstellung der Vertraulichkeit und Authentizität aller internen und externen E−Mail−Nachrichten mittels Verschlüsselung und digitalen Signaturen. Es wird dabei der S/MIME−Standard (Abschnitt 3.5.4) für verschlüsselte und signierte Nachrichten verwendet, da dieser auf ohnehin für die Sicherung von Server−Zugriffen benötigten X.509v3−Client−Zertifikaten beruht und neben Netscape auch andere große Software−Herstellern diesen offenen Standard unterstützen.
- Beim WWW−Dienst soll die Sicherung der Vertraulichkeit mittels SSL−Verbindungen und X.509v3−Server−Zertifikaten und eine sichere Authentifizierung von Benutzern basierend auf X.509v3−Client−Zertifikaten durchgeführt werden.
- Alle weiteren Internet−Dienste wie z.B. POP3, IMAP und NNTP sind mittels SSL−Verbindungen und X.509v3−Server−Zertifikaten gegen Abhören zu schützen.
- Die Vertraulichkeit und Integrität von Datenkommunikation über Leitungen öffentlicher Netze soll mittels Verschlüsselung auf Routing−Ebene durch IPSec (Abschnitt 3.5.2) gesichert werden, falls auf Kommunikationsstrecken Ende−zu−Ende−Sicherung zwischen Client und Server nicht für alle Anwendungen möglich ist.

- Es wird festgelegt, daß nirgends Kryptosysteme zum Einsatz kommen, deren Schlüsselängen durch die Gesetzgebung künstlich herabgesetzt wurde. Entsprechende Client−Software wird als Mindestvoraussetzung zur Nutzung der angebotenen Dienste definiert. Die jeweilige Server−Software ist derart zu konfigurieren, daß Verbindungsaufbauten von Client−Software mit zu niedriger Schlüssellänge als unzulässig abgewiesen wird.
- Zugriffe über Internet und dedizierte Leitungen sollten in der Zukunft möglichst gleich behandelt werden, da auch dedizierte Leitungen über öffentliche Netze letztlich unsicher sind. Desweiteren werden die Konfigurationen durch Vereinheitlichung einfacher und damit ebenfalls sicherer.

Idealerweise sollten dabei alle beteiligten Systeme keine Trennung zwischen internen und externen Systemen bzw. Benutzern mehr machen müssen. Es sollten überall einheitliche Mechanismen, basierend auf X.509v3−Zertifikaten, zur Authentifizierung der Benutzer verwendet werden.

4.4 Zusammenfassung

Die inzwischen hohe Mobilität mancher Anwender und die vielfältigen, internationalen Kooperationen stellen hohe Anforderungen an die Flexibilität der Authentifizierungs− und Autorisierungsmechanismen beim verteilten Zugriff auf Informationen bei der Propack Data GmbH. Um die Komplexität zu reduzieren, sollte man anstreben, interne und externe Zugriffe und alle Klassen von Benutzern möglichst gleich zu behandeln, also für alle Zugriffe einheitliche Authentifizierungs− und Autorisierungsmechanismen zu verwenden. Aufgrund der heterogenen Zusammensetzung der Systeme und fehlender Standardisierung ist dies oft sehr schwierig zu realisieren.
Die oben dargestellten Anforderungen und gewünschten Anwendungen ließen die Sicherung der E−Mail− und WWW−Dienste als vordringlich erscheinen, zumal diese Dienste bedingt durch offene Standards auch in einer heterogenen Systemlandschaft einheitlich behandelt werden können. Bei beiden Diensten sollten dabei die Vertraulichkeit, Authentizität und Integrität durch den Einsatz von Verschlüsselung und digitalen Signaturen gewährleistet werden. Allerdings ist beim Einsatz kryptographischer Techniken über Ländergrenzen hinweg oft eine länderspezifische Gesetzgebung für den Einsatz von Kryptographie zu beachten.
Die schon vor der Arbeit gemachte Festlegung auf die Verwendung von offenen Internet−Standards und die darauf basierende Software−Auswahl legten keine radikalen Änderungen an der Software−Auswahl nahe. Vielmehr wurden, auf der Situation aufbauend, die schon verwendeten Software−Pakete auf ihre Tauglichkeit zum Einsatz kryptographischer Techniken untersucht (Kapitel 5).

In den nun folgenden Kapiteln wird das Ziel verfolgt, eine Architektur und Infrastruktur zu entwerfen und aufzubauen, welche als Basis zur kryptographischen Sicherung der externen Kommunikationsbeziehungen benutzt werden kann (Kapitel 6). Desweiteren werden auch den o.g. Wünschen entsprechende Beispielanwendungen (Kapitel 7) erstellt, um Möglichkeiten der Nutzung dieser Basisinfrastruktur aufzuzeigen.

5 Verfügbare Software

In diesem Kapitel wird verfügbare Software und ihre Eigenschaften kurz vorgestellt. Zuerst werden verschiedene Programme für die Client−Seite (z.B. WWW−Browser, E−Mail−Client etc.) beschrieben. Danach folgt eine Darstellung verschiedener Software für WWW−Server und kryptographische Werkzeuge zum Aufbau einer Zertifizierungsstelle. Es wird jeweils eine kurze Bewertung für den Einsatz der jeweiligen Software gegeben, wobei ein besonderes Augenmerk auf Software gelegt wird, die bereits bei der Propack Data GmbH in Gebrauch ist.

5.1 Client−Software

Da ganz normale Anwender die kryptographischen Techniken benutzen sollen, kommt der Auswahl der Client−Software besondere Bedeutung zu. Die jeweilige Client−Software muß dem Benutzer möglichst transparent Routineaufgaben abnehmen, aber trotzdem den Benutzer auf sicherheitskritische Entscheidungen hinreichend ausführlich aufmerksam machen. Der Erfolg beim Einsatz kryptographischer Techniken hängt zu einem großen Teil auch von der Akzeptanz seitens der Benutzer ab. Zudem ist die Entscheidung für eine bestimmte Software, selbst beim konsequenten Einsatz nicht proprietärer, weit verbreiteter Standards, meist sehr langfristig, da der Installationsaufwand von Client−Software auf allen Arbeitsplatzrechnern sehr hoch ist und die Benutzer sich nur ungern von Vertrautem trennen. In den folgenden Abschnitten werden nur wirklich getestete Programme beschrieben, es wird also keine Marktübersicht gegeben, sondern es werden Programme kurz vorgestellt, welche bei der Propack Data GmbH zum Einsatz kommen.

5.1.1 Netscape Communicator

Der *Netscape Communicator* in der frei verfügbaren Basisversion (Version 4.0+ und 4.5) besteht im wesentlichen aus dem Produkt *Netscape Navigator*, einem WWW−Browser, und dem *Netscape Messenger*, einem E−Mail−Client und News−Reader.
Das Produkt bietet den Zugriff auf folgende Internet−Dienste:

- WWW (über HTTP)
- Mail (Senden der E−Mail über SMTP, Empfangen von E−Mail über POP3 und IMAP)
- Verzeichnisdienst (über LDAP)

Jeder Zugriff auf Server−Dienste kann dabei transparent über SSL (Abschnitt 3.5.3) kryptographisch gesichert werden, wobei auch die Möglichkeit besteht, beim SSL−Verbindungsaufbau Client−Zertifikate zur Authentifizierung gegenüber einem Server zu verwenden. Der *Netscape Messenger* enthält ferner die Möglichkeit zur Ende−zu−Ende Nachrichtenverschlüsselung und Signatur gemäß S/MIME−Standard (Abschnitt 3.5.4). Eine schematische Übersicht der an den kryptographischen Funktionen beteiligten Module gibt Abbildung 16.

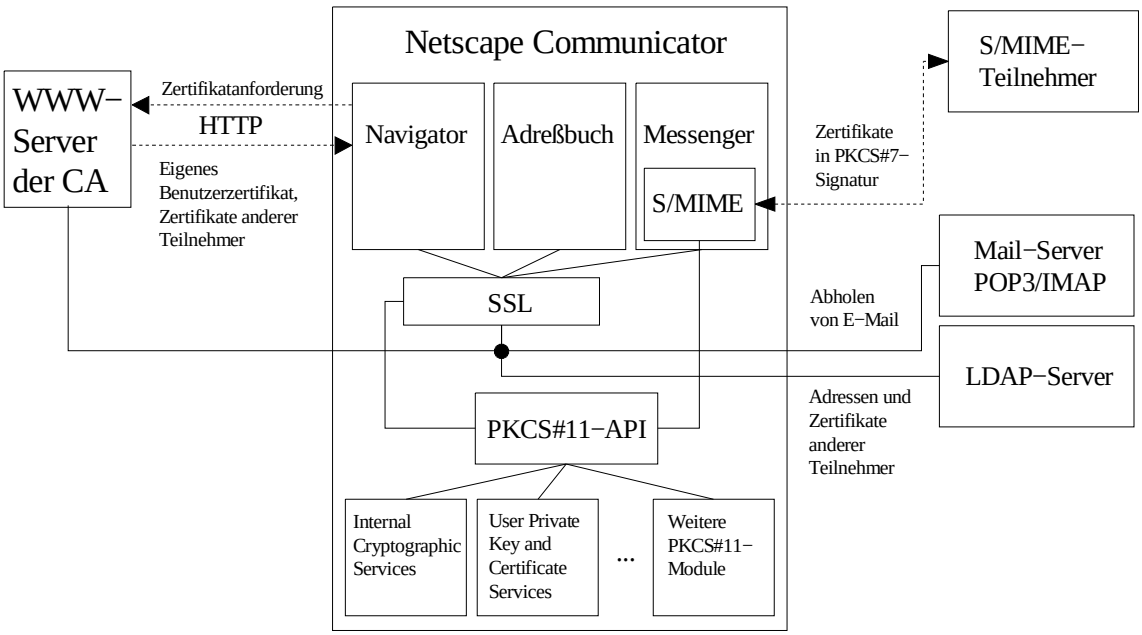


Abbildung 16: Übersicht der kryptographischen Funktionen des Netscape Communicator

Eine zentrale Rolle spielt die SSL−Implementierung, welche von allen Anwendungsteilen zur Sicherung der Protokolle bestimmter Internet−Dienste benutzt wird. Ebenso hervorzuheben ist die konsequente Verwaltung der Kryptofunktionen in PKCS#11−Modulen (Abschnitt 5.1.1.7).
Der Netscape Communicator unterliegt als U.S.−amerikanisches Produkt den U.S.−Exportbestimmungen, d.h. die symmetrischen Algorithmen sind in ihrer Schlüssellänge auf 40 Bit und die Schlüssellänge der asymmetrischen Algorithmen auf 512 Bit beschränkt. Abhilfe schafft hier nur der für kommerziellen Einsatz kostenpflichtige Patch *Fortify* oder die Installation einer illegal exportierten U.S.−Version.

Im folgenden werden die verschiedenen kryptographischen Funktionen des Netscape Communicator beschrieben. Die Bezeichnungen der Benutzeroberfläche sind der U.S.−amerikanischen Version entnommen, die Abschnitte gliedern sich nach den Rubriken des *"Security Info"*−Fenster−Dialogs, zu dem man sehr leicht über das *"Communicator"*−Menü oder einen gut sichtbaren Druckknopf gelangen kann.

5.1.1.1 Security Info

Unter "Security Info" werden immer die beim Einsatz der kryptographischen Techniken relevanten Daten bezogen auf den aktuellen Kontext angezeigt. Die Meldungen sind recht ausführlich, spiegeln aber nicht immer den exakten Sachverhalt wider, was aber auch an den oft mehrdeutigen Situationen liegt. Zudem sind die Benutzer oft mit der Fülle der Informationen überfordert und lesen die meist klar verständlichen Meldungen nicht ausreichend sorgfältig durch.

5.1.1.1.1 Sicherheitsinformationen zu Server−Zugriffen

Zeigt man die Sicherheitsinformationen an, wenn man gerade eine über SSL empfangene WWW−Seite anschaut oder einen Mail− bzw. News−Server ausgewählt hat, so hat man die Möglichkeit, das Server−Zertifikat anzeigen zu lassen. Erfolgt der Zugriff nicht über SSL gesichert, so wird ein entsprechender Hinweis gegeben.

Ferner kann man bei WWW−Servern auch die Seiteninformation ("Page Info") anzeigen lassen. In der Seiteninformation erscheint dann ein Fensterdialog, welcher auch einen Druckknopf zur Online−Verifikation des Server−Zertifikats bereitstellt, falls dieses das Attribut *nsRevocationUrl* enthält (vergleiche Abschnitte 3.3.8.1.1 und 6.7.3.5). Außerdem ist ein Knopf zur Anzeige der dem Server−Zertifikat zugrunde liegenden Zertifizierungsrichtlinien vorhanden, falls das Attribut *nsCaPolicyUrl* gesetzt ist (Abschnitt 3.3.8.1.1).

5.1.1.1.2 Sicherheitsinformationen zu Nachrichten

Bei der Anzeige von Nachrichten werden immer die folgenden Aussagen gemacht:

- "Encrypted Message"
  Es wird angegeben, ob eine Nachricht verschlüsselt war. Im Falle einer verschlüsselten Nachricht werden der verwendete kryptographische Algorithmus und die Schlüsselstärke genannt. Kann die Nachricht nicht entschlüsselt werden, weil der zugehörige private Schlüssel oder das Empfängerzertifikat fehlt, so wird auf dieses Problem hingewiesen.
- "Signed Message"
  Es wird angegeben, ob eine Nachricht vom Sender signiert war. Im Falle einer signierten Nachricht kann das Zertifikat des Senders angezeigt werden. Konnte das Senderzertifikat nicht überprüft werden, so wird dies ebenfalls beschrieben.

Beide Informationen werden in prägnanter Form auch im Netscape Messenger in einem Symbol zusammen mit der Nachricht angezeigt.

5.1.1.2 Passwords

In dieser Rubrik kann man die Kennwörter für den Zugriff auf die kryptographischen Module, insbesondere den Zugriff auf die Zertifikatdatenbank, setzen bzw. ändern. Es kann auch bestimmt werden, wie oft nach dem Kennwort gefragt wird.

5.1.1.3 Navigator

In dieser Rubrik werden verschiedene Optionen rund um den SSL−gesicherten Zugriff eingestellt.

5.1.1.3.1 Warnoptionen für den WWW−Zugriff

Speziell für den WWW−Zugriff können verschiedene Warnoptionen eingestellt werden ("Show a warning before"):

- "Entering an encrypted site"
  Ein Warnung wird angezeigt, falls erstmals auf einen SSL−Server zugegriffen wird. Dies ist nicht unbedingt erforderlich. Man sollte diese Option eher ausschalten, damit der Benutzer nicht durch Pop−Up−Fenster überbeansprucht wird.
- "Leaving an encrypted site"
  Es wird eine Warnung angezeigt, falls von einem SSL−Server kommend wieder auf eine unverschlüsselte Seite zugegriffen wird. Dies ist sinnvoll, um dem Benutzer das Verlassen gesicherter Bereiche bewußt zu machen.

- "Viewing a page with an encrypted/unencrypted mix"
  Speziell beim Einsatz von Frames in WWW−Seiten oder Verweisen auf fremde Grafiken kann die Situation für den Benutzer schnell unübersichtlich werden. Es kann leicht vorkommen, daß Teile der WWW−Seite nicht kryptographisch authentifiziert sind. Eine diesbezügliche Warnung erscheint daher sehr sinnvoll.
- "Sending unencrypted information to a Site"
  Füllt ein Benutzer ein HTML−Formular aus, so wird er darauf aufmerksam gemacht, daß die von ihm angegebenen Daten im Klartext übertragen werden. Diese Option ist auf jeden Fall sinnvoll.

Trotz dieser ganzen Warnungen bleibt es dem Benutzer überlassen, zu entscheiden, was er tun möchte und was nicht. Das ist aus zwei Gründen problematisch: Zum einen hat ein normaler WWW−Benutzer meist keine tieferen Kenntnisse der verwendeten Mechanismen. Die Warnmeldungen sind für ihn also das eigentlich Kryptische und führen u.U. beim Benutzer schnell zu einer Saturierung der Wahrnehmung. Zum anderen ist auch auf Seite des Server−Betreibers eine gewisse Sensibilität bei der Erstellung der gesicherten WWW−Inhalte gefragt. Allzugern wird ein Benutzer von einem Web−Designer gezwungen, sicherheitskritische Strukturen im WWW−Design zu akzeptieren. Im Zweifelsfall wird der Benutzer eher eine Warnung ignorieren oder abschalten, denn auf den Zugriff der von ihm gewünschten Inhalte zu verzichten.

5.1.1.3.2 Authentifizierung mittels Benutzerzertifikaten

Der Benutzer kann einstellen, mit welchem Benutzerzertifikat er sich gegenüber einem WWW−Server authentifizieren möchte, sofern der Server beim SSL−Verbindungsaufbau vom Benutzer ein Zertifikat anfordert. Folgende Optionen stehen zur Auswahl:

- Es kann ein Benutzerzertifikat für alle Zugriffe ausgewählt werden. Zur Auswahl stehen nur diejenigen Zertifikate, deren Attribut *nsCertType,* falls vorhanden, entsprechend gesetzt ist (vergleiche Tabelle 2 in Abschnitt 3.3.8.1.2).
- "Select Automatically"
  Es wird automatisch dasjenige Benutzerzertifikat verwendet, welches die gleiche Zertifizierungsstelle hat wie das Server−Zertifikat.
- "Always Ask"
  Der Benutzer wird immer wieder mit einem Pop−Up−Fenster nach dem zu verwendenden Zertifikat gefragt.

5.1.1.3.3 SSL−Optionen

Der Benutzer kann zusätzlich entscheiden, welche SSL−Versionen er akzeptiert und welche Kryptosysteme dabei zum Einsatz kommen können. Das ist nützlich, um die Verwendung von inakzeptablen Kryptosystemen mit zu kurzen Schlüsseln ganz abzuschalten.

5.1.1.4 Messenger

In der Rubrik "Messenger" kann der Benutzer festlegen, ob Mail− und News−Nachrichten immer verschlüsselt oder signiert werden sollen (News−Artikel können nur signiert werden) und welches Zertifikat zum Signieren verwendet wird.
Zudem kann mit dem Druckknopf "Send Certificate To Directory" das Zertifikat an einen LDAP−Server geschickt werden, wo das Zertifikat als Attribut *usersmimecertificate;binary* abgelegt wird (vergleiche auch 6.7.3.2).

Schließlich kann auch mit dem Druckknopf "Select S/MIME Ciphers" für das S/MIME−Protokoll bestimmt werden, welche Kryptosysteme zum Einsatz kommen können, um die Verwendung von inakzeptabel schwachen Kryptosystemen zu vermeiden. Von dieser Möglichkeit sollte unbedingt Gebrauch gemacht werden: Kryptosysteme mit Schlüssellängen kleiner als 128 Bit sind abzuschalten, um den Anwender vor versteckten Sicherheitsrisiken zu bewahren.

5.1.1.5 Java/JavaScript

Diese Rubrik betrifft das Signieren von dynamisch ladbaren und direkt im Navigator ausführbaren Programmen, welche in Java oder JavaScript geschrieben sind. Um sicherzustellen, daß nur vertrauenswürdiger Java− oder JavaScript−Code zur Ausführung kommt, kann der Urheber diesen signieren [Nets98f]. Die Netscape−Software überprüft dann vor der Ausführung eines Java−Applets oder eines JavaScript−Programms die Signatur und ob der Unterzeichner des Codes als vertrauenswürdig angesehen wird. In diesem Dialog kann der Benutzer einstellen, welchen Unterzeichnern von Java− oder JavaScript−Code er vertraut.

5.1.1.6 Certificates

In dieser Rubrik sind alle der Netscape−Software bekannten Zertifikate nach Kategorien geordnet aufgelistet. Jedes Zertifikat kann angezeigt, off−line überprüft oder gelöscht werden. Zusätzlich kann man bei manchen Zertifikaten noch Optionen einstellen. Die Zertifikatdatenbank selbst wird über ein PKCS#11−Modul verwaltet (Abschnitt 5.1.1.7).

5.1.1.6.1 Yours

Die eigenen Zertifikate des Benutzers werden unter "Yours" angezeigt. Man kann an dieser Stelle auch eigene Zertifikate zusammen mit dem privaten Schlüssel als PKCS#12−Datei exportieren oder eine PKCS#12−Datei importieren. Es empfiehlt sich auf jeden, Fall Sicherheitskopien aller eigenen Zertifikate als PKCS#12−Dateien anzufertigen und diese sorgfältig aufzuheben.
Leider bietet die Netscape−Software für den Benutzer keinerlei Möglichkeit, ohne Interaktion mit einem WWW−Server ein selbstsigniertes Zertifikat oder eine Zertifikatanforderung (certificate request) zu erstellen, was einen eklatanten Nachteil bei der Identitätsprüfung darstellt, da der Benutzer z.B. nicht mit einer Diskette mit Zertifikanforderung und Personalausweis bei der CA erscheinen kann. Vielmehr interpretiert der Netscape Navigator das proprietäre HTML−Tag <KEYGEN> in einem HTML−Formular als ein spezielles Eingabefeld und zeigt eine Select−Box zum Auswählen der Schlüssellänge für den asymmetrischen Algorithmus an [Nets96][Nets97f]. Schickt der Benutzer das HTML−Formular ab, so erzeugt der Netscape Navigator nach der Anzeige eines Dialogs ein Schlüsselpaar und übermittelt mit dem HTML−Formular den öffentlichen Schlüssel als Formularparameter, welcher von einem CGI−BIN−Programm bearbeitet werden muß. In der Regel schreibt der WWW−Server dann einen sog. SPKAC−Request, eine bestimmte Form einer Zertifikatanforderung, auf seine lokale Platte. Dem CA−Betreuer liegt also nur diese SPKAC−Datei vor, welche aus über das Netzwerk übertragenen Daten besteht, sonst nichts. Die Möglichkeit einer direkten Identitätsprüfung entfällt.
Der Benutzer bekommt leider auch von der Netscape−Software keine Möglichkeit, die Zertifikatanforderung lokal zu speichern oder zumindest die ausstehenden Zertifikatanforderungen anzuzeigen, wodurch die einfache Möglichkeit einer out−of−band−Überprüfung entfällt. Das Speichern der SPKAC−Zertifikatanforderung beim Benutzer muß also durch das bearbeitende CGI−BIN−Programm ermöglicht werden, sei es durch Anzeige des SPKAC−Requests auf dem WWW−Server oder durch Versand einer entsprechenden Mail. Allerdings kann der Benutzer nicht die Zugehörigkeit des angezeigten SPKAC−Requests zu seinem eigenen privaten Schlüssel testen. Wenn die

Zertifizierungsstelle das Zertifikat ausgestellt hat, so sendet sie dem Benutzer eine URL−Adresse, wo er sein Zertifikat laden kann. Benutzerzertifikate werden mit dem MIME−Typ *application/x−x509−user−cert* in den Netscape Navigator eingebracht, welcher beim Empfang überprüft, ob der öffentliche Schlüssel im Zertifikat zu einem privaten Schlüssel in der Zertifikatdatenbank gehört, und danach dem Benutzer einen entsprechenden Dialog zur Anzeige und zum Akzeptieren des eigenen Zertifikats anzeigt.

5.1.1.6.2 People

Unter "People" sind die Zertifikate anderer Benutzer aufgelistet. Möchte man eine verschlüsselte E−Mail−Nachricht schicken, so muß hier ein Zertifikat mit der passenden E−Mail−Adresse vorhanden sein. Bei selbstsignierten Zertifikaten, die nicht von einer Zertifizierungsstelle herausgegeben wurden, muß außerdem noch der Verwendungszweck des Zertifikats gesetzt werden (Druckknopf "View/Edit"), ähnlich wie bei einer Zertifizierungsstelle (Abschnitt 5.1.1.6.4).
Benutzerzertifikate können auf mehreren Wegen in die Zertifikatdatenbank gelangen: Zum einen kann ein Benutzerzertifikat via HTTP als MIME−Typ *application/x−x509−email−cert* von einem WWW−Server geladen werden (Abschnitt 6.7.3.1).
Es kann aber über den Druckknopf "Search Directory" auch auf einem LDAP−Server nach dem Zertifikat gesucht und dieses geladen werden (Abschnitt 6.7.3.2). Der Netscape Communicator sucht zunächst nach LDAP−Einträgen mit der gewünschten E−Mail−Adresse im Attribut *mail* und verwendet dann vorzugsweise das Zertifikat aus dem Attribut *usersmimecertificate;binary,* falls dieses vorhanden ist. Falls nicht, so wird das Zertifikat aus dem Attribut *usercertificate;binary* verwendet.
Desweiteren wird ein Benutzerzertifikat auch über eine signierte S/MIME−Mail ohne Rückfrage in die Zertifikatdatenbank übernommen, falls es dort noch nicht vorhanden ist.

5.1.1.6.3 Web Sites

Es kann vorkommen, daß ein Benutzer eine SSL−gesicherte Verbindung zu einem Server aufbaut, welcher ein Server−Zertifikat präsentiert, zu dem das CA−Zertifikat fehlt. Das Server−Zertifikat kann also nicht überprüft werden. In diesem Fall präsentiert der Netscape Navigator einen mehrstufigen Dialog, welcher das vom Server vorgelegte Zertifikat anzeigt und den Benutzer vor einem möglichen Mißbrauch warnt, aber trotzdem einen SSL−Verbindungsaufbau zulässt, falls der Benutzer dies explizit wünscht. Der Benutzer kann mehrere Möglichkeiten in Anspruch nehmen:

- "Accept this certificate for this session"
  Das Server−Zertifikat wird bis zum Beenden des Netscape Communicator akzeptiert.
- "Do not accept this certificate and do not connect"
  Das Server−Zertifikat wird nicht akzeptiert, der SSL−Verbindungsaufbau wird abgebrochen.
- "Accept this certificate forever (until it expires)"
  Das Server−Zertifikat wird bis zu seinem Verfallsdatum akzeptiert und in die Zertifikatdatenbank aufgenommen.

Der Benutzer kann zudem festlegen, ob er gewarnt werden möchte, falls er genau diesem Server Formulardaten schickt.

5.1.1.6.4 Signers

In dieser Zertifikat−Rubrik sind die Zertifikate der Zertifizierungsstellen untergebracht. Nach der Neuinstallation der Netscape−Software sind hier nur die fest in der Netscape−Software eingebauten CA−Zertifikate zu finden. Weitere CA−Zertifikate können über einen WWW−Server−Zugriff (vergleiche Abschnitt 6.7.1) oder eine signierte S/MIME−Nachricht hinzugefügt werden.

Im ersten Fall wird dem Benutzer ein mehrstufiger Dialog präsentiert, der ihn auf die Bedeutung eines neuen CA−Zertifikats und die dabei dem Benutzer obliegende, notwendige Sorgfaltspflicht bei der Beurteilung der Vertrauenswürdigkeit hinweist. Der Benutzer kann schließlich bestimmen, für welchen Verwendungszweck (Ausstellen von Server−, E−Mail oder Programm−Zertifikaten) dem CA−Zertifikat vertraut werden soll. Enthält das CA−Zertifikat das X.509v3−Attribut *nsCertType* (Abschnitt 3.3.8.1.1, Tabelle 2), so werden dem Benutzer nur die dort angegebenen Möglichkeiten zur Auswahl gegeben.
Ein neues CA−Zertifikat kann auch mittels einer signierten S/MIME−Nachricht in die Zertifikatdatenbank gelangen, da in der PKCS#7−Signatur die komplette Zertifikatkette untergebracht wird. Dies geschieht für den Benutzer unmerklich beim Öffnen der E−Mail−Nachricht, und es wird leider kein Dialog zum Akzeptieren einer neuen Zertifizierungsstelle präsentiert. Dies kann dazu führen, daß ein CA−Zertifikat zwar vorhanden ist, aber die Vertrauensoptionen für den Verwendungszweck noch nicht gesetzt sind. Wird dann eine mit einem Benutzerzertifikat dieser CA erzeugte Signatur präsentiert, so bezeichnet Netscape diese als ungültig, obwohl alle zur Überprüfung benötigten Zertifikate vorhanden sind. Ein erneutes Laden des CA−Zertifikates von einem WWW−Server wird ebenfalls mit Hinweis abgelehnt, daß dieses CA−Zertifikat schon vorhanden ist. Der Benutzer muß in diesem Fall das Zertifikat in der Liste der Zertifizierungsstellen suchen und die Optionen von Hand setzen (Button *"Edit"*). Eine weitere Anmerkung muß zur Version 4.5 des Netscape Communicators gemacht werden: Dort haben die Entwickler stillschweigend die Zertifikatkette in der PKCS#7−Signatur gekürzt. Die Top−Level−CA ist "aus Sicherheitsgründen" nicht mehr in der CA−Kette vorhanden, wie das zukünftige Verhalten sein wird, ist unbekannt[13].
Unter der Rubrik "Signers" werden auch Zertifikatwiderruflisten verwaltet. CRL−Zertifikate werden via HTTP oder S/MIME−Mail mit dem MIME−Typ *application/x−pkcs7−crl* in Netscape Communicator eingebracht. Hat die Netscape−Software mindestens eine CRL erhalten, so zeigt sie unter "Signers" den Druckknopf "View/Edit CRLs", mit welchem man zu einem weiteren Dialogfenster gelangt, in dem man CRLs anzeigen, neu laden, löschen oder die zugehörige URL−Adresse ändern kann. Hier zeigt sich eine schwerwiegende Unzulänglichkeit in der Benutzerführung: Selbst wenn ein Zertifikat das erweiterte Attribut *nsCaRevocationUrl* gesetzt hat, hat der Benutzer keine Möglichkeit, von sich aus initiativ die CRL zu laden. Erst wenn er mindestens einmal tatsächlich eine CRL geladen hat, erscheint der Druckknopf mit dem Zugang zum CRL−Dialog. Löscht der Benutzer jetzt aus Versehen alle CRLs, so ist der Dialog wieder unzugänglich. Es bleibt dem Benutzer nur die Möglichkeit, vom WWW−Server der CA wieder eine CRL zu laden, welche dort an prominenter Stelle leicht zugänglich sein sollte.

5.1.1.7 Cryptographic Modules

Die Netscape−Software bietet die Möglichkeit, kryptographische Module (*Cryptographic Modules*) nach dem PKCS#11−Standard (oder auch Cryptoki−API, siehe Abschnitt 3.5.1) einzubinden. Dies ermöglicht es, kryptographische Module von Fremdherstellern einzubinden, z.B. für den benutzerfreundlichen, transparenten Einsatz von externer Kryptohardware (Chipkarten etc.), welche die sichere Handhabung von privaten Schlüsseldaten verbessert. In dem Fensterdialog können genauere Informationen über die einzelnen Module angezeigt, neue Module eingebunden und vorhandene Module gelöscht werden.
Nach einer Standardinstallation ist hier lediglich ein in der Netscape−Software enthaltenes Modul unter dem Namen *Netscape Internal PKCS #11 Module* angemeldet, welches aus zwei sog. *Slots* besteht:

[13] Diese Informationen waren Gegenstand einer Diskussion in einer News−Gruppe auf news://secnews.netscape.com und lassen sich deshalb schlecht mit einer klassischen Literaturangabe belegen.

- *Communicator Internal Cryptographic Services*
  Dies ist die interne Netscape−Bibliothek mit den symmetrischen und asymmetrischen kryptographischen Algorithmen.
- *Communicator User Private Key and Certificate Services*
  Dies ist ein Modul zur Verwaltung und verschlüsselten Speicherung der Zertifikatdatenbank. Die Zertifikate werden von diesem Modul in der Datei *cert3.db* und die privaten Schlüsseldaten in einer Datei namens *key7.db* gehalten, welche sich im gleichen Verzeichnis wie die Netscape−Einstellungen befinden (unter OS/2 oder Windows 95/NT im Profile−Verzeichnis des Benutzers, unter Unix in ~/.netscape/) (nähere Angaben zum Format der Dateien macht [Hens98b]).

Diese Erweiterungsmöglichkeit mittels PKCS#11 macht Netscape sehr interessant für den unternehmensweiten Einsatz für verschiedene Internet−Dienste, da es eine konsequent sichere Handhabung personenbezogener, privater Schlüsseldaten auf Chipkarten ermöglicht. Der Benutzer muß sich nicht mehr um seine Arbeitsplatzrechnerkonfiguration sorgen, die Authentifizierung gegenüber den oft genutzten Internet−Diensten (WWW, Mail, Verzeichnisdienst) geschieht mittels Chipkarte, was dem Paradigma eines herkömmlichen Schlüssels sehr nahe kommt und damit dem Benutzer in der Handhabung vertrauter ist. Aufgrund der weiten Verbreitung der Netscape−Software sind in naher Zukunft viele kostengünstige PKCS#11−basierte Produkte zu erwarten.

5.1.2 Opera

Die Software *Opera* (derzeit Version 3.51) ist ebenfalls ein WWW−Browser mit Mail− und News−Funktionalität und wird bei der Propack Data GmbH als integrativer Bestandteil eigener Produkte eingesetzt, da die Software im Vergleich zum Netscape Communicator klein, schnell und leicht zentral zu administrieren ist. Opera ist verfügbar für alle Windows−Versionen, OS/2− und X11−Portierungen sind derzeit angekündigt und in der Entwicklung. Im Hinblick auf den Einsatz kryptographischer Techniken ist vor allem positiv zu erwähnen, daß Opera als europäisches Produkt derzeit keinen Export−Einschränkungen hinsichtlich der Stärke kryptographischer Routinen unterliegt.
Die Verschlüsselungsfunktionen in Opera sind bezüglich der Server−Zugriffe ähnlich umfassend wie die des Netscape Communicator. Verbindungen zu Servern können mittels SSL gesichert werden, und die Authentifizierung eines Benutzers gegenüber einem Server mittels Client−Zertifikaten ist möglich. Die Verschlüsselung bzw. das Signieren von E−Mail−Nachrichten und News−Artikeln mit S/MIME wird aber nicht unterstützt. Die Zertifikate werden ebenfalls in einer durch ein Kennwort gesicherten Datenbank gespeichert. Opera unterstützt ebenfalls das von Netscape spezifizierte <KEYGEN>−Tag, der Zertifizierungsablauf für Client−Zertifikate ist also exakt der selbe wie in Abschnitt 6.7.2 beschrieben.
Die MIME−Typen *application/x−x509−ca−cert* (für CA−Zertifikate), *application/x−x509−server−cert* (für Server−Zertifikate) und *application/x−x509−user−cert* (für Client−Zertifikate) werden unterstützt, um Zertifikate in die Opera−Zertifikatdatenbank einzubringen. Entsprechende Dialoge stellen Rückfragen an den Benutzer mit Anzeige der Zertifikate. Der MIME−Typ *application/x−x509−email−cert* (Client −Zertifikate anderer S/MIME−Teilnehmer) wird mangels S/MIME−Funktionalität nicht benötigt und demnach auch nicht erkannt. Import und Export von Client−Zertifikaten sind derzeit noch nicht vorgesehen, so daß man leider nicht Opera und Netscape Communicator oder andere Client−Software parallel mit den selben Client−Zertifikaten verwenden kann.
Alles in allem ist Opera als WWW−Client−Software eine gute Alternative zum Netscape Communicator, vor allem im Hinblick auf die Benutzung kryptographischer Sicherungsmechanismen, da die Schlüssellängen nicht künstlich durch Exportrestriktionen beschränkt sind. Vor allem die fehlende

S/MIME−Funktionalität und der fehlende Import und Export von Client−Zertifikaten lassen Opera aber nicht als vollwertigen Client−Ersatz zum Netscape Communicator erscheinen. Zudem sind auch keine weitergehenden Konzepte, wie beispielsweise die Integration PKCS#11−kompatibler Komponenten, erkennbar, und die Benutzerführung bei der Handhabung der kryptographischen Daten läßt noch zu wünschen übrig.

5.1.3 Lynx mit SSL

Lynx ist ein WWW−Browser für Textterminals und daher vor allem auch interessant für Behinderte, da alle Ausgaben rein textuell sind und sich der Browser ausschließlich mit der Tastatur bedienen läßt. Beispielsweise sind Blinde auf Lynx als WWW−Browser angewiesen.
Die Originalversion von Lynx unterstützt keinerlei kryptographische Techniken, es gibt aber einen Patch, welcher Lynx die Krypto−Routinen von SSLeay zugänglich macht. Damit kann Lynx auch auf mit SSL gesicherte WWW−Server zugreifen. Auch Lynx unterliegt damit keinerlei Exportbeschränkungen, der kommerzielle Einsatz in den USA ist aber aus patentrechtlichen Gründen verboten (näheres zur Patentlage von *SSLeay* in Abschnitt 5.3.1).
Lynx bietet leider keinerlei Möglichkeiten, sich gegenüber einem WWW−Server mithilfe von Benutzerzertifikaten zu authentifizieren. Auch das Einbringen von CA− und Server−Zertifikaten ist nicht steuerbar, dem Benutzer werden keinerlei Verschlüsselungsinformationen angezeigt. Alles in allem dient der Lynx/SSL−Patch nur dazu, überhaupt mit Lynx auf SSL−gesicherte WWW−Server zugreifen zu können. Für einen ernsthaften Einsatz zur kryptographisch gesicherten Nutzung von Internet−Diensten ist Lynx daher nicht geeignet.

5.2 Server−Software

In diesem Abschnitt wird die im Rahmen der Arbeit untersuchte Server−Software für den WWW−Dienst beschrieben.

5.2.1 ApacheSSL

ApacheSSL ist ein frei verfügbarer Patch für die ebenfalls frei verfügbare und sehr weitverbreitete WWW−Server−Software *Apache*. Auf die vielfältigen Konfigurations− und Erweiterungsmöglichkeiten von Apache soll hier nicht näher eingegangen werden, jedoch wird bei der Propack Data GmbH aufgrund dieser Eigenschaften Apache schon seit längerem erfolgreich als WWW−Server−Software eingesetzt.
ApacheSSL integriert das SSL−Protokoll (Abschnitt 3.5.3) mithilfe der SSLeay−Bibliothek (Abschnitt 5.3.1) in den Apache−Server und stellt noch einige Mechanismen und Optionen zur Nutzung der kryptographischen Erweiterungen zur Verfügung. ApacheSSL läßt sich ohne Lizenzgebühren auch kommerziell benutzen, und es gibt Unterstützung über eine von den Autoren stark frequentierte Mailing−Liste. Die Dokumentation ist ausreichend gut, und ApacheSSL ist auf allen Plattformen lauffähig, für die auch Apache verfügbar ist, derzeit mit Ausnahme von Windows NT. Zudem unterliegt ApacheSSL nicht den U.S.−amerikanischen Export−Restriktionen für kryptographische Software. Das Paket liegt vollständig im Quelltext vor, was eine unabhängige Prüfung durch Dritte möglich macht.
Die Sicherheitscharakteristika lassen sich wir folgt zusammenfassen:

- Engagierte Weiterentwicklung des Apache−Servers (Code−Reviews, Vendor−Initiated Bulletins bei CERT), Quelltexte sind frei verfügbar

- Verwendung starker Verschlüsselungsalgorithmen durch die Verwendung von SSLeay (Abschnitt 5.3.1), keine Exportrestriktionen
- Schutz gegen Mithören durch SSL
- Authentifizierung der Benutzer durch Passwörter und/oder X.509v3 Client−Zertifikate
- verzeichnisbasierte Autorisierungsmechanismen basierend auf IP−Quelladresse, Benutzer und Gruppen und X.509v3 Client−Zertifikaten
- Vielfältige Konfigurations− und Erweiterungsmöglichkeiten durch mächtige Rewrite−Rules, CGI−BINs, (Java−)Servlets, eingebaute Module etc.

All diese Eigenschaften und der bereits erfolgreiche Einsatz von Apache bei der Propack Data GmbH legen den Einsatz von ApacheSSL für gesicherte WWW−Server nahe. Anzumerken sei noch, daß ein Derivat namens *mod_ssl* [Eng98a] existiert, welches dieselbe Funktionalität bietet, aber besser in die Modulstruktur des Apache−Projekts integriert sein soll.

5.2.2 Internet−Dienste mit Lotus Notes

Da *Lotus Notes* bei der Propack Data GmbH zum Einsatz kommt, wurde auch bei dieser Software die Eignung als kryptographisch gesicherter Internet−Server untersucht. Anzumerken ist, daß Lotus Notes die Funktionen eines Messaging−Systems, einer Datenbank und eines Internet−Servers mit einer passenden Programmierumgebung in sich vereint, der direkte Vergleich mit z.B. ApacheSSL (Abschnitt 5.2.1) also zu kurz greift. Deshalb beschränkt sich die Darstellung hauptsächlich auf die wichtigsten Merkmale bezüglich der verfügbaren Internet−Dienste und deren kryptographischer Funktionen. Der ganze Funktionsumfang von Lotus Notes wird nicht beschrieben.
Die Server−Software von Lotus Notes ist auf mehreren Plattformen (Windows NT, OS/2, diverse Unix−Derivate außer Linux) verfügbar, es wird z.Zt. aber vor allem die Entwicklung unter Windows NT forciert. Die Software hat einen sehr hohen Marktanteil und vergleichsweise hohe Zuwachsraten. Zudem gibt es sehr viele Ergänzungsprodukte und Schulungsangebote von Drittanbietern.
Als kommerzielle, U.S.−amerikanische Software liegt Lotus Notes natürlich nicht im Quelltext vor und unterliegt den Exportrestriktionen der USA für kryptographische Software. Daraus folgen zwei sicherheitsrelevante Nachteile:

- Lotus Notes kann nicht von unabhängigen Dritten auf Sicherheitsrisiken überprüft werden. Man kann noch nicht einmal sicher sein, daß nicht absichtlich Hintertüren (Back doors) eingebaut sind.
- Die asymmetrischen Kryptosysteme sind auf eine Schlüssellänge von 512 Bit reduziert, die symmetrischen Kryptosysteme auf 40 Bit (von der Scheinschlüssellänge 64 Bit abgesehen [Kos97]). Diese Schlüsselängen gelten als mit vergleichsweise geringem Aufwand brechbar und sind für den ernsthaften Einsatz nicht geeignet.

Im Hinblick auf die Integration von Internet−Diensten lassen sich folgende Feststellungen treffen:

- Die Integration der im Internet gängigen Standards (SMTP, POP3, HTTP, LDAP, S/MIME, SSL) wird stark vorangetrieben, welche aber vor allem dazu dienen soll, die Migration zu Notes zu erleichtern. Nutzt man typische Notes−Dienste, so ist dies über Internet−Standard−Protokolle nur mit Einschränkungen zu bewerkstelligen.
- Mächtige Erweiterungs−/Programmiermöglichkeiten in eigener Entwicklungsumgebung (NotesScript,aber auch Java) machen komplexe Internet−Anwendungen möglich.

Als grundsätzliche Nachteile von Notes sind zu nennen:

- Notes ist sehr komplex, eine intensive Schulung der Administratoren und Benutzer ist nötig.
- Lotus Notes ist sehr teuer in der Anschaffung. Für jeden Benutzer fallen weitere Lizenzgebühren an, sogar wenn der Zugriff ausschließlich über Internet−Dienste erfolgt.
- Der Ressourcenbedarf ist recht hoch, was die Anschaffung teurer Hardware zur Folge hat.

Die Sicherheitscharakteristika lassen sich wir folgt zusammenfassen:

- Durchgängige Authentifizierung von Benutzern bei allen Diensten mittels Notes−Kennwort.
- Mögliche Authentifizierung eines Benutzers bei Internet−Diensten mittels Benutzer−Zertifikat.
- Zentrale Administration der Zugriffe mittels Autorisierungsschemata für Notes−Datenbanken bzw. −Dokumente.
- Notes−Dokumente können von einem Benutzer signiert werden.
- Die Zugriffe auf alle verfügbaren Internet−Dienste (WWW, POP3, IMAP, LDAP) werden durch SSL gegen Mithören geschützt.

An Notes gefällt vor allem die durchgängige Integration aller Dienste, was die Administration systemweit vereinheitlicht und damit auch sicherheitsrelevante Konfigurationsfehler vermeiden hilft. Der hohe Initialaufwand bei der Anschaffung und Installation läßt Notes aber für manche Anwendungen als zu kostenintensiv erscheinen, es wird also meist "konventionelle" Systeme, wie z.B. kleine WWW−Server für spezielle Anwendungen, nur schwerlich ganz ersetzen. Hat ein Administrator nicht schon sehr viel Erfahrung mit dem System, so wird er bei der Installation auch viele sicherheitsrelevante Fehler machen, was ein nicht unerhebliches Risiko darstellt.

5.3 Software für Zertifizierungsstellen

Verschiedene Softwarepakete kamen für den Betrieb einer Zertifizierungsstelle in Frage. Positiv zu vermerken ist bei der untersuchten Auswahl, daß alle Pakete sich bei Zertifikatdaten an gängige Standards halten (PKCS und X.509, siehe Abschnitt 3.5.1 bzw. Abschnitt 3.3.8.1).

5.3.1 SSLeay

Als Software für die eigene Zertifizierungsstelle (Kapitel 6) wurde *SSLeay* in der Version 0.9.0b eingesetzt. Ursprünglich war diese frei verfügbare Software ein Freizeitprojekt des Australiers Eric Andrew Young und sollte lediglich eine prototypische Implementierung des verschlüsselnden Übertragungsprotokolls SSL werden. Allmählich kamen aber, auch durch andere Mitwirkende, sehr viele neue Funktionen hinzu. SSLeay stellt seine Funktionen auch über eine Programmierschnittstelle zur Verfügung, so daß SSLeay als Krypto−Bibliothek von anderen Programmen genutzt werden kann (siehe auch Abschnitt 5.1.3 und 5.2.1). Zusammen mit den Kommandozeilenprogrammen für die Manipulation kryptographischer Dateien bietet SSleay die Möglichkeit, als Software für eine Zertifizierungsstelle eingesetzt zu werden.

Die Vorteile von SSLeay sind:

- Es fallen keine Lizenzgebühren an, eine kommerzielle Nutzung auch in eigenen Produkten ist kostenfrei erlaubt.
- SSLeay unterliegt keinerlei Exportbeschränkungen, wie sie z.B. bei U.S.−Software üblich sind.
- Mit SSLeay liegt eine Implementierung von SSL Version 2.0 und 3.0 und TLS einschließlich der Verschlüsselungsalgorithmen mit hinreichenden Schlüssellängen vor.
- SSLeay ist als Software−Bibliothek in eigener Software benutzbar, um Netzwerkverbindungen mittels SSL/TLS zu sichern und Client−Zertifikate zu benutzen.
- Es werden sehr verschiedene Hardwareplattformen und verschiedene Betriebssysteme unterstützt.
- Der Quellcode ist frei verfügbar und damit von unabhängigen Personen überprüfbar, was vor allem bei den Implementierungen der Verschlüsselungsalgorithmen wichtig ist.

Aus der Entstehung als Freizeitprojekt resultieren auch verschiedene Nachteile:

- Es gibt nur eine sehr rudimentäre Benutzerschnittstelle auf Shell−Ebene. Abläufe der Zertifizierung müssen in geeigneten Skripten selbst programmiert werden.
- Die Dokumentation seitens der Programm−Autoren ist sehr schlecht und hinkt oft weit der Entwicklung hinterher. Allerdings gibt es eine Reihe von einführenden Publikationen von weiteren Autoren [Hir97][Huds98].
- Die Programmierschnittstellen sind noch starken, teilweise nicht abwärtskompatiblen Änderungen unterworfen.
- Verschlüsselungsalgorithmen müssen für Verwendung in den USA noch lizensiert werden. Die Verschlüsselungsbibliotheken RSARef oder BSAFE können zu diesem Zweck eingebunden werden.

Der ursprüngliche Autor hat die Weiterentwicklung von SSLeay aufgegeben, die Quellen sind von einer Arbeitsgruppe in das Projekt *OpenSSL* übernommen worden, womit die Fortentwicklung erst einmal gesichert scheint.

5.3.2 SECUDE

SECUDE von der GMD Darmstadt ist ebenfalls ein Werkzeugkasten kryptographischer Funktionen und damit direkt vergleichbar mit SSLeay (Abschnitt 5.3.1). Auch SECUDE unterliegt als deutsches Produkt keinen Einschränkungen hinsichtlich der Verschlüsselungsstärke, ist aber nur zu Testzwecken frei verfügbar.
Alle Funktionen sind über die zentrale SECUDE−Shell oder für eigene Programme über die SECUDE−Bibliothek zugänglich. Das Paket enthält alle zum Betrieb einer Zertifizierungsstelle benötigten Funktionen und eine gute Dokumentation und erfährt Unterstützung durch den Hersteller. In dieser Arbeit wurde aus Lizenzgründen SSLeay verwendet, eine Anschaffung von SECUDE für den Betrieb der Zertifizierungsstelle ist allerdings ernsthaft zu erwägen.

5.3.3 Lotus Notes für Zertifizierungsstellen

Auch Lotus Notes hat Funktionen zum Betrieb einer Zertifizierungsstelle eingebaut. Vor allem sind die Funktionen zur Erstellung und Nutzung von Benutzer− und Server−Zertifikaten gut in das Gesamtsystem

integriert. Es gelten aber die in Abschnitt 5.2.2 beschriebenen Sicherheitsbedenken auch und gerade für den Betrieb von Notes als Software für eine Zertifizierungsstelle. Zudem ist die Zertifizierungsstellensoftware in Notes integriert, läuft also zwingenderweise mit auf dem vernetzten Notes−Server, was längst nicht dem für Zertifizierungsstellen erforderlichen Sicherheitsstandard entspricht. Zudem ist die Schlüssellänge durch die U.S.−Exportgesetze limitiert. Daher ist von Lotus Notes als Zertifizierungsstellensystem unbedingt abzusehen.

5.4 Zusammenfassung

In diesem Kapitel wurde bestehende Software hinsichtlich der Integration kryptographischer Funktionen untersucht und beschrieben. Bei der Auswahl der Software stößt man ziemlich schnell auf massive juristische Schwierigkeiten. Zum einen verhindern die U.S.−Exportgesetze den Export ernstzunehmender Krypto−Software, was vor allem bei Client−Software eine echte Behinderung darstellt. Zum anderen sind gerade bei der Benutzung frei verfügbarer Software oft auch patentrechtliche Fragen zu klären. Man muß also beachten, in welchem Teil der Welt man welche Software einsetzt, was insbesondere bei einer internationalen Expansion eines Unternehmens zusätzliche Planungsschwierigkeiten aufwirft.
Da sich der kryptographisch gesicherte WWW−Dienst und die Verwendung von verschlüsselten und signierten Mails als wichtigste Anwendungen bei der Firma Propack Data GmbH darstellen, wurde dabei ein besonderes Gewicht auf die Darstellung des *Netscape Communicator* (Abschnitt 5.1.1) als integrierte WWW− und Mail−Client−Software gelegt. Andere Alternativen für Client−Software wurden nur kurz angerissen, da sie nicht alle benötigten Belange abdecken.
Auch auf der Server−Seite galt der Sicherung des WWW−Dienstes gegen Mithören und der sicheren Authentifizierung von zugreifenden Benutzern besondere Aufmerksamkeit. Es wurden dabei die beiden schon bei der Propack Data GmbH im Einsatz befindlichen Server−Systeme *ApacheSSL* (Abschnitt 5.2.1) und *Lotus Notes* (Abschnitt 5.2.2) miteinander verglichen, soweit dies bei den unterschiedlichen Konzepten möglich war.
Schließlich wurde *SSLeay* (Abschnitt 5.3.1) als vielseitiges kryptographisches Werkzeug und Basisprogrammbibliothek einiger Client− und Server−Software−Pakete ausführlicher dargestellt. Die Funktionen zum Betrieb als Software für eine Zertifizierungsstelle wurden erörtert und denen des Software−Pakets *SECUDE* (Abschnitt 5.3.2) und des integrierten Systems *Lotus Notes* (Abschnitt 5.3.3) gegenübergestellt. Als Ergebnis der Gegenüberstellung wurde SSLeay als Software für den Aufbau einer eigenen Zertifizierungsstelle ausgewählt.

6 Aufbau einer Zertifizierungsstelle

In diesem Abschnitt wird die konkrete Realisierung einer Zertifizierungsstelle beschrieben. Dieser Testaufbau sollte Erfahrungen mit den Abläufen beim Ausstellen und Benutzen von Zertifikaten bringen. Einige prinzipielle Entscheidungen mußten beim Aufsetzen der Zertifzierungsstelle getroffen werden: Zum einen war dies die Topologie und Art der Zertifizierungsstellen, welche entscheidend zur differenzierten Benutzbarkeit der CA−Schlüssel beiträgt (Abschnitt 6.1). Insbesondere wurde auch entschieden, daß ausschließlich die PKI−Teilnehmer selbst Zugriff auf ihre privaten Schlüssel haben sollen (Abschnitt 6.4). Anforderungen an die sichere Handhabung der privaten CA−Schlüssel schlugen sich in einer bestimmten Hardwareanordnung nieder (Abschnitt 6.2). Desweiteren mußten verschiedene Zertifikattypen für verschiedene Anwendungszwecke definiert werden (Abschnitt 6.5).
Schließlich wurde noch die verwendete Software bestimmt: Es kam das Krypto−Paket *SSLeay* für den Betrieb der Zertifizierungsstelle und der *Netscape Communicator* auf Client−Seite zum Einsatz (Abschnitt 6.3). Um dem Anwender die Benutzung der Zertifizierungsstelle zu ermöglichen, mußten noch verschiedene eigene Programme geschrieben werden, die Zertifikatanforderungen entgegennehmen bzw. der Zertifikatbereitstellung dienen (Abschnitt 6.7).

6.1 CA−Topologie

Trotz evtl. in Client−Software auftretenden Problemen mit Zertifikatketten wurde eine zweistufige CA−Hierarchie aufgebaut. Eine eigene Root−CA, deren privater Schlüssel strengsten Sicherheitsvorkehrungen unterliegt (Vergleiche Abschnitt 3.4.6 und 6.2), zertifiziert weitere Unter−Zertifizierungsstellen mit jeweils eigenem privaten Schlüssel, welche jeweils Zertifikate eines bestimmten Typs herausgeben (Abbildung 17). Die Root−CA zertifiziert ausschließlich untergeordnete Zertifizierungsstellen, sonst keine weiteren Objekte.

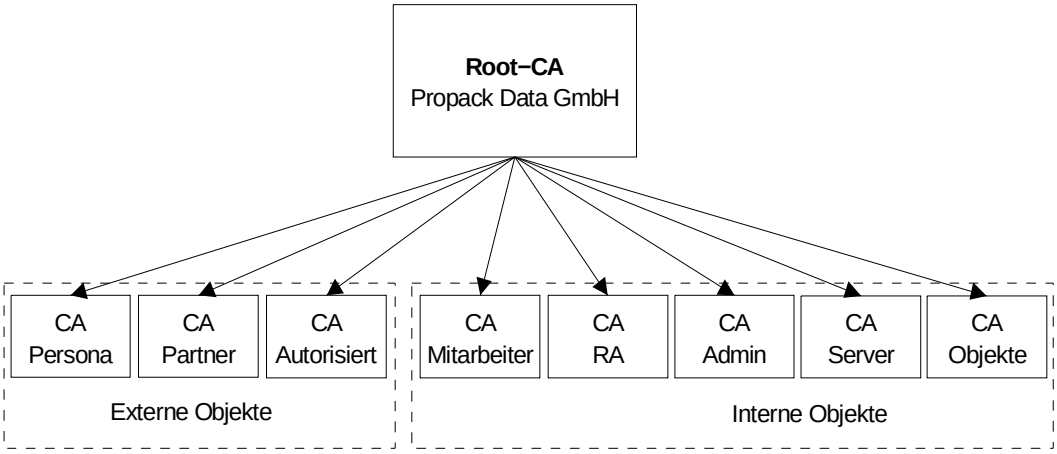


Abbildung 17: Zertifizierungshierarchie der TestCA

Für jeden Zertifikattyp eine eigene CA zu haben, bietet die Möglichkeit, eine Sub−CA ohne weitere menschliche Interaktion Testzertifikate ausstellen zu lassen. D. h. der private Schlüssel dieser Sub−CA kann von einem Zertifizierungsprozeß automatisch benutzt werden, ohne daß ein Administrator ein Kennwort eingeben müßte. Natürlich ist eine solche automatische Handhabung eines privaten Schlüssels ein Sicherheitsrisiko und muß wohlüberlegt eingesetzt werden. Es sollte aber nicht schon durch die CA−Topologie die Möglichkeit einer automatisierten Zertifikaterstellung verhindert werden.
Eine CA je Zertifikattyp erfordert übrigens nicht zwingend eine eigene Root−CA. Man könnte auch lauter selbstsignierte CA−Zertifikate für jeden Zertifikattyp verwenden, was Schwierigkeiten bei der Behandlung von Zertifikatketten vermeiden würde. Die meisten bereits im Netscape Communicator vorhandenen, kommerziellen CA−Zertifikate von Verisign, Thawte etc. sind solche Root−CA−Zertifikate für jeweils einen bestimmten Zertifikattyp. Eine eigene Root−CA, welche jeweils eine Unterzertifizierungsstelle für einen Zertifikattyp zertifiziert, hat aber den Vorteil, daß man nur einmal das Zertifikat der Root−CA mittels out−of−band−Kommunikation sicher übermitteln bzw. überprüfen muß. Die Zertifikate jeder Sub−CA sind danach durch die Signatur überprüfbar.

6.2 Anordnung der CA−Hardware

Die Verläßlichkeit der Zertifizierungsstelle ist vor allem auch von der hinreichend gesicherten Speicherung der Schlüsseldaten abhängig. Es wurde folgende Anordnung für die Speicherung der verschiedenen Daten eingerichtet (Abbildung 18):

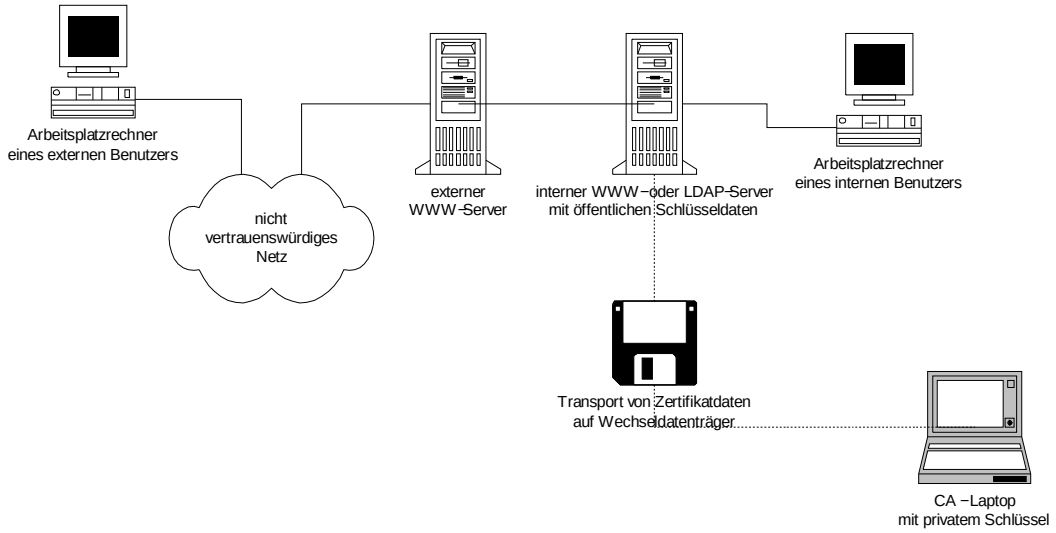


Abbildung 18: Anordnung der Hardware

- Ein extern zugänglicher WWW−Server stellt über CGI−BIN−Programme den Lese−Zugriff auf Zertifikatdaten via HTTP zur Verfügung, speichert aber selbst keinerlei Schlüsseldaten (vergleiche Abschnitt 6.7.3.1). Die CGI−BIN−Programme zur Abfrage von CA−Daten enthalten einen primitiven Proxy−Mechanismus für den Zugriff auf einen internen WWW−Server, bzw. ein LDAP−WWW−Gateway übernimmt die Weiterleitung der externen Anfragen auf einen internen LDAP−Server (7.4.1).
- Interne Benutzer haben direkten Zugriff auf die öffentlich zugänglichen Schlüsseldaten via LDAP oder HTTP. Insbesondere kann ein Benutzer sein Zertifikat auch selbst in seinem eigenen LDAP−Eintrag ablegen (vergleiche Abschnitt 6.7.3.2).

- Die internen WWW− bzw. LDAP−Server speichern nur die öffentlich zugänglichen Schlüsseldaten (CA− und Benutzerzertifikate, Zertifikatwiderruflisten).
- Der eigentliche Zertifizierungsvorgang, bei dem private Schlüsseldaten der Zertifizierungsstellen zur Benutzung kommen, findet auf einem nicht vernetzten Notebook statt, welches in einem Tresor gelagert wird. Jeglicher Datentransport zu und von dem CA−Laptop findet über Wechseldatenträger statt.

Die Außenanbindung ist durch klassisches Firewalling gesichert. Eine detaillierte Darstellung der Firewall−Konfiguration ist aber nicht Gegenstand dieser Arbeit.

6.3 Verwendete Software

Es sollen im folgenden die Gründe für die konkrete Auswahl der Software dargestellt werden. Viele Entscheidungen beim Aufsetzen der eigenen Zertifizierungsstelle und bei der Auswahl der Software sind allerdings auch nichttechnischer Natur, da die praktische Anwendbarkeit im Vordergrund steht:

- Es muß für die Verfahren bereits reale Implementierungen geben, welche schon ein alltagstaugliches Stadium erreicht haben.
- Von Anwendern verwendete Software muß leicht bedienbar sein, damit die Akzeptanz seitens der Anwender gegeben ist.
- Eingesetzte Standards sollten nicht proprietär sein und eine breite Akzeptanz haben, damit der Fortbestand gesichert ist. D. h. es wurde auch darauf geachtet, daß die Standards bei bedeutenden Softwareherstellern eine große Unterstützung erfahren.
- Für den Testaufbau verwendete Software sollte am besten frei verfügbar und der Einsatz auch in einer kommerziellen Umgebung ohne Lizenzprobleme möglich sein.
- Die Software sollte noch eine gewisse Betreuung durch den/die Autor(en) erfahren.
- Der Anwenderkreis für die verwendete Software sollte möglichst groß sein, da man erfahrungsgemäß durch mehr Benutzer an mehr Informationen kommt.
- Die in der Software enthaltenen Verschlüsselungsroutinen sollten nicht durch Exportrestriktionen eingeschränkt sein.

Die Software für die Zertifizierungsstelle wurde auf Basis einer aktuellen *Linux*−Distribution auf einer handelsüblichen PC−Hardware aufgesetzt. Für die Programmierung von Abläufen in Skripten und von außen zugänglichen CGI−BIN−Programmen kam die interpretierte, objektorientierte Sprache *Python* [Löw97] zum Einsatz, welche bei der Propack Data GmbH bereits weite Verwendung in eigenen Produkten findet. Als WWW−Server−Software wurde der weitverbreitete HTTP−Server *Apache* eingesetzt, welcher mit einem frei verfügbaren Patch auch SSL−Verbindungen unterstützt (Abschnitt 5.2.1). Als kryptographisches Werkzeug für die Schlüsselerzeugung und eigentliche Zertifizierung wurde das Paket SSLeay verwendet (Abschnitt 5.3.1). Als Client−Software kam der Netscape Communicator 4.0 und höher zum Einsatz (Abschnitt 5.1.1).

6.4 Schlüsselerzeugung für Benutzer

Die Schlüsselerzeugung erfolgt aus folgenden Gründen beim Benutzer:

- Der Benutzer soll Vertrauen in die Benutzung der PKI entwickeln, um eine größere Akzeptanz der Verschlüsselungstechniken zu erreichen. Dazu werden alle Schritte des Zertifizierungsvorgangs dem Benutzer möglichst transparent gemacht. Insbesondere wird dem Benutzer plausibel gemacht, daß er der alleinige Besitzer seines privaten Schlüssels ist.
- Eine wirklich sichere Löschung von privaten Schlüsseln ist nur mit hohem Aufwand möglich.
- Die Speicherung von Schlüsseldaten muß ebenfalls wieder verschlüsselt erfolgen. Dabei ist die Vergabe eines Initialkennworts erforderlich. Man kann nicht sichergehen, daß der Benutzer auch wirklich allen Kopien der privaten Schlüsseldaten ein neues, eigenes Kennwort vergibt. Erfolgt die Erzeugung und Handhabung des privaten Schlüssels ausschließlich beim Benutzer, so vergibt dieser sofort seine eigenen Kennworte für die Speicherung der privaten Schlüsseldaten, sofern die Software dies geeignet unterstützt.
- Es ist rechtlich strittig, ob der Betreiber einer Telekommunikationsanlage auf Grundlage des TKG dazu verpflichtet werden kann, von seinen Teilnehmern die privaten Schlüssel zur Hinterlegung einzufordern, da diese nicht Bestandteil der Nachrichtenübertragung, sondern des Nachrichteninhalts sind [Kop98]. Erfolgt die Schlüsselerzeugung beim Benutzer, also teilnehmerautonom, so kann man damit u.U. einen Zugriff seitens der Ermittlungsbehörden auf sensible Firmendaten verhindern.

6.5 Zertifikattypen

Die Zertifizierungsstelle stellt verschiedene Kategorien von Zertifikaten für zertifizierte Objekte zur Verfügung, welche jeweils mit einem eigenen privaten CA−Schlüssel signiert werden. Zum Zeitpunkt dieser Arbeit waren Attributzertifikate und deren Verwendung noch nicht genau spezifiziert. Zudem sind Attributzertifikate auch für die geplanten Anwendungen nicht erforderlich, da die Autorisierung immer mittels interner Server−Konfiguration erfolgen kann, d.h. die Rechtevergabe für einen Benutzer kann von internen Administratoren schnell geändert werden und wird on−line überprüft. Die Vergabe von Rechten in einem Attributzertifikat mit einer vergleichsweise langen Gültigkeitsdauer würde bei Änderungen an der Rechtevergabe einen Zertifikatwiderruf nach sich ziehen, was eher umständlich und mit Risiken behaftet ist. Daher stellt die Test−Zertifizierungsstelle nur Identitätszertifikate aus.
Den Zertifikattypen wurden leicht einprägsame Namen gegeben, um den CA−Benutzern die Interpretation der Bedeutung zu erleichtern. Es folgt jeweils eine kurze Beschreibung der einzelnen Zertifikattypen, welche in der jeweiligen Policy vertieft wird. Die Definition eines Zertifikattyps umfaßt jeweils:

Verwendungszweck

Die Beschreibung des Verwendungszwecks soll dem Zertifikatbenutzer die (eingeschränkte) Nutzung eines Zertifikats plausibel machen. Da formale Methoden derzeit noch in der Entwicklung sind und die Definition vor allem von Menschen gelesen wird, ist der Verwendungszweck umgangssprachlich formuliert.

Zertifikataussage

Schließlich wird noch einmal in der Zertifikataussage jeweils die Minimalaussage eines ausgestellten

Zertifikates des angegebenen Typs formuliert, gerade auch um dem Zertifikatbenutzer darzulegen, was das Zertifikat *nicht* aussagt.

Semantik der Zertifikatattribute

Je nach Verwendungszweck haben die Attribute im X.509−Zertifikat eine völlig unterschiedliche Semantik, welche also auch für jeden Zertifikattyp getrennt definiert werden muß. Z.B. ist in Identitätszertifikaten für Personen das CN−Attribut (Common Name) meist Vor− und Nachname der zertifizierten Person, während bei zertifizierten Servern meist der DNS−Name des Servers im CN−Attribut vermerkt ist. Die Semantik ist oft nicht wirklich eindeutig festzulegen, insbesondere gibt es noch keine Standardisierung (vergleiche hierzu auch Abschnitt 3.3.7.2). Ortsangaben in Zertifikaten sollen es einem Zertifikatbenutzer erleichtern, zu beurteilen, wohin (geographisch) eine Nachricht für die angegebene E−Mail−Adresse geschickt wird (Land, Staat und Ort) bzw. wo (geographisch) ein Server steht. Namen sollen möglichst gut die Identität und Funktion des zertifizierten Objekts beschreiben.
Desweiteren werden die erweiterten X.509v3−Attribute für die Verwendung mit dem Netscape Communicator (Abschnitt 3.3.8.1.1 und 5.1.1) für den jeweiligen Zertifikattyp geeignet gesetzt (siehe SSLeay−Konfigurationsdatei *ssleay.cnf* in Anhang C).

6.5.1 »Persona«

6.5.1.1 Verwendungszweck

Die Richtlinie »Persona« erlaubt das anonyme Zertifizieren einer Mail−Adresse. Beim Ausstellen dieser Zertifikate wird lediglich die Mail−Adresse auf Funktionsfähigkeit geprüft. Die Identität des Objektes wird explizit *nicht* geprüft, insbesondere findet keinerlei weitere out−of−band−Prüfung statt. Diese Zertifikate haben nur eine Gültigkeitsdauer von sieben Tagen und sind nur für das Verschicken und Signieren von E−Mail gedacht. Solche Zertifikate stellen für den CA−Benutzer eine Möglichkeit dar, in dringenden Fällen ohne Mithilfe der CA−Betreuer schnell ein Zertifikat zu erhalten, also Mail in dringenden Fällen verschlüsseln zu können. Vom Anwender zu beachten ist, daß diese Zertifikate noch nicht einmal bestätigen, daß der im Zertifikat enthaltene öffentliche Schlüssel auch wirklich zur E−Mail−Adresse gehört. Der Name »Persona« für diese Art von anonymen Zertifikaten wurde in [RFC1422] festgelegt.

6.5.1.2 Zertifikataussage

Mit einem Zertifikat »Persona« wird von der TestCA bestätigt, daß zum Zeitpunkt der Zertifizierung ein unbekanntes Objekt in der Lage war, bei der angegebenen E−Mail−Adresse Mail abzuholen.

6.5.1.3 Semantik der Zertifikatattribute

Da die einzelnen Angaben des Zertifikatantragstellers nicht geprüft werden, haben die einzelnen Attribute keine definierte Semantik. Es obliegt völlig dem Zertifikatantragsteller, ob er überhaupt sinnvolle Angaben macht.

6.5.2 »Partner«

6.5.2.1 Verwendungszweck

Zertifikate der Richtlinie »Partner« weisen die Richtigkeit einer Mail−Adresse externer Partner (Kunden, Projektpartner, Logo−Partner, externe E−Mail−Adressen von Mitarbeitern etc.) aus und erlauben das

Versenden verschlüsselter E−Mail an die angegebenen Personen. Bei diesen Zertifikaten muß eine hinreichende Überprüfung der Identität durch die Zertifizierungsstelle selbst oder eine regionale Stelle (RA) erfolgen. Diese Zertifikate sind nur für das Verschlüsseln und Signieren von E−Mail gedacht und sollen zur sicheren Nutzung von E−Mail im Geschäftsablauf mit externen Teilnehmern sorgen.

6.5.2.2 Zertifikataussage

Mit einem Zertifikat »Partner« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu der angegebenen Person und E−Mail−Adresse gehört, die Mail−Adresse richtig ist und diese Person in einer, nicht näher bezeichneten, Beziehung zur Propack Data GmbH steht.

6.5.2.3 Semantik der Zertifikatattribute

Land (X.509−Attribut C):
ISO−Kürzel des Landes, im Sinne eines autonomen Staates, in welchem die angegebene Person und Organisation bzw. Abteilung sich befindet. Es wird die Angabe des Landes akzeptiert, in dem sich die Person meistens aufhält.
Staat (X.509−Attribut ST):
Bundesstaat, Bundesland oder Provinz, wo sich die angegebene Person meistens aufhält oder sich die angegebene Organisation bzw. Abteilung befindet.
Ort (X.509−Attribut L):
Ort (Stadt), wo sich die angegebene Person meistens aufhält oder sich die angegebene Organisation bzw. Abteilung befindet.
Organisation (X.509−Attribut O):
Der wohlbekannte, offiziell im Geschäftsablauf verwendete Name der Organisation (Firma, Institut, Behörde etc.), bei der die angegebene Person beschäftigt ist.
Abteilung (X.509−Attribut OU):
Der wohlbekannte, organisationsintern vergebene Name derjenigen Abteilung innerhalb der o.g. Organisation, in welcher die angegebene Person arbeitet.
Name (X.509−Attribut CN):
Der allgemein verwendete Name, unter dem die Person speziell innerhalb ihrer Organisation, aber auch außerhalb bekannt ist.
E−Mail (X.509−Attribut Email):
Die persönliche E−Mail−Adresse, unter der nur die angegebene Person Nachrichten empfängt.

6.5.3 »Autorisiert«

6.5.3.1 Verwendungszweck

Zertifikate der Richtlinie »Autorisiert« erlauben externen Objekten den eingeschränkten Zugriff auf interne Ressourcen. Das Ausstellen des Zertifikates ist lediglich die notwendige Voraussetzung für einen Zugriff. Der auf einem solchen Zertifikat beruhende autorisierte Zugriff wird im Einzelfall getrennt eingerichtet, was nicht Aufgabe der Zertifizierungsstelle ist. Bei diesen Zertifikaten muß eine hinreichende Überprüfung der Identität durch die Zertifizierungsstelle selbst oder eine regionale Stelle (RA) erfolgen. Die Zertifikate dienen der Authentifizierung gegenüber Servern (z.B. WWW−, LDAP− und Mail−Server etc.).

6.5.3.2 Zertifikataussage

Mit einem Zertifikat »Autorisiert« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu der angegebenen Person und E−Mail−Adresse gehört und diese Person in einer, nicht näher bezeichneten, Beziehung zur Propack Data GmbH steht.

6.5.3.3 Semantik der Zertifikatattribute

Land (X.509−Attribut C):
ISO−Kürzel des Landes, im Sinne eines autonomen Staates, in welchem die angegebene Person und Organisation bzw. Abteilung sich befindet. Es wird die Angabe des Landes akzeptiert, wo sich die Person meistens aufhält.
Staat (X.509−Attribut ST):
Bundesstaat, Bundesland oder Provinz, wo sich die angegebene Person meistens aufhält oder sich die angegebene Organisation bzw. Abteilung befindet.
Ort (X.509−Attribut L):
Ort (Stadt), wo sich die angegebene Person meistens aufhält oder sich die angegebene Organisation bzw. Abteilung befindet.
Organisation (X.509−Attribut O):
Der wohlbekannte, offiziell im Geschäftsablauf verwendete Name der Organisation (Firma, Institut, Behörde etc.), bei der die angegebene Person beschäftigt ist.
Abteilung (X.509−Attribut OU):
Der wohlbekannte, organisationsintern vergebene Name derjenigen Abteilung innerhalb der o.g. Organisation, in welcher die angegebene Person arbeitet.
Name (X.509−Attribut CN):
Der allgemein verwendete Name, unter dem die Person speziell innerhalb ihrer Organisation, aber auch außerhalb bekannt ist.
E−Mail (X.509−Attribut Email):
Die persönliche E−Mail−Adresse, unter der nur die angegebene Person Nachrichten empfängt. Zu beachten ist, daß die E−Mail−Adresse nur informativen Charakter hat, da das Zertifikat nicht zum Verschlüsseln von Nachrichten gedacht ist.

6.5.4 »Mitarbeiter«

6.5.4.1 Verwendungszweck

Zertifikate der Richtlinie »Mitarbeiter« werden nur für Mitarbeiter ausgestellt und dienen u.a. dazu, die Zugehörigkeit zur Propack Data GmbH nachzuweisen. Auch hier wird der evtl. benötigte autorisierte Zugriff auf Ressourcen getrennt eingerichtet. Die Überprüfung der Identität muß intern durch die Zertifizierungsstelle selbst erfolgen. Wenn ein Mitarbeiter die Firma verläßt, muß sein Mitarbeiter−Zertifikat widerrufen werden.

6.5.4.2 Zertifikataussage

Mit einem Zertifikat »Mitarbeiter« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu der angegebenen Person und E−Mail−Adresse gehört und diese Person Mitarbeiter der Propack Data GmbH ist.

6.5.4.3 Semantik der Zertifikatattribute

Land (X.509−Attribut C):
ISO−Kürzel des Landes, im Sinne eines autonomen Staates, in welchem sich die dem Mitarbeiter zugeordnete Geschäftsstelle befindet.
Staat (X.509−Attribut ST):
Bundesstaat, Bundesland oder Provinz, wo sich die dem Mitarbeiter zugeordnete Geschäftsstelle befindet.
Ort (X.509−Attribut L):
Ort (Stadt), wo sich die dem Mitarbeiter zugeordnete Geschäftsstelle befindet.
Organisation (X.509−Attribut O):
Der wohlbekannte, offiziell im Geschäftsablauf verwendete Name der Firma Propack Data GmbH.
Abteilung (X.509−Attribut OU):
Ein oder mehrere den QM−Richtlinien entsprechende Namen einer oder mehrerer Abteilungen innerhalb der Firma Propack Data GmbH, welchen der Mitarbeiter zugeordnet ist.
Name (X.509−Attribut CN):
Der allgemein verwendete Name, unter dem der Mitarbeiter bekannt ist.
E−Mail (X.509−Attribut Email):
Die persönliche E−Mail−Adresse des Mitarbeiters, in der Regel von der Form [vorname].[nachname]@propack−data.de.
Initialen (X.509−Attribut I):
Das eindeutige, intern vergebene Namenskürzel des Mitarbeiters.

6.5.5 »RA«

6.5.5.1 Verwendungszweck

Da es aufgrund der räumlichen Verteilung der zu zertifizierenden Objekte (Geschäftsstellen, Kunden auch im Ausland etc.) für zentrale CA−Betreuer nahezu unmöglich ist, die Identität aller Objekte sicher zu prüfen, muß diese Prüfung an Personal vor Ort delegiert werden können. Zu diesem Zweck bekommen entsprechend geschulte Mitarbeiter zusätzliche Zertifikate zur Arbeit als *Regionale Identitätsprüfstelle* (Regional Authority). Nur Inhaber eines solchen Zertifikats, im folgenden *RA−Beauftragte* genannt, gelten als ausreichend kompetent, eine hinreichende Identitätsüberprüfung durchzuführen. Die Angaben zur Identität von Dritten sind mit einem »RA«−Zertifikat unterzeichnet an die zentralen CA−Betreuer zu übermitteln. Dieses Zertifikat dient weder dem Zugriff auf Ressourcen noch dem Signieren persönlicher Nachrichten einer Person, sondern lediglich dem Signieren von E−Mail für die Übermittlung von Identitätsangaben. Insbesondere stellt eine RA auch keine Zertifikate aus. Wenn ein RA−Beauftragter die Firma verläßt, muß sein RA−Zertifikat widerrufen werden.

6.5.5.2 Zertifikataussage

Mit einem Zertifikat »RA« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu der angegebenen Person und E−Mail−Adresse gehört, diese Person Mitarbeiter der Propack Data GmbH ist und diese Person ausreichend kompetent ist, eine für die Zertifizierung hinreichend zuverlässige Identitätsprüfung zu zertifizierender Objekte durchzuführen

6.5.5.3 Semantik der Zertifikatattribute

Die Semantik der Zertifikat−Attribute ist gleich denen des Zertifikattyps »Mitarbeiter« (Abschnitt 6.5.4).

6.5.6 »Admin«

6.5.6.1 Verwendungszweck

Der Zertifikattyp »Admin« zertifiziert Administratoren bei der Firma Propack Data GmbH. Auch hier wird die Autorisierung zur Durchführung bestimmter Aufgaben im Einzelfall geprüft und eingerichtet. Bei diesen Zertifikaten muß eine hinreichende Überprüfung der Identität durch die Zertifizierungsstelle selbst erfolgen. Dieses Zertifikat wird zusätzlich zum normalen Mitarbeiterzertifikat vergeben und dient *nicht* dem Signieren von Mails, sondern dem Zugriff auf Ressourcen ausschließlich zu administrativen Zwecken. Alle CA−Benutzer werden darüber in Kenntnis gesetzt, daß es mit solchen Zertifikaten unterschriebene E−Mails definitionsgemäß nicht gibt. Nachrichten von Administratoren an Anwender müssen mit deren Mitarbeiterzertifikat signiert sein, die administrative Rolle des Unterzeichners muß dem Empfänger vorher bekannt sein, ansonsten muß er dem Inhalt mißtrauen. Wenn ein Administrator die Firma verläßt bzw. die Rolle als Administrator nicht mehr inne hat, muß sein Admin−Zertifikat widerrufen werden.

6.5.6.2 Zertifikataussage

Mit einem Zertifikat »Admin« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu der angegebenen Person und E−Mail−Adresse gehört, diese Person Mitarbeiter der Propack Data GmbH ist und auf nicht näher bezeichneten Servern administrative Aufgaben wahrnimmt.

6.5.6.3 Semantik der Zertifikatattribute

Die Semantik der Zertifikat−Attribute ist gleich denen des Zertifikattyps »Mitarbeiter« (Abschnitt 6.5.4).

6.5.7 »Server«

6.5.7.1 Verwendungszweck

Ein weiterer Zertifikattyp »Server« ist für die Zertifizierung von eigenen Servern der Firma Propack Data GmbH gedacht. Im Zertifikat selbst wird im CN−Attribut der DNS−Name oder DNS−Alias des Servers untergebracht. Um dem Benutzer die Prüfung der Plausibilität zu erleichtern, sollten zentralen Servern einprägsame DNS−Aliase für bestimmte Dienste gegeben werden (z.B. www.domain.com, pop3.domain.com, ldap.domain.com).

6.5.7.2 Zertifikataussage

Mit einem Zertifikat »Server« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu dem im CN−Attribut angegebenen DNS−Namen gehört und das Zertifikat nur auf einem Rechner mit diesem DNS−Namen Verwendung findet.

6.5.7.3 Semantik der Zertifikatattribute

Land (X.509−Attribut C):
ISO−Kürzel des Landes, im Sinne eines autonomen Staates, in welchem sich der Standort des Servers befindet.
Staat (X.509−Attribut ST):
Bundesstaat, Bundesland oder Provinz, wo sich der Standort des Servers befindet.

Ort (X.509−Attribut L):
Ort (Stadt), wo sich der Standort des Servers befindet.
Organisation (X.509−Attribut O):
Der wohlbekannte, offiziell im Geschäftsablauf verwendete Name der Firma Propack Data GmbH.
Abteilung (X.509−Attribut OU):
Ein den QM−Richtlinien entsprechender Name der für den Server zuständigen Abteilung innerhalb der Firma Propack Data GmbH.
Name (X.509−Attribut CN):
Der DNS−Name des Servers. Insbesondere ist darauf zu achten, daß ein den Dienst beschreibender DNS−Alias eingesetzt wird und dieser Name über das Anwendungsprotokoll (HTTP, POP3 o. ä.) der Client−Software korrekt übermittelt wird.
E−Mail (X.509−Attribut Email):
Die generische E−Mail−Adresse des Server−Administrators, z.B. www@propack−data.de.

6.5.8 »Objekte«

6.5.8.1 Verwendungszweck

Gesicherte Softwareverteilung über das Internet wird speziell bei dynamisch ladbarem Code, wie z.B. Java− und JavaScript−Code, immer wichtiger. Zu diesem Zweck können einzelne Mitarbeiter als "Object Signers" zertifiziert werden. Diese Mitarbeiter sind berechtigt, Software der Propack Data GmbH als gültigen, getesteten Release−Code zu zertifizieren, dessen Signatur dann von einem Benutzer vor der Ausführung überprüft werden kann. Diese Zertifikate dienen ausschließlich nur dem Signieren von Programm−Code, können also weder zur Authentifizierung gegenüber Server−Diensten, noch zum Signieren von E−Mail−Nachrichten verwendet werden.

6.5.8.2 Zertifikataussage

Mit einem Zertifikat »Objekte« wird von der TestCA für die Gültigkeitsdauer des Zertifikates bestätigt, daß der öffentliche Schlüssel zu der angegebenen Person und E−Mail−Adresse gehört, diese Person Mitarbeiter der Propack Data GmbH ist und diese Person zur Zertifizierung von Software der Propack Data GmbH berechtigt ist.

6.5.8.3 Semantik der Zertifikatattribute

Die Semantik der Zertifikat−Attribute ist gleich denen des Zertifikattyps »Mitarbeiter« (Abschnitt 6.5.4).

6.6 Erzeugung der CA−Hierarchie

In diesem Abschnitt wird die Erzeugung der CA−Hierarchie mit SSLeay erläutert. Zu beachten ist, daß an dieser Stelle CA−Schlüsselpaare erzeugt werden und die Handhabung der privaten Schlüsseldaten höchster Sorgfalt bedarf.

6.6.1 Handhabung der privaten CA−Schlüssel

Die privaten Schlüsselteile dürfen auf keinen Fall kompromittiert werden. Das Konzept sieht vor, daß der Vorgang der CA−Schlüsselerzeugung auf einem nicht vernetzten, mobilen Rechner (Notebook) geschieht, welcher nach der Erzeugung der CA−Hierarchie und Kopieren aller öffentlichen CA−Daten der

Hierarchie auf einen Wechseldatenträger in einen nur der Geschäftsleitung zugänglichen Tresor gelegt wird. Kopien der Festplatte zu Datensicherungsmaßnahmen werden ausschließlich von Mitgliedern der Geschäftsleitung angefertigt oder zumindestens persönlich von der Geschäftsleitung überwacht. Danach werden die Datensicherungsmedien von der Geschäftsleitung in einem Bankschließfach deponiert.

6.6.2 Erzeugung der CA−Hierarchie mit SSLeay

6.6.2.1 Verzeichnisstruktur

Es muß zunächst eine geeignete Verzeichnisstruktur für die CA−Hierarchie angelegt und in der Konfigurationsdatei von SSLeay eingetragen werden (Anhang C). Zu beachten ist, daß verschiedene Sub−CAs auch auf verschiedenen Rechnern liegen können.

6.6.2.2 Root−CA

Die Erzeugung des Root−CA−Schlüsselpaares und selbstsignierten Root−CA−Zertifikates im Verzeichnis der Root−CA wird mit folgendem Kommando ausgelöst:

```
ssleay req −newkey rsa:2048 −x509 −days 300 −out cacert.pem.nofix −keyout private/cakey.pem
```

Die Erzeugung einer Zertifikatanforderung fordert auch immer zur Eingabe der Passphrase für den privaten Schlüssel und zusätzlicher Informationen für den Zertifikat−DN auf. Bei der Erzeugung des Zertifikats der Root−CA muß die Option *−x509* angegeben werden, welche dafür sorgt, daß statt einer PKCS#10−Zertifikatanforderung gleich ein selbstsigniertes X.509−Zertifikat erzeugt wird. Der Parameter *−newkey rsa:2048* veranlaßt die Erzeugung eines neuen RSA−Schlüsselpaares der Länge 2048 Bit. Danach sollte zur korrekten Verwendung des CA−Zertifikates mit Netscape Communicator das erweiterte X.509v3−Attribut *nscerttype* mit dem Programm *ca−fix* gesetzt werden [Hens98c].

```
ca−fix −in cacert.pem.nofix −out cacert.pem −caset −nscertype 0x07 −inkey private/cakey.pem
```

Zum Einbringen des CA−Zertifikates in einen der gängigen WWW−Browser ist die Umwandlung ins DER−Format nötig.

```
ssleay x509 −in cacert.pem −outform DER −out cacert.der
```

6.6.2.3 Sub−CA

Die Erzeugung der Sub−CA−Zertifikatanforderungen (PKCS#10−Format) geschieht im jeweiligen Sub−CA−Verzeichnis:

```
ssleay req −newkey rsa:2048 −out careq.pem −keyout private/cakey.pem
```

Danach kann das Sub−CA−Zertifikat von der Root−CA ausgestellt werden:

```
ssleay ca −name CA_CA −in careq.pem −out cacert.pem.nofix
```

Das CA−Zertifikat einer Sub−CA sollte mit *ca−fix* auf den jeweiligen Verwendungszweck eingeschränkt werden:

```
ca−fix −in cacert.pem.nofix −out cacert.pem −caset −nscertype [CA−Typ] −inkey [Verzeichnis der
Root−CA]/private/cakey.pem
```

Danach erfolgt wieder die Umwandlung ins DER−Format:

```
ssleay x509 −in cacert.pem −outform DER −out cacert.der
```

6.7 Benutzung der Zertifizierungsstelle

Es folgt eine detailliertere Darstellung der technischen Zusammenhänge bei der Benutzung der Zertifizierungsstelle.

6.7.1 Anerkennen der Zertifizierungsstelle

Der Anwender muß zunächst das selbstsignierte Zertifikat der obersten Zertifizierungsstelle (Root−CA) laden und akzeptieren. Der Vorgang ist für jedes CA−Zertifikat der Sub−Zertifizierungsstellen zu wiederholen.
Dies geschieht in unserem Fall, indem der Benutzer durch Anwählen der betreffenden URL ein CGI−BIN−Programm aufruft (HTTP−GET), welches ihm via HTTP das selbstsignierte CA−Zertifikat im DER−Format mit dem MIME−Typ *application/x−x509−ca−cert* gekennzeichnet zurückliefert (Abbildung 19).

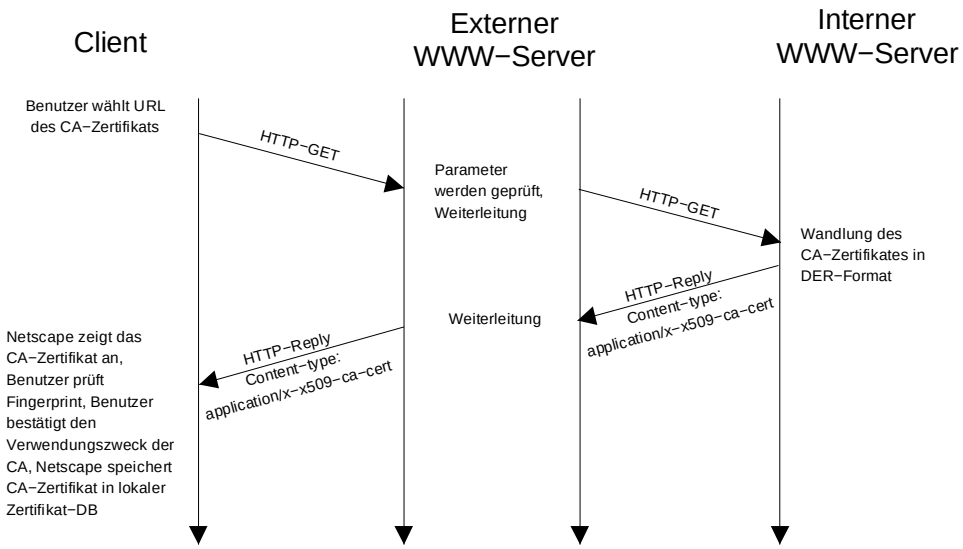


Abbildung 19: Laden des CA−Zertifikates

Das in Python geschriebene CGI−BIN−Programm *get−cert.py* hat einen Proxy−Modus: Ist im Python−Quellcode in der Variable *internalurl* eine URL gesetzt, so wird die GET−Anfrage an diese URL weitergeleitet und das Ergebnis dieser HTTP−Anfrage an den Client gegeben.

Nach Empfang des CA−Zertifikates wird der Anwender in einem Dialog des Netscape Navigator aufgefordert zu entscheiden, ob und wie weit er dieser Zertifizierungsstelle vertrauen möchte. In unserem Fall muß der Anwender mindestens der Zertifizierungsstelle für das Ausstellen von E−Mail− und Server−Zertifikaten vertrauen. Das CA−Zertifikat erscheint dann in der Sicherheitsinformation bei den CA−Zertifikaten (siehe auch Abschnitt 5.1.1.6.4).
Zur weiteren Überprüfung der Echtheit des selbstsignierten RootCA−Zertifikates muß der Anwender sich den Fingerprint des CA−Zertifikates anzeigen lassen und sich telefonisch an den CA−Administrator wenden. Der Anwender kann auch eine Mail an *ca−admin@propack−data.de* schicken, um einen telefonischen Rückruf oder die Übermittlung des TestCA−Fingerprints via Fax oder Briefpost zu

veranlassen. Eine weitere Möglichkeit ist die Einführung und Betreuung eines neuen Anwenders direkt vor Ort. Bei einer solchen Gelegenheit kann ein Betreuer (z.B. ein RA−Beauftragter) das Root−CA−Zertifikat oder dessen Fingerprint bereits auf einem Wechseldatenträger oder in gedruckter Form mitbringen (vergleiche hierzu auch Abschnitt 3.4.4).
Nach der sicheren Übermittlung bzw. Überprüfung des Root−CA−Zertifikates können die jeweiligen Sub−CA−Zertifikate mit dem Root−CA−Zertifikat überprüft werden.

6.7.2 Zertifizierungsablauf

Der Zertifizierungsablauf ist in mehrere Schritte aufgeteilt:

1. Zertifikatanforderung
2. Überprüfung der Identität des zu zertifizierenden Objekts
3. Ausstellen des Zertifikates durch die Zertifizierungsstelle
4. Zertifikatbereitstellung an den Antragsteller

Die einzelnen Schritte werden in den folgenden Abschnitten näher beschrieben.

6.7.2.1 Zertifikatanforderung

Die *Zertifikatanforderung* (certificate request) ist die Übermittlung von Identitätsdaten und öffentlichem Schlüssel an die Zertifizierungsstelle. Im Falle von Benutzerzertifikaten ist die Zertifikatanforderung in mehrere Schritte aufgeteilt:

1. Übermittlung von Angaben über die Identität des Antragstellers mittels HTML−Formular an die Zertifizierungsstelle.
2. Erzeugung des Schlüsselpaares auf Benutzerseite, angestoßen durch ein <KEYGEN>−Tag in einem zweiten HTML−Formular, und Übermittlung der Identitätsangaben zusammen mit dem öffentlichen Schlüssel.
3. E−Mail−Dialog mit Zufalls−ID, um die Funktionsfähigkeit der angegebenen E−Mail−Adresse zu prüfen.

Der Vorgang läuft über einen zweistufigen Dialog des CGI−BIN−Programms *ns−enroll.py* ab, welches ebenfalls in Python geschrieben ist. Dieses CGI−BIN−Programm ist primär für die Nutzung mit *Netscape Navigator* auf der Client−Seite gedacht, andere Browser unterstützen das verwendete <KEYGEN>−Tag nicht unbedingt.

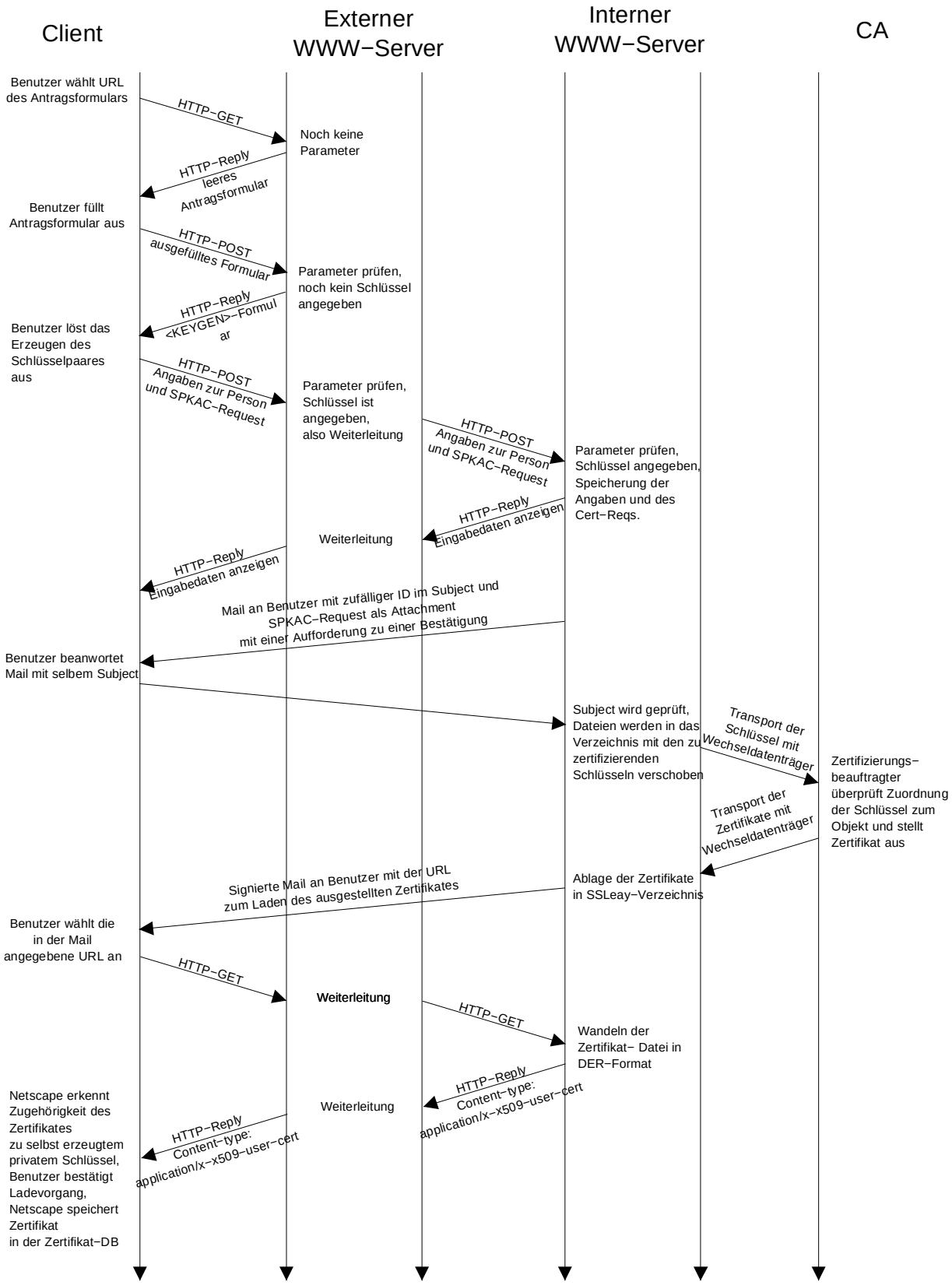


Abbildung 20: Zeitlicher Ablauf der Zertifizierung

6.7.2.1.1 Angabe von Identitätsdaten

Zuerst muß der Benutzer seine persönlichen, für den Vorgang notwendigen Daten im Anmeldeformular des CGI−BIN−Programms *ns−enroll.py* eintragen (Abbildung 21).

Abbildung 21: HTML−Formular zur Angabe der Identitätsdaten

Mit Druck auf den Button *Senden* am Ende des Formulars werden die Angaben zur Person an den Server übermittelt. Bei der Formularbearbeitung wird auch auf die strikte Einhaltung bestimmter Formatvorgaben geachtet und bei Fehleingaben der Benutzer entsprechend zur Berichtigung aufgefordert. Zu beachten ist, daß die folgenden Felder Bestandteil des Zertifikates werden:

- Land (X.509−Attribut C, LDAP−Attribut c)
- Staat, im Sinne von Bundesstaat oder Bundesland (X.509−Attribut ST, LDAP−Attribut st)
- Ort (X.509−Attribut L, LDAP−Attribut l)
- Organisation (X.509−Attribut O, LDAP−Attribut o)
- Abteilung (X.509−Attribut OU, LDAP−Attribut ou)
- Name (X.509−Attribut CN, LDAP−Attribut cn)

- E−Mail (X.509−Attribut Email, LDAP−Attribut mail)
  Meist ist die E−Mail−Adresse global eindeutig, so daß, selbst wenn alle anderen Felder zu einem weiteren Zertifikat−DN identisch sind, der gesamte aus diesen Attributen zusammengesetzte Zertifikat−DN eindeutig wird. Daher und auch für den weiteren Ablauf ist die Angabe der E−Mail−Adresse zwingend vorgeschrieben. Die Domäne der E−Mail−Adresse wird im DNS geprüft, um schon einmal triviale Fehleingaben bzw. dilettantischen Mißbrauch zu verhindern.

Der Anwender muß darauf hingewiesen werden, nur die Zertifikat−Felder auszufüllen, welche von seiner Seite als datenschutzrechtlich nicht bedenklich einzustufen sind.

Die folgenden Felder dienen nur zur Identitätsprüfung bzw. sind CA−Parameter und werden nur intern auf einem LDAP−Server gespeichert. Der Zugriff auf diese Daten ist nur den CA−Beauftragten gestattet.

- Telefonnr. (LDAP−Attribut telephonenumber)
  im internationalen Format +[Ländercode]−[Bereichsvorwahl]−[Teilnehmerkennung]
- Fax (LDAP−Attribut facsmilenumber)
  im internationalen Format +[Ländercode]−[Bereichsvorwahl]−[Teilnehmerkennung]
- Ansprechpartner (nicht standardisiertes LDAP−Attribut contactPerson)
  Ein interner Ansprechpartner, welcher hilfreich bei der Identitätsprüfung sein kann.
- Gewünschte Gültigkeitsdauer (nicht standardisiertes LDAP−Attribut)
  Hier kann eine gewünschte Gültigkeitsdauer für das Zertifikat gewählt werden.
- Kennwort (LDAP−Attribut userpassword)
  Zweimalige, verdeckte Eingabe eines Kennworts für einen Zertifikatwiderruf und für den Challenge−Wert des <KEYGEN>−Tags.

6.7.2.1.2 Schlüsselerzeugung

Sind alle vorher gemachten Eingaben formal in Ordnung, so erscheint ein neues Formular, welches diese Angaben zur Kontrolle noch einmal anzeigt, die Angaben als versteckte Eingabefelder (<INPUT TYPE=hidden>) enthält und ein Auswahlfeld für die Schlüssellänge bereitstellt. Dieses Auswahlfeld entsteht durch das Netscape−spezifische HTML−Tag <KEYGEN> und stößt die Schlüsselerzeugung in Netscape an [Nets97f]. Als Parameter CHALLENGE enthält das <KEYGEN>−Tag auch das vorher eingegebene Kennwort. Mit Druck auf den Knopf *Schlüssel erzeugen* wird die Erzeugung eines Schlüsselpaares in Netscape angestoßen und der öffentliche Teil des Schlüsselpaares zusammen mit den versteckten Identitätsangaben an den Server übermittelt. Der private Teil des Schlüsselpaares verbleibt beim Benutzer. Seitens des Servers wird die Zertifikatanforderung, bestehend aus Angaben für das Zertifikat und dem öffentlichen Schlüssel, als sog. *SPKAC−Request* gespeichert, und alle Identitätsangaben werden in einem speziellen Format für das Programm *ldapadd* zur späteren Speicherung auf einem LDAP−Server abgelegt. Diese Dateien werden vom CGI−BIN−Programm *ns−enroll*.py in ein Verzeichnis geschrieben, welches für den Benutzer *wwwrun*, unter welchem der WWW−Server läuft, nur schreibbar ist.

Beispiel für einen SPKAC−Request (Zeichensatz ISO−8859−1, SPKAC−Eintrag gekürzt):

```
countryName = DE
stateOrProvinceName = Baden-Württemberg
localityName = Karlsruhe
organizationName = Universität Karlsruhe
organizationalUnitName = Informatik
commonName = Michael Ströder
initials = MS
```

```
uid = stroeder
emailAddress = michael@ms.my
SPKAC = MIIBQDCBqjCBnzA [..]
1Y6KaQkfucVzjcURj4EmMdl2Tk0tT7fVmi0D/NR4dy4N8OCLjR9nPMmHCaR6us
```

Beispiel für eine ldapadd−Datei (Zeichensatz UTF−8):

```
dn: cn=Michael Ströder,mail=michael@ms.my,o=Universität Karlsruhe
objectclass: top
objectclass: person
objectclass: strongauthenticationuser
certselect: Persona
c: DE
st: Baden−Württemberg
o: Universität Karlsruhe
l: Karlsruhe
ou: Informatik
mail: michael@ms.my
cn: Michael Ströder
postalcode: 76128
initials: MS
telephoneNumber: +49−721−123456
facsimileTelephoneNumber: +49−721−654321
contactPerson: Mr. Nobody
streetaddress: Kaiserstr. 12
userpassword: geheim
uid: stroeder
```

6.7.2.1.3 E−Mail−Dialog mit Zufalls−ID

Nach Empfang und vorläufiger Speicherung der Zertifikatsanforderung schickt der Server eine E−Mail mit einer zufälligen ID an die vom Benutzer angegebene E−Mail−Adresse. Die Mail enthält die vorher gemachten Angaben (ohne Kennwort) und ist vom Zertifikatantragsteller innerhalb von 24 Std. mit dem gleichen Betreff (Subject) zu beantworten. Geschieht dies nicht, so wird die Zertifikatanforderung gelöscht, und der Zertifizierungsablauf ist abgebrochen. Dies dient dazu, die E−Mail−Adresse auf Funktionsfähigkeit zu testen und zu verhindern, daß allzu leicht Zertifikate für Personen von Dritten angefordert werden können. Natürlich schützt dies nicht vor einem echten Angriff der Art Man−in−the−middle−attack und beweist auch nicht ernsthaft die Zugehörigkeit des öffentlichen Schlüssels in der Zertifikatanforderung zu der E−Mail−Adresse. Es schließt nur einen Mißbrauch durch technisch nicht versierte Benutzer aus.
Beantwortet der Benutzer die Mail, so werden die gespeicherte Zertifikatanforderung und die LDAP−Datei in ein weiteres Spool−Verzeichnis verschoben. Erst jetzt ist die Zertifikatanforderung gültig, und der Antrag wird bearbeitet.

6.7.2.2 Identitätsprüfung

Der Idealfall wäre, wenn der Benutzer seinen öffentlichen Schlüssel mit einer Diskette oder einem ähnlichen Wechseldatenträger bei der Zertifizierungsstelle oder einer regionalen Stelle vorlegen würde (Abschnitt 3.2.2.5). Leider gibt es diese Möglichkeit bei der Schlüsselerzeugung im Netscape Communicator nicht, sie erfolgt zwingenderweise in einem ungesicherten Dialog mit einem WWW−Server (Abschnitt 5.1.1.6.1).
Zur Prüfung der Identität des Zertifikatantragstellers stehen der Zertifizierungsstelle folgende Daten zur Verfügung:

- Im Antragsformular gemachte Angaben zur Person, welche erst einmal als nicht vertrauenswürdig anzusehen sind.
- Angaben zur Person, welche im normalen Geschäftsablauf gesammelt wurden und als vertrauenswürdig angesehen werden.

Diese beiden Informationsquellen müssen von den CA−Betreuern vor der Zertifizierung geeignet verknüpft werden, um die Identität sicher zu überprüfen.

6.7.2.3 Ausstellen des Zertifikats

Zur Ausstellung des Zertifikats wird die Datei mit der Zertifikatanforderung mittels Wechseldatenträger auf das CA−Notebook transportiert, auf welchem der zum Zertifikattyp gehörende private CA−Schlüssel gespeichert ist (Abschnitt 6.2). Nach Signieren der Zertifikatanforderung werden das entstandene Zertifikat und der aktualisierte Zertifikatdatenbank−Index von SSLeay ebenfalls mittels Wechseldatenträger auf den Server mit den öffentlichen Schlüsseldaten übertragen.

6.7.2.4 Übermittlung des ausgestellten Zertifikats

Ist das Zertifikat schließlich ausgestellt, so wird dem Benutzer mit einer signierten E−Mail eine URL mitgeteilt, unter welcher er das Zertifikat laden kann. Die Browser−Software erkennt das Zertifikat als zu einem selbst erzeugten privaten Schlüssel zugehörig und zeigt dem Anwender in einem Dialog das Zertifikat an. Geschieht dies nicht, so kann es sein, daß Schlüsseldaten verloren gegangen sind oder verfälscht wurden. Akzeptiert der Anwender den Ladevorgang, so speichert Netscape das Zertifikat in seiner ebenfalls verschlüsselten Zertifikatdatenbank. Der Netscape−Browser bietet an dieser Stelle auch gleich die Möglichkeit, eine verschlüsselte Kopie des Zertifikates, mit einem Kennwort versehen, getrennt in einer PKCS#12−Datei (Abschnitt 3.5.1) zu sichern.

6.7.2.5 Zukünftige Verbesserungen des Zertifizierungsablaufs

Gemäß den PKIX−Empfehlungen in [Ada98] läßt sich der Zertifizierungsablauf noch an zwei Stellen mit geringem Aufwand verbessern:

1. Vor dem Erstkontakt des Benutzers mit der Zertifizierungsstelle kann dem Benutzer "out−of−band" ein Authentifizierungsmerkmal gegeben werden, welches mit den Identitätsangaben und der Zertifikatanforderung an die Zertifizierungsstelle übermittelt wird (Abschnitt 6.7.2.1.1). Dies würde die Identitätsprüfung externer Benutzer erheblich erleichtern und ist vergleichsweise einfach zu bewerkstelligen.
2. Außerdem sollte der Benutzer nach erfolgreicher Installation seines Zertifikats (Abschnitt 6.7.2.4) dieses Authentifizierungsmerkmal wieder an die Zertifizierungsstelle übertragen (z.B. als verschlüsselte und signierte Mail oder in einem HTML−Formular unter Benutzung des Client−Zertifikats), damit die Zertifizierungsstelle eine Quittung über den erfolgreichen Abschluß des Zertifizierungsablaufs bekommt.

6.7.3 Zertifikatbereitstellung

In diesem Abschnitt sollen die konkret verwendeten Mechanismen der Zertifikatbereitstellung an die PKI−Benutzer beschrieben werden. Es bieten sich mehrere Internet−Dienste dafür an:

- E−Mail
  Man kann einen automatischen Mail−Beantworter schreiben, welcher basierend auf einem definierten Abfragebefehlssatz Zertifikate aus der SSLeay−Zertifikatdatenbank als signierte MIME−Mail an Benutzer schicken kann. Dies hat Vorteile für den Offline−Betrieb, also für Benutzer, die nur temporären Zugang zum Internet haben (über Wählleitungen). Eine CA kann auch ohne Initiative des Benutzers aktuelle Zertifikatdaten als E−Mail an alle CA−Benutzer verschicken. Dies ist vor allem für die rasche Verbreitung von Zertifikatwiderruflisten interessant.

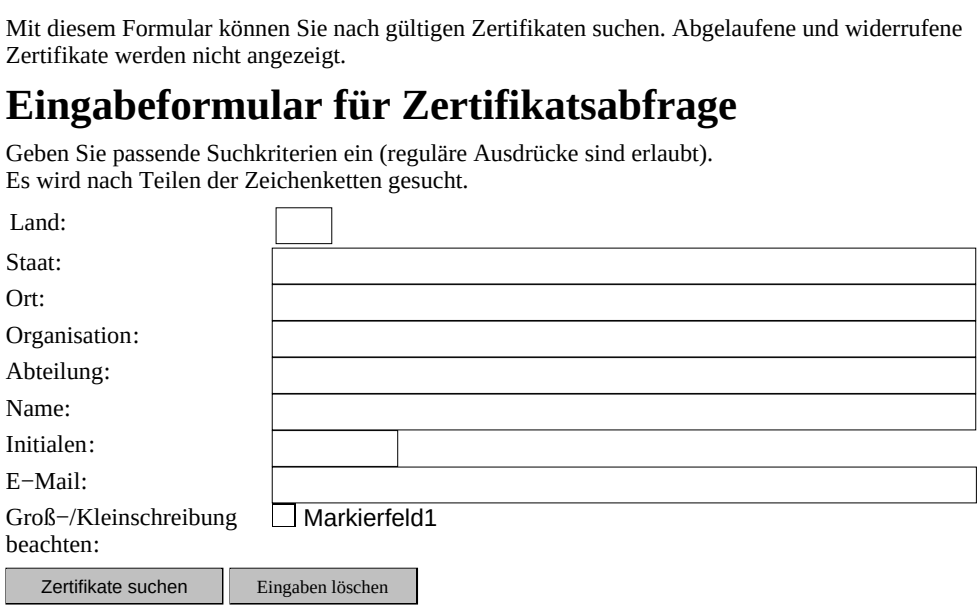
- WWW
  Basierend auf den Netscape−Spezifikationen für die MIME−Typen und Kodierung von Zertifikaten [Nets97e] wurde ein CGI−BIN−Programm für die Abfrage der SSLeay−Zertifikatdatenbank und das Herunterladen von Zertifikaten geschrieben. Dies ist für den Benutzer mit Abstand die einfachste Möglichkeit, er muß nur URL−Adressen anwählen.
- LDAP
  Favorisiert wird von Netscape und PKIX die Bereitstellung über Verzeichnisdienste, speziell über LDAP−Server. Sowohl Netscape als auch PKIX sehen hierfür spezielle LDAP−Schemata vor, welche leider unterschiedlich sind. Verfügt der Benutzer nicht über einen LDAP−Client, so kann auch ein WWW−LDAP−Gateway zum Einsatz kommen (Abschnitt 7.4.1).

6.7.3.1 Bereitstellung von Zertifikaten via WWW

Im Rahmen der Arbeit wurden CGI−BIN−Programme entwickelt, welche den direkten WWW−Zugriff auf Zertifikatdaten der SSLeay−Software ermöglichen. Diese direkte Abfragemöglichkeit aus der SSLeay−Zertifikatdatenbank und den in der SSLeay−Konfigurationsdatei angegebenen Dateien ist von der Verwendung der SSLeay−Software als Zertifizierungsstellen−Software abhängig. Dies ist allerdings nur als Interimslösung gedacht, bis der ebenfalls aufgesetzte LDAP−Server alle Einträge mit den korrekten LDAP−Schemata aufnimmt. Der WWW−Zugriff auf die Zertifikatdaten soll dann ausschließlich über das CGI−BIN−Programm *ldap−client−cgi.py* erfolgen (siehe auch Abschnitt 7.4.1).

6.7.3.1.1 Benutzerzertifikate via WWW

Zur Bereitstellung von Benutzerzertifikaten wurden die beiden CGI−BIN−Programme *cert−query.py* und *get−cert.py* entwickelt. Das Programm *cert−query.py* ermöglicht die Schlüsselwortsuche nach Zertifikaten in der SSLeay−Zertifikatdatenbank und *get−cert.py* liefert ein Benutzerzertifikat mit dem passenden MIME−Typ *application/x−x509−email−cert.*
Zuerst kann der Benutzer in einem HTML−Formular reguläre Ausdrücke als Suchkriterien für das gewünschte Zertifikat eingeben (Abbildung 22).


Mit diesem Formular können Sie nach gültigen Zertifikaten suchen. Abgelaufene und widerrufene Zertifikate werden nicht angezeigt.

Eingabeformular für Zertifikatsabfrage

Geben Sie passende Suchkriterien ein (reguläre Ausdrücke sind erlaubt).
Es wird nach Teilen der Zeichenketten gesucht.

Land:
Staat:
Ort:
Organisation:
Abteilung:
Name:
Initialen:
E−Mail:
Groß−/Kleinschreibung beachten: Markierfeld1

Zertifikate suchen | Eingaben löschen

Abbildung 22: Eingabeformular zur Suche in der Zertifikatdatenbank

Als Abfrageergebnis liefert das CGI−BIN−Programm *cert−query.py* eine Tabelle mit den passenden Einträgen der Zertifikat−DB und jeweils einem URL−Hyperlink für den Zertifikatabruf mittels *get−cert.py*. Wählt man eine der angebotenen URL−Adressen (die Seriennr. des Zertifikats) an, so lädt man mittels des CGI−BIN−Programms *get−cert.py* das DER−kodierte Benutzerzertifikat. Der Netscape−Browser erkennt den MIME−Typ *application/x−x509−email−cert* und fordert den Benutzer mit einem Dialog auf, zu entscheiden, ob er das Zertifikat laden will. Abbildung 23 illustriert den gesamten Ablauf der Abfrage und des Ladens.

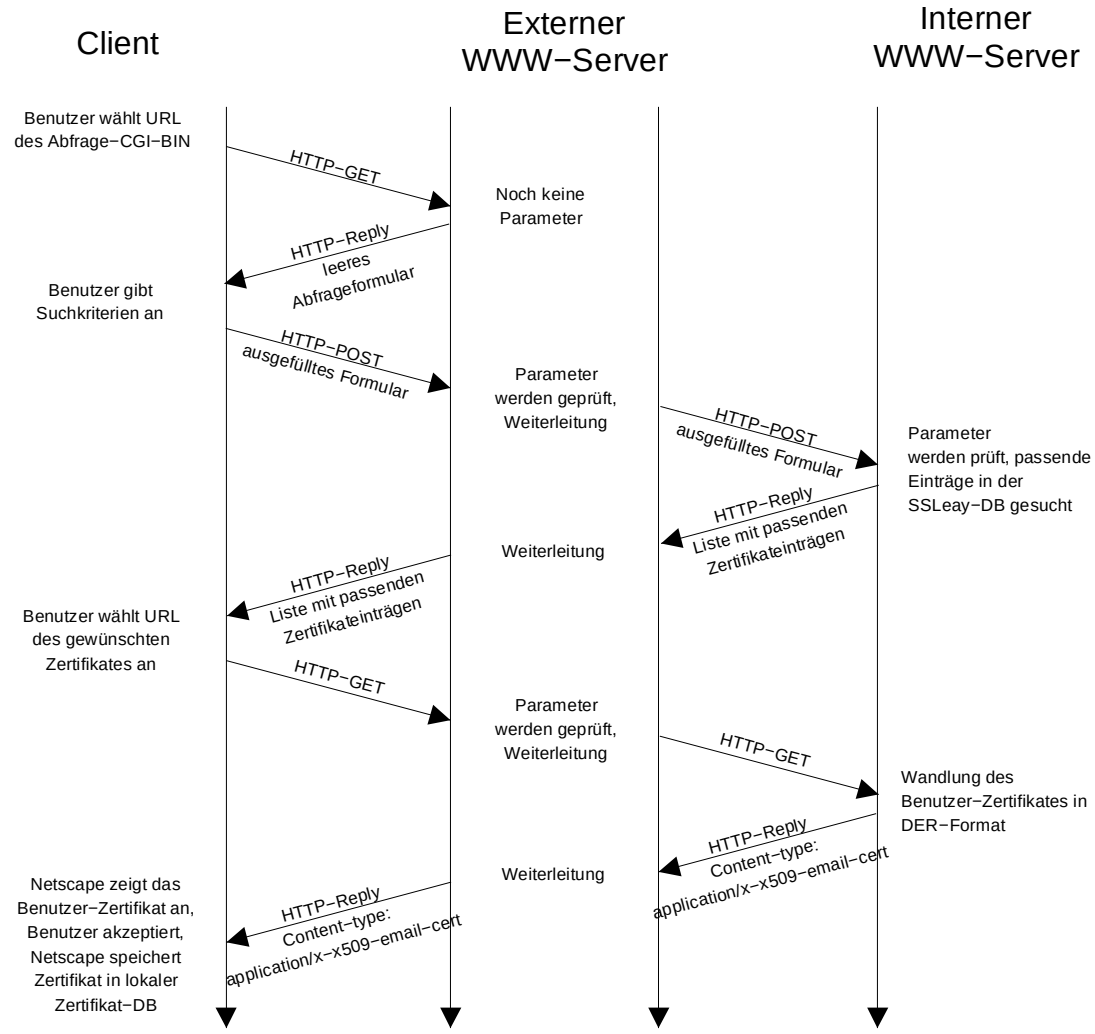


Abbildung 23: Abfrage der SSLeay−DB und Laden eines Benutzerzertifikates

6.7.3.1.2 Zertifikatwiderruflisten via WWW

Mit dem CGI−BIN−Programm *get−crl.py* kann ein Benutzer die aktuellste Zertifikatwiderrufliste einer CA bzw. Sub−CA laden. Den Ablauf des Ladens einer CRL zeigt Abbildung 24.

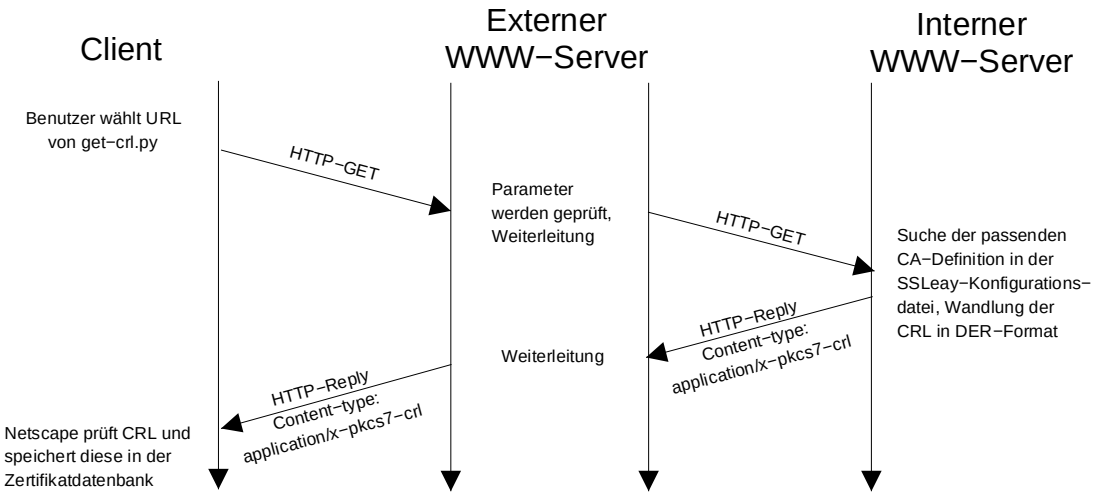


Abbildung 24: Laden einer CRL über WWW

Als Parameter (in der Umgebungsvariable PATH_INFO) wird der Name der CA in der SSLeay−Konfigurationsdatei angegeben. Das CGI−BIN−Programm sendet als Ergebnis die DER−kodierte CRL mit dem MIME−Typ *application/x−pkcs7−crl*. Der Netscape−Client prüft die Signatur der CRL (im Prinzip die Überprüfung eines Zertifikates) und speichert die CRL mit ihrer URLin der Zertifikatdatenbank ab.

6.7.3.2 Bereitstellung von Zertifikaten via LDAP

Bei allen Entwürfen für eine PKI wird die Bereitstellung aller Zertifikate und Zertifikatwiderruflisten über Verzeichnisdienste favorisiert. Besondere Bedeutung hat dabei in der letzten Zeit das X.500−Derivat LDAP durch die Unterstützung aller großen Software−Hersteller erhalten. Netscape Inc. und die PKIX−Arbeitsgruppe haben jeweils Spezifikationen der LDAP−Attribute für Zertifikatdaten definiert, wobei z. Zt. nur die Netscape−Spezifikation von praktischer Bedeutung ist, da sie auf den LDAPv3−Spezifikationen in [RFC2256] aufsetzt und im Netscape Communicator implementiert ist, wogegen PKIX derzeit nur den Status eines Internet−Draft hat [PKIX]. Der Netscape Communicator enthält einen inzwischen brauchbaren LDAP−Client und spezielle Menüpunkte zum Laden der Zertifikatdaten von und zu LDAP−Servern.

Für Zertifikatdaten sind folgende Attributnamen vorgesehen [RFC2256][Nets97i]:

- *usersmimecertificate* bzw. *usersmimecertificate;binary* (OID)
  nimmt eine vom Zertifikatbesitzer selbst signierte PKCS#7−Nachricht der Länge 0 auf, welche auch alle CA−Zertifikate bis zur Root−CA enthält. Die CA selbst kann in der Regel also das Attribut *usersmimecertificate* nicht auf einem LDAP−Server erzeugen, es ist dazu der private Schlüssel des zertifizierten Objekts nötig.
- *usercertificate* bzw. *usercertificate;binary* (OID 2.5.4.36)
  enthält nur das Benutzerzertifikat, kann also von der CA auf einem LDAP−Server erzeugt werden.
- *cACertificate bzw. cACertificate;binary* (OID 2.5.4.37)
  enthält das (im Falle einer Root−CA selbstsignierte) Zertifikat der Zertifizierungsstelle.
- *authorityRevocationList* bzw. *authorityRevocationList;binary* (OID 2.5.4.38)
  enthält die CRL von widerrufenen CA−Zertifikaten.

- *CertificateRevocationList* bzw. *certificateRevocationList;binary* (OID OID 2.5.4.39)
  enthält die CRL der von einer CA widerrufenen Benutzerzertifikate.
- *crossCertificatePair bzw. crossCertificatePair;binary* (OID 2.5.4.40)
  enthält Cross−Zertifikate für fremde Zertifizierungsstellen.
- *supportedAlgorithms* (OID 2.5.4.52)
- *deltaRevocationList;binary* (OID 2.5.4.53)
  enthält die Delta−CRL der von einer CA widerrufenen Benutzerzertifikate.
- *memberCertificateDescription* (OID 2.16.840.1.113730.3.1.199)

Dabei sind die Attribute mit Suffix ';binary' für DER−kodierte Zertifikate gedacht, wogegen die Attribute ohne Suffix Base64−kodierte Zertifikate aufnehmen. In allen Spezifikationen wird ausdrücklich die Verwendung der Attribute mit Suffix ';binary' angeraten.

Außerdem sind folgende LDAP−Objektklassen für Zertifikatdaten relevant:

- *strongAuthenticationUser* (OID 2.5.6.15)
  beschreibt einen Benutzer, welcher über ein Zertifikat zu authentifizieren ist, und enthält das Attribut *usercertificate(;binary).*
- *certificationAuthority* (OID 2.5.6.16)
  ist die Objektklasse für Einträge, welche eine CA beschreiben, und enthält die Attribute *cACertificate;binary, authorityRevocationList;binary, certificateRevocationList;binary* und *crossCertificatePair;binary*.
- *InetOrgPerson* (OID 2.16.840.1.113730.3.2.2)
  ist ein Personeintrag für "Intranet−Benutzer" und enthält u.a. auch das Attribut *usercertificate(;binary).*
- *groupOfCertificates* (OID 2.16.840.1.113730.3.2.31)
  ist bei Netscape zur GruppenAuthentifizierung gedacht.

Das Attribut *usercertificate* enthält nur das Zertifikat, kann also von der CA auf einem LDAP−Server erzeugt werden. Im Rahmen dieser Arbeit wurde das Python−Skript *certs2ldap.py* entwickelt, welches im Batch−Betrieb versucht, alle Zertifikate bzw. die zu einem Abfragemuster passenden Zertifikate aus einer SSLeay−Datenbank in die Personeneinträge auf einen LDAP−Server mit passender LDAP−Objektklasse einzubringen. Dabei wurden nur die in [RFC2256] definierten Attribute und Objektklassen benutzt, die Netscape−spezifischen Attribute (erkennbar am OID−Prefix 2.16.840.1.113730.3) wurden vermieden.

Die CA−Struktur wurde direkt auf eine entsprechende LDAP−Strukur abgebildet. Die Root−CA und jede Sub−CA haben jeweils ihren eigenen LDAP−Eintrag. Das Python−Skript *ssleayca2ldif.py* erzeugt nach den CA−Angaben in einer SSLeay−Konfigurationsdatei *ssleay.cnf* und den CA−Zertifikatdateien eine LDIF−Datei der CA−Hierarchie, welche die LDAP−Einträge einer oder mehrerer Zertifizierungsstellen enthält und in einen LDAP−Server importiert werden kann.

Die CA−Einträge sollten insbesondere auch Angaben zur out−of−band−Kontaktaufnahme mit der Zertifizierungsstelle enthalten. Als Angaben in einem CA−Eintrag unter LDAP kommen in Frage:

- allgemeiner Name der CA (Attribut *cn*)
- Telefonnummer der zuständigen CA−Betreuer (Attribut *telephonenumber*)

- Fax−Nummer (Attribut *facsimiletelephonenumber*)
- postalische Adresse der CA (Attribut *postaladdress*)
- E−Mail−Adresse der zuständigen CA−Betreuer (Attribut *mail*)
- die CA−Zertifikate selbst (Attribute *cacertificate;binary*)
- die aktuelle von der CA herausgegebene CRL (Attribut c*ertificaterevocationlist;binary*)
- eine URL−Adresse, welche auf eine erläuternde WWW−Seite zeigt (Attribut *labelederi*)

Es sollte bei allen LDAP−Einträgen darauf geachtet werden, daß die Angaben in Attributen des LDAP−Eintrags und der entsprechenden Attribute im Zertifikat−DN übereinstimmen. Um dies zu gewährleisten, erzeugt das Skript *ssleayca2ldif.py* die LDAP−Angaben zur Zertifizierungsstelle aus dem Zertifikat−DN.

6.7.3.3 Abgelaufene Zertifikate

Abgelaufene Zertifikate müssen in der SSLeay−Zertifikatdatenbank als solche markiert werden. Zu diesem Zweck wird über CRON zyklisch einmal pro Stunde das Skript *ca−expire.py* aufgerufen, welches alle abgelaufenen Zertifikate als *expired* markiert.

6.7.3.4 Zertifikatwiderruf

Die verschiedenen Gründe für einen Zertifikatwiderruf wurden bereits in Abschnitt 3.3.6.1 dargelegt. An dieser Stelle geht es um konkrete Verfahren des Zertifikatwiderrufs bei der Zertifizierungsstelle. Der Zertifikatwiderruf kann auf mehrere Arten geschehen:

- Der Zertifikatinhaber widerruft sein eigenes Zertifikat.
- Eine RA widerruft ein Zertifikat eines Zertifikatinhabers.
- Die CA widerruft ein Zertifikat eines Zertifikatinhabers.

6.7.3.4.1 Zertifikatwiderruf durch Zertifikatinhaber

Möchte ein Zertifikatinhaber sein eigenes Zertifikat widerrufen, so kann er dies direkt über den Aufruf des CGI−BIN−Programms *ns−revoke.py* mit der Seriennummer des zu widerrufenden Zertifikates (in hexadezimaler Darstellung) als einzigem Parameter tun. Dies ist die schnellste Möglichkeit des Widerrufs, da der Widerruf vollkommen automatisch durchgeführt wird. Eine Online−Zertifikatprüfung weist das Zertifikat dann sofort als ungültig aus, und das Zertifikat kann nicht mehr von anderen Benutzern abgerufen werden. Der Zertifikatwiderruf wird als gültig anerkannt, falls der Zertifikatinhaber beim Widerruf genau das zu widerrufende Zertifikat vorlegt. Das CGI−BIN−Programm *ns−revoke.py* muß dazu auf einem SSL−fähigen WWW−Server installiert sein, der Benutzerzertifikate beim SSL−Verbindungsaufbau anfordert. Zudem muß der Zertifikatinhaber noch im Besitz seines privaten Schlüssels sein. Ist der private Schlüssel eines Zertifikates kompromittiert, so kann ein Zertifikat auf diesem Weg mißbräuchlich widerrufen werden. In diesem Fall kann aber diese Art des Mißbrauchs dem Zertifikatinhaber nur recht sein, da das kompromittierte Zertifikat danach ungültig ist.
Ist es dem Zertifikatinhaber nicht möglich, sein Zertifikat on−line zu widerrufen, beispielsweise weil der zum Zertifikat gehörige private Schlüssel nicht mehr verfügbar ist, so muß er sich direkt an die Zertifizierungsstelle oder eine der regionalen Autoritäten (RA) wenden, welche die Gültigkeit des Zertifikatwiderrufs überprüft.

6.7.3.4.2 Zertifikatwiderruf durch RA

Erhält eine Regionale Autorität (RA) Kenntnis über die Notwendigkeit eines Zertifikatwiderrufs, so muß sie umgehend die Gültigkeit des Zertifikatwiderrufs selbstständig prüfen. Falls die Gültigkeit eines Widerrufs nicht zweifelsfrei erwiesen oder widerlegt werden kann, so sollte eine RA die Gültigkeit des Zertifikatwiderrufs annehmen. Der Zertifikatwiderruf wird entweder als mit einem RA−Zertifikat signierte E−Mail−Nachricht an die CA übermittelt, oder es wird das CGI−BIN−Programm *ns−revoke.py* aufgerufen, wobei beim WWW−Zugriff durch den RA−Beauftragten ein gültiges RA−Zertifikat vorgelegt werden muß.

6.7.3.5 Online−Zertifikatprüfung

Die ausgegebenen Zertifikate bekommen ein erweitertes Attribut *nsRevocationUrl*, welches eine URL−Adresse zur Online−Überprüfung von Zertifikaten enthält (vergleiche Abschnitt 3.3.8.1.1). Über diese URL−Adresse wird vom Netscape−Client das CGI−BIN−Programm *ns−check−rev.py* mit der Seriennummer des zu überprüfenden Zertifikats aufgerufen (vergleiche Abschnitt 5.1.1), welches als Ergebnis vom MIME−Typ *application/x−netscape−revocation* entweder gültig (Wert 0) oder ungültig (Wert 1) zurückliefert (siehe Abbildung 25).

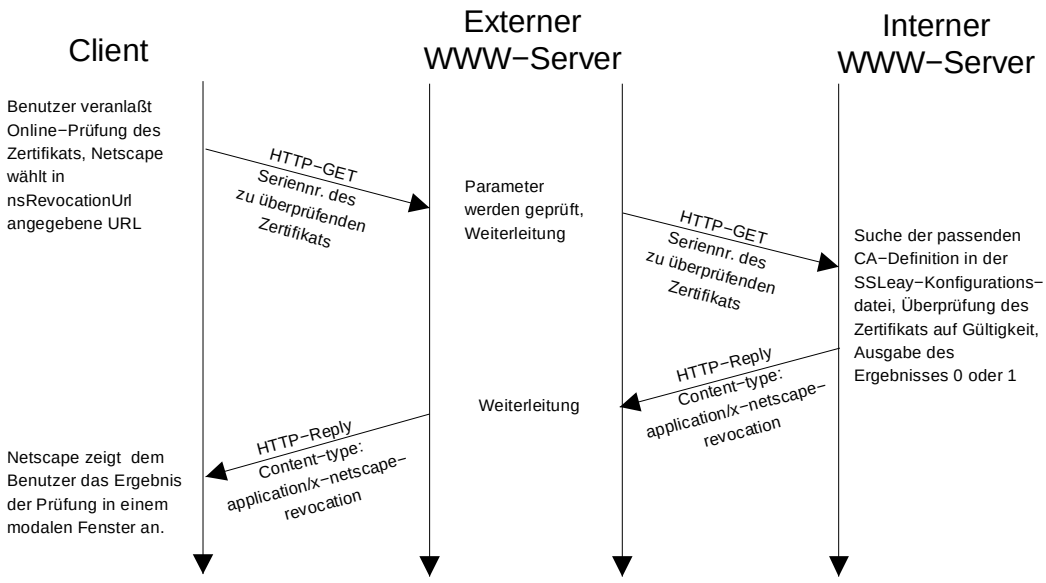


Abbildung 25: Online−Prüfung eines Zertifikats

6.8 Zusammenfassung

Es hat sich gezeigt, daß es möglich ist, mit dem *Netscape Communicator* als Client−Software, *SSLeay* als Zertifizierungsstellensoftware und einigen selbst geschriebenen Programmen zur Integration eine für den Anwender einfach und komfortabel benutzbare PKI aufzubauen, die durchaus, je nach Sorgfalt bei der Betreuung, auch mittleren Sicherheitsanforderungen genügen kann. Bei dem Entwurf wurde darauf geachtet, daß die Funktion der Zertifizierungsstelle sich möglichst nicht in wichtigen Teilen auf herstellerspezifische Spezifikationen stützt, sondern eine möglichst offene Basis für Erweiterungen geschaffen wurde.

Auch traten einige organisatorische Schwierigkeiten, z.B. beim Ablauf der Identitätsprüfung externer Objekte, zutage, welche leider noch nicht einheitlich geregelt werden konnten. Insbesondere die Identitätsprüfung wird im täglichen Zertifizierungsbetrieb den CA−Betreuern ein hohes Maß an Sorgfalt abverlangen. Vor allem ist beim Testbetrieb aber auch die Notwendigkeit zur Schulung der PKI−Teilnehmer als besonders wichtig für einen erfolgreichen Einsatz kryptographischer Techniken erkannt worden.
Der z.Zt. in der Entwicklung befindliche PKIX−Standard (Abschnitt 3.3.8.1.4) gibt auch Empfehlungen für die einzelnen Aufgaben einer Zertifizierungsstelle. Diese Empfehlungen sollten in Zukunft für organisatorische Abläufe übernommen werden, insbesondere für solche Abläufe, die in dieser Arbeit noch nicht geregelt wurden (z.B. Rollover des Root−CA−Zertifikats).

7 Spezielle Beispielanwendungen

In diesem Kapitel werden nach einem kurzen Resümee der durch den Einsatz kryptographischer Techniken gewonnenen Vorteile und verbleibenden Restrisiken, ein paar im Rahmen der Arbeit realisierte Beispielanwendungen dargestellt, welche ein wenig von den üblichen Ende−zu−Ende−Anwendungen abweichen. Dies soll auch ein paar Überlegungen zur sicheren Nutzung von Diensten veranschaulichen, welche selbst keine Verschlüsselung benutzen, sondern in eine verschlüsselte Umgebung eingebettet werden. Desweiteren sollen mit diesem Kapitel Anregungen für weitere Anwendungen gegeben werden.
Zuerst wird ein System zur transparent verschlüsselnden Weiterleitung von E−Mail−Nachrichten beschrieben, als Interimslösung für den Dokumentenaustausch mit externen Mitarbeitern (Abschnitt 7.2). Im Abschnitt 7.3 werden SSL−Wrapper und −Proxies als Lösungen vorgestellt und analysiert, die dazu dienen, nicht SSL−fähige Server−Dienste mittels SSL zu sichern. Abschnitt 7.4 enthält Lösungen und Analysen, wie man unverschlüsselte Server−Dienste über WWW−Gateways anbieten und sichern kann.

7.1 Gewonnene Sicherheitsvorteile

Dieser Abschnitt soll ein kurzes Resümee geben, welche der in Kapitel 2 beschriebenen Sicherheitsprobleme durch den Einsatz kryptographischer Techniken vermieden werden, aber auch welche Restrisiken verbleiben. Es wird dabei angenommen, daß die verwendeten kryptographischen Protokolle und deren Implementierungen keine sicherheitsrelevanten Fehler aufweisen, die verwendete Software korrekt installiert ist und der Benutzer die notwendigen Verfahrensweisen kennt und korrekt anwendet. Als besonders kritisch ist dabei die sichere Handhabung der Schlüsseldaten zu nennen. Alle Vorteile des Einsatzes kryptographischer Techniken beruhen letztlich darauf, daß nur die zertifizierten Objekte die zu Zertifikaten gehörigen privaten Schlüssel benutzen können und die öffentlichen Schlüssel gesichert zum Kommunikationspartner übertragen werden.
Für alle hier aufgeführten Anwendungen gilt, daß weiterhin Denial−Of−Service−Attacken auf die Dienste möglich sind, d.h. die Diensterbringung unmöglich gemacht werden kann. Im Vergleich zu Vertraulichkeits−, Authentizitäts− und Integritätsverlusten ist diese Angriffsart allerdings als vergleichsweise harmlos einzustufen, zumal der Angegriffene diese Art von Attacken leichter bemerken kann.

7.1.1 Routing mit IPSec

IPSec (siehe Abschnitt 3.5.2) bietet mit dem ESP−Protokoll die Verschlüsselung übertragener IP−Pakete an. Damit ist der Inhalt von IP−Paketen (IP Payload) auf der Strecke zwischen den verschlüsselnden IP−Routern gegen Mitlesen geschützt. Eine Verkehrsanalyse ist weiterhin möglich, welche sich durch Verwendung des Tunnel−Modus auf die Analyse des Datenverkehrs zwischen den direkt am öffentlichen Netz angeschlossenen "Firewall"−Systemen beschränken läßt. Natürlich ist zwischen den

verschlüsselnden Routern und den nicht verschlüsselnden End−Systemen die Vertraulichkeit *nicht* gesichert.
Mit dem AH− und ESP−Protokoll ist es möglich, die Authentizität von IP−Sender− und Empfängeradresse zu sichern, was einen wirksamen Schutz gegen IP−Spoofing darstellt und eine notwendige Voraussetzung für den sinnvollen Einsatz von IP−Paketfiltern ist. Zudem können die signierten IP−Pakete auch nicht während der Übertragung verfälscht werden.

7.1.2 Server−Zugriffe via SSL

SSL bietet eine Ende−zu−Ende−Verschlüsselung zwischen den Transportschicht−Endpunkten, womit ein wirksamer Vertraulichkeitsschutz gegen Abhören von Client−/Server−Verbindungen, auch z.B. im LAN, gegeben ist. Die meisten klassischen Sicherheitsprobleme bzgl. Angriffen auf Server bleiben aber bestehen.
Sowohl der Client als auch der Server können sich gegenseitig mittels Client− bzw. Server−Zertifikaten authentifizieren. Damit ist ebenfalls ein Schutz gegen DNS− und IP−Spoofing−Attacken gegeben und eine ernstzunehmende Benutzerauthentifizierung wird möglich. Durch die signierte Datenübertragung ist ferner die Integrität der übertragenen Daten gesichert.

7.1.3 E−Mail mit S/MIME

S/MIME bietet die Verschlüsselung und das Signieren von E−Mail−Nachrichten, was einen wirksamen Schutz zum Erreichen der Vertraulichkeit, Authentizität und Integrität bei diesem Dienst darstellt.
Ein verbleibendes Sicherheitsproblem sind die weiterhin unverschlüsselt übertragenen Kopfdaten der Nachrichten, in welchen neben den Sender− und Empfänger−Adressen meist alle an der Übertragung beteiligten Zwischensysteme (Mail−Relays) mit deren Zeitstempeln, die verwendete Mail−Software etc. enthalten sind. Diese Kopfdaten ermöglichen einem Angreifer weitreichende Verkehrsanalysen und Einblicke in die interne Netzstruktur einer Organisation.

7.2 Mail−Weiterleitung

Eine sehr einfache Anwendung ist, intern unverschlüsselt verschickte Nachrichten in verschlüsselter Form an externe Mail−Adressen weiterzuleiten. Dies ist z.B. in folgenden Fällen sinnvoll:

- Der interne Sender verfügt noch nicht über die erforderliche Ausstattung und ausreichende Kenntnisse, um dem externen Empfänger korrekt verschlüsselte Nachrichten zu senden.
- Ein Mitarbeiter hat eine externe E−Mail−Adresse und möchte alle − auch interne − Nachrichten dorthin weiterleiten, weil er Teile seiner Tätigkeiten zuhause erledigt. Für interne Sender ist diese Weiterleitung nicht sichtbar − vertrauliche Nachrichten würden ohne verschlüsselnde Weiterleitung unkontrolliert nach außen gesendet.
- Manche Mitarbeiter sind sehr mobil und haben deshalb einen Zugang über einen externen Provider, welcher weltweit einheitliche Zugänge anbietet (Roaming). Da es von außen keinen Zugang auf interne Mail−Server gibt, ist eine Weiterleitung die einzige Möglichkeit, bei langer Abwesenheit trotzdem interne Nachrichten zu empfangen.

Abbildung 26 stellt eine schematische Übersicht einer solchen Weiterleitung dar. Ein interner Benutzer sendet eine unverschlüsselte Mail an eine interne Mail−Adresse. Der interne MTA erkennt als Alias für

diese Adresse den Verschlüsselungsprozeß, welcher als Parameter die externe Mail−Adresse erhält und die Nachricht mit dem öffentlichen Schlüssel dieser externen Mail−Adresse verschlüsselt. Die verschlüsselte Nachricht wird wieder an den MTA übergeben, welcher sie dann zum externen Empfänger sendet.

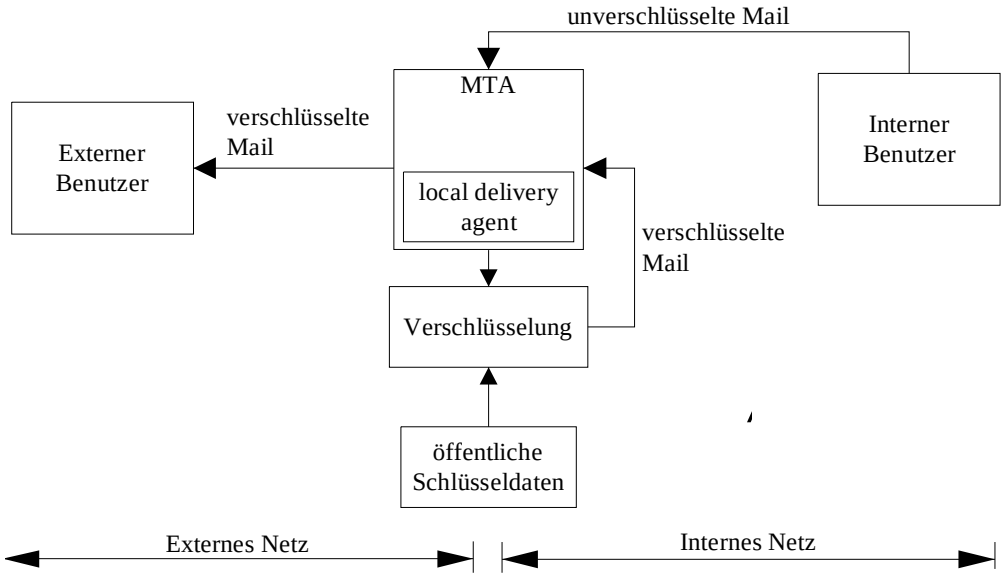


Abbildung 26: Schematische Darstellung der transparent verschlüsselnden Mail−Weiterleitung

Da der Mail−Server unter Linux mit *sendmail/procmail* läuft, ist es kein Problem, eine entsprechende Weiterleitung einzurichten. In der Regel richtet man dazu entweder ein Skript als Mail−Alias ein (meist in der Datei */etc/aliases*), welches die Nachricht verschlüsselt, oder der Empfänger konfiguriert das lokale Ausliefern der Mail entsprechend (Datei *$HOME/.procmailrc*; siehe *man procmailrc*).

7.2.1 Weiterleitung mit PGP

Zur Weiterleitung kann man PGP verwenden, falls dies auf Empfängerseite bevorzugt wird. PGP selbst bietet schon einen Filter−Modus an, bei welchem der unverschlüsselte Quelltext auf Standardeingabe (stdin) erwartet und der Chiffretext auf Standardausgabe (stdout) ausgegeben wird. Man muß also nur *pgp* und *sendmail* mit geeigneten Parametern aufrufen und die Nachrichten werden transparent verschlüsselt.
Beispiel in /etc/aliases:

```
internername: "|pgp −fea externername@ms.you | mail −s "(pgp−fwd)$SUBJECT" externername@ms.you"
```

Beispiel in $HOME/.procmailrc:

```
:0 c
* !^From: internername@ms.my
|pgp −fea externername@ms.you | mail −s "(pgp−fwd)$SUBJECT" externername@ms.you
```

Beide Beispiele verschlüsseln an die interne Adresse (internername@ms.my) gerichtete Mails und leiten sie an die angegebene externe Adresse (externername@ms.you) weiter.

7.2.2 Weiterleitung mit S/MIME

Es gibt leider derzeit keine frei verfügbaren Shell−Programme, welche S/MIME−Mails erzeugen und verarbeiten. Es wurden daher die S/MIMEv2−Programme von Dr. Stephen Henson gekauft, welche aber nur auf Anfrage erhältlich sind. Es sollen aber bald Referenz−Implementierungen des kommenden Standards S/MIMEv3 frei erhältlich sein (siehe Ankündigung unter [RSA97]).
Das Programm *p7encr* zum Verschlüsseln einer Mail erzeugt als Ausgabe die rohe S/MIME−Nachricht. Insbesondere muß man dem Programm direkt den Dateinamen des PEM−kodierten Zertifikates nennen. Für den praktischen Einsatz wurde daher für die verschlüsselte Weiterleitung ein Python−Skript *smime−forward.py* entwickelt, welches folgende Funktionalität beinhaltet:

- Mail−Kopfdaten umsetzen
- Verschlüsselung für mehrere Empfänger
- Suche eines Zertifikates in der SSLeay−Zertifikatdatenbank oder in einem LDAP−Verzeichnis.

Das Weiterleitungs−Skript nimmt Mails auf Standardeingabe (stdin) entgegen und gibt die weiterzuleitende Nachricht auf Standardausgabe (stdout) aus. Demnach kann auch *smime−forward.py* leicht zur Weiterleitung mit den Mechanismen des *sendmail/procmail*−Paketes verwendet werden.
Bereits mit S/MIME verschlüsselte Nachrichten − erkennbar an dem Kopfeintrag *'Content−Type: application/x−pkcs7−mime'* − werden unverändert weitergeleitet. Signierte Nachrichten werden in eine verschlüsselte S/MIME−Nachricht eingebettet. Aufgrund der Eigenschaften des S/MIME−Formates bleibt die ursprüngliche Signatur für das eingebettete MIME−Objekt dabei erhalten. Dem Skript wird als Parameter der gewünschte Verschlüsselungsalgorithmus und eine Liste von RFC822−konformen E−Mail−Adressen angegeben. Zu allen angegebenen Mail−Adressen werden die Zertifikate erst in einer evtl. lokal vorhandenen SSLeay−Datenbank gesucht. Falls sie dort nicht gefunden werden, werden die Zertifikate auf einem LDAP−Server gesucht und von dort geladen, falls vorhanden. Die lokale Ablage von benötigten Zertifikaten ist aus Performance−Gründen zu bevorzugen, falls sehr viele Mails an eine bestimmte Adresse weitergeleitet werden.

Konfigurationsbeispiel in /etc/aliases:

```
internername: "|/usr/sbin/smime−forward.py −−des3 externername@ms.you"
```

Konfigurationsbeispiel in $HOME/.procmailrc:

```
:0 c
* !^From: internername@ms.my
|/usr/sbin/smime−forward.py −−des3 externername@ms.you
```

Beide Beispiele verschlüsseln an die interne Adresse (internername@ms.my) gerichtete Mails gemäß S/MIME−Standard und leiten sie an die angegebene externe Adresse (externername@ms.you) weiter. In den o.g. Beispielen wird außerdem noch als symmetrischer Verschlüsselungsalgorithmus Triple−DES vorgegeben.

7.2.3 Verbleibende Sicherheitsprobleme

Auch PGP− und S/MIME−verschlüsselte Mails enthalten immer unverschlüsselte Anteile, welche u.U. vertraulich bleiben sollten. Dies sind vor allem Sender− und Empfängeradresse, aber auch der Betreff (Subject), welcher unverschlüsselt bleibt. Man könnte natürlich bei der Weiterleitung die gesamte Nachricht mit Betreff verschlüsseln. Das entspricht aber nicht der Handhabung in den E−Mail−Clients,

wäre also für den Anwender mit einem großen Komfortverlust verbunden. Auch stellt die im Klartext vorliegende Sender− und Empfängeradresse eine in jedem Fall verbleibende Möglichkeit zur Verkehrskontrolle dar.
Eine verschlüsselnde Weiterleitung bietet auch die Möglichkeit einer kryptanalytischen chosen−plaintext−Attacke auf den Schlüssel des Empfängers. Ein Angreifer könnte beliebige Klartextnachrichten an eine Weiterleitungsadresse schicken und die chiffrierten Nachrichten mitlesen. Derzeit sind aber keine kryptanalytischen Angriffe gegen die verwendeten Algorithmen bekannt, so daß man diese Gefahr gegenüber anderen Gefahren vernachlässigen kann.
Desweiteren ist auch das Benutzerverhalten ein Problem: Ein interner Empfänger verläßt sich u.U. auf die vermeintliche Vertraulichkeit und läßt sich von einem externen Sender Nachrichten schicken. Die Nachrichten erscheinen dem Empfänger verschlüsselt zu sein, haben aber vorher das Internet im Klartext durchquert. Nachrichten werden auch nicht vom Weiterleitungs−MTA signiert, d.h. das Ziel der Verbindlichkeit erreicht man damit ebenfalls nicht.

Klar ist demnach, daß eine solche Lösung nur ein Provisorium für bestimmte Anwendungsfälle sein kann. Man muß als eigentliches Ziel die breite Einführung der Ende−zu−Ende−Verschlüsselung von Nachrichten für jeden Benutzer anstreben.

7.3 SSL−Weiterleitung

Nicht immer kann man die Ende−zu−Ende−Verschlüsselung vom Client zu einem Server−Dienst ermöglichen. Oft ist die Server−Software nicht in der Lage, den Dienst über SSL anzubieten, oder es liegt z.B. eine Firewall auf dem Weg vom Client zum Server, welche einen direkten TCP−Verbindungsaufbau vom Client zum Server verhindert. Deshalb werden hier kurz mehrere Möglichkeiten erörtert, die wenigstens auf kritischen Teilstrecken die Daten mittels SSL−Verbindung gegen unbefugtes Mitlesen sichern.

7.3.1 SSL−Wrapper

Die kryptographischen Protokolle *SSL* (Secure Socket Layer) bzw. *TLS* (Transport Layer Security) stellen oberhalb von TCP eine Sitzungsschicht für verschlüsselte Verbindungen bereit. Im Prinzip können alle Internet−Dienste damit transparent gesichert werden, indem man die Protokolle der Anwendungsschicht (HTTP, POP3, NNTP, IMAP etc.) einfach auf einen SSL−Socket setzt, anstatt direkt auf TCP aufzusetzen (siehe auch Abschnitt 3.5.3). Im Idealfall hat die Client− und Server−Software die Möglichkeit SSL−Verbindungen aufzubauen, bereits fest eingebaut. Dies ist für eine Authentifizierung des Clients gegenüber dem Server−Dienst mittels Benutzerzertifikaten erforderlich.
Andererseits sind viele Server−Programme im Einsatz, für die es noch nicht so leicht einen Ersatz mit SSL−Unterstützung gibt. An dieser Stelle setzen *SSL−Wrapper* an, welche in den Datenstrom zwischen dem TCP−Socket und dem Internet−Dienst eingeklinkt werden (siehe Abbildung 27).
Dies geschieht meist über den *inetd*, einen Zwischen−Server−Dienst, welcher sich direkt auf einen Socket bindet. Der *inetd* lenkt den Eingabe−Datenstrom vom Client auf *stdin* des Server−Prozesses, und der Ausgabe−Datenstrom von *stdout* des Server−Prozesses wird über den TCP−Socket an den Client gesendet [Ches96][Gar91]. SSL−Wrapper schalten sich ebenfalls mittels dieser stdin/stdout−Weiterleitung zwischen den *inetd* und den Server−Prozeß.

Man erreicht damit, daß Verbindungen von einem Client zu einem Server abhörsicher werden. Es können allerdings keine Benutzerzertifikate zur Authentifizierung des Clients benutzt werden, da der SSL−Wrapper−Prozeß Zertifikatdaten nicht an den Server−Dienstprozeß weitergeben kann. Typische Anwendungsbeispiele für den Einsatz eines SSL−Wrappers sind die Mail−Client−Protokolle POP3 und IMAP, bei deren Ablauf Kennwörter von Benutzern im Klartext über das Netz gehen und sich leicht mitlesen lassen.

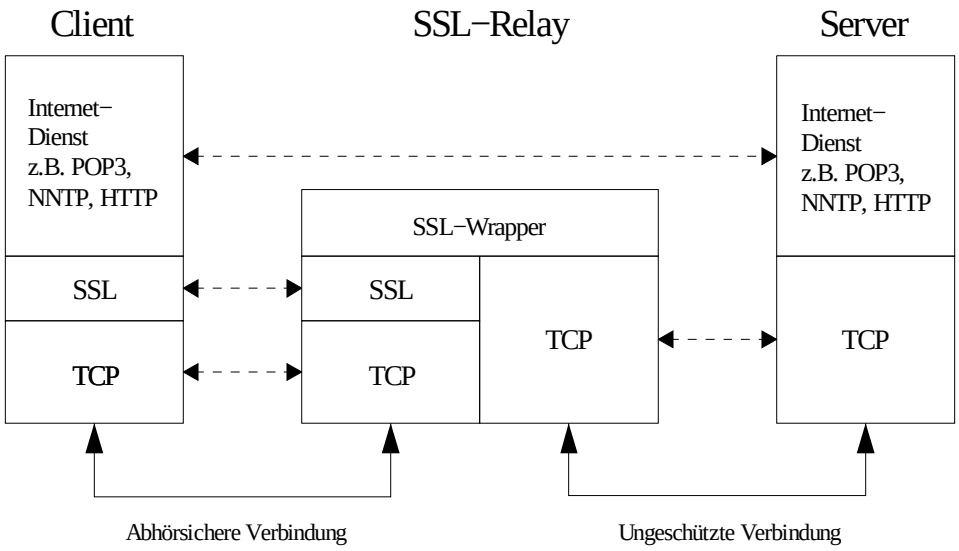


Abbildung 27: SSL−Wrapper im Schichtenmodell

Anzumerken ist noch, daß der SSL−Wrapper−Prozeß und der Server−Dienstprozeß nicht zwingenderweise auf dem selben Rechner laufen müssen. Vielmehr kann eine SSL−Verbindung vom Client zu einem SSL−Relay über eine ungesicherte TCP−Verbindung zum eigentlichen Server−Rechner weitergeleitet werden. Es ist jedoch in den meisten Fällen anzuraten, auf dem Server selbst den SSL−Wrapper−Prozeß zu installieren, damit nicht wieder eine ungesicherte Verbindung über das Netzwerk benötigt wird. Ein Sonderfall liegt vor, wenn man den SSL−Wrapper auf einer speziellen Relay−Station vor ein anderweitig gegen Abhören geschütztes Netzwerk setzt. Dies wäre aus Gründen der Performance und Sicherheit durchaus sinnvoll. Ein dedizierter Rechner mit hoher CPU−Leistung könnte als SSL−Verbindungs−Relay fungieren und böte dabei selbst keine weiteren Internet−Dienste an. Der SSL−Verbindungsaufbau ließe sich dann mit dem SSL−Caching auf dem SSL−Relay, vor allem bei häufigen SSL−Zugriffen von den gleichen Client−Rechnern, ohne großen Aufwand für alle Internet−Dienste enorm beschleunigen.
Zu Testzwecken wurde die SSL−Wrapper−Software *stunnel* in der Version 2.1 in Betrieb genommen, um die POP3− und IMAP−Dienste auf einem internen Mail−Server zu sichern.

7.3.2 SSL−Proxy mit ApacheSSL

Ungesicherte FTP− und WWW−Server−Dienste können auch mit einem *SSL−Proxy* gesichert werden. Dabei wird vom externen Client eine über SSL gesicherte HTTP−Verbindung zu einem WWW−Proxy aufgebaut, und dieser wiederum baut eine FTP− bzw. HTTP−Verbindung zu dem internen Ziel−Server auf (Abbildung 28). Im Gegensatz zu einem SSL−Wrapper (Abschnitt 7.3.1) wird der Datenstrom hier

nicht transparent auf SSL−Ebene weitergeleitet, sondern die Weiterleitung geschieht in der Anwendungsschicht (Gateway−Betrieb).
Für eine solche Konfiguration wurde der WWW−Server ApacheSSL (Abschnitt 5.2.1) eingesetzt, welcher auch über ein Proxy−Modul verfügt. Dieses spezielle Apache−Modul *mod_proxy* ermöglicht es, externe Anfragen auf interne Server weiterzuleiten [Apa98a]. Es wird die *ProxyPass*−Direktive von ApacheSSL verwendet, um Inhalte des Ziel−Servers in ein Unterverzeichnis auf dem Proxy einzubinden. Beispiel:

```
ProxyPass /Projekte/ http://intern.domain.my/Extern/Projekte/
```

Diese Direktive in der Serverkonfiguration des (virtuellen) WWW−Proxy−Servers extern.domain.my:443 veranlaßt den Proxy, SSL−Zugriffe auf die extern zugängliche URL−Adresse https://extern.domain.my:443/Projekte/ auf die nur intern erreichbare URL−Adresse http://intern.domain.my/Projekte/ weiterzuleiten. Diese Methode ist einfach einzurichten und leicht zu durchschauen.

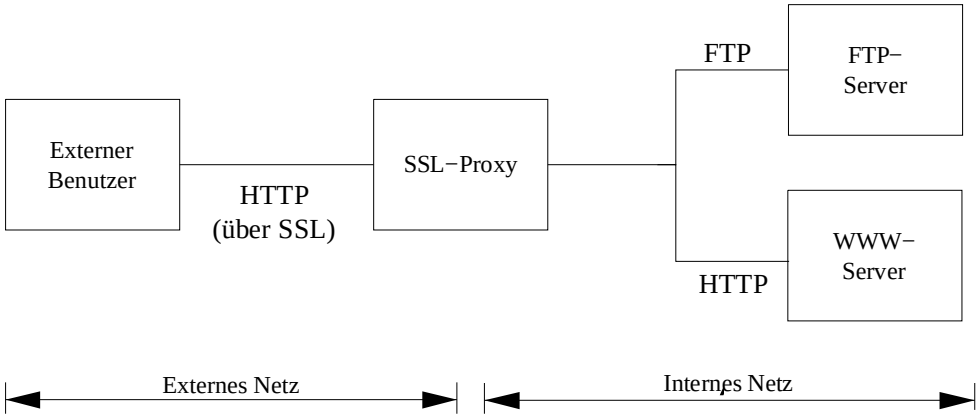


Abbildung 28: SSL−Proxy

Der Schutz gegen unbefugtes Mitlesen bei der Übertragung vom SSL−Proxy zum Benutzer ist bei dieser Methode gegeben, wenn der Server korrekt konfiguriert ist. Die Verbindung vom SSL−Proxy zum Ziel−Server ist aber unverschlüsselt, da das Proxy−Modul von ApacheSSL derzeit nur normale HTTP−Verbindungen zum Ziel−Server aufbauen kann. Das ist keine Ende−zu−Ende−Verschlüsselung, wie sie im Idealfall zu wünschen wäre, ermöglicht aber die SSL−Sicherung von Servern nach außen, welche selbst kein SSL beherrschen oder nur unzureichende Schlüssellängen haben (z.B. Sicherung der externen WWW−Zugriffe auf einen internen Notes−Server).
Ein Problem bei dieser Methode ist die Benutzerauthentifizierung und der darauf basierende autorisierte Zugriff: Zum einen hat man die Authentifizierung des externen Benutzers gegenüber dem SSL−Proxy und zum andern die Authentifizierung des SSL−Proxy−Servers gegenüber den internen Servern. Auf dem SSL−Proxy besteht die Möglichkeit, Benutzer mittels Benutzerzertifikaten und Benutzername/Kennwort zu authentifizieren. Beim Zugriff auf den internen Server ist nur noch die Benutzerauthentifizierung mittels Benutzername/Kennwort möglich, da das Benutzerzertifikat bei dieser Methode nicht zum internen Server gelangt. Man muß also zwei Authentifizierungsmechanismen pflegen, was aber eher umständlich zu nennen ist. Weitere praktische Probleme betreffen Querverweise (Hyperlinks) mit absoluten URL−Adressen des internen Servers, welche vom externen Benutzer nicht verfolgt werden können. Es können nur Verzeichnishierarchien mit ausschließlich relativen Verweisen benutzt werden.

7.4 WWW−Gateways

Manche internen Dienste lassen sich auch über Gateways mittels des Internet−Dienstes WWW nach außen anbieten. Dieses Vorgehen hat verschiedene Vorteile:

- Einfache Client−/Server−Schnittstelle über HTTP und HTML.
- Die Nutzung bereits beim Benutzer vorhandener Software (WWW−Browser) reduziert den Installationsaufwand.
- Die Nutzung bereits gut getesteter Server−Software (z.B. ApacheSSL) vermeidet zusätzliche Sicherheitsprobleme bei den verschiedenen Internet−Diensten.
- Ein WWW−Server ist meist schon in Betrieb und oft ist noch nicht einmal eine Änderung an der Konfiguration nötig.
- Es werden von WWW−Servern detaillierte Zugriffsprotokolle geführt.

Die in Erwägung zu ziehenden Nachteile bei einer WWW−Gateway−Lösung sind:

- Ein CGI−BIN−Programm kann wieder ein zusätzliches Sicherheitsrisiko sein (Abschnitt 2.4).
- Die Performance beim WWW−Zugriff, speziell auf CGI−BINs, ist je nach erbrachtem Dienst und Implementierung oft unzureichend.
- CGI−BIN−Programme sind normalerweise zustandslos, der Zustand in einem Ablauf muß umständlich auf Client−Seite gehalten werden. Die Mittel dazu (versteckte Eingabefelder, Cookies) bergen oft auch wieder ein Sicherheitsrisiko.

7.4.1 LDAP−WWW−Gateway

Die im Rahmen der Arbeit zu Testzwecken in Betrieb genommene LDAP−Server−Software befindet sich derzeit noch in einem unsicheren Entwicklungstand. Insbesondere sind noch keine Kenntnisse über eventuell vorhandene Sicherheitsprobleme vorhanden, die Inbetriebnahme auf einem extern zugänglichen Server wäre ein nicht abschätzbares Sicherheitsrisiko. Dies gilt auch für einen ebenfalls intern vorhandenen Notes−Server, welcher bestimmte Notes−Datenbanken ebenfalls über LDAP zur Verfügung stellen kann. Trotzdem sollte eine sichere Möglichkeit geschaffen werden, von außerhalb auf diese Datenbestände, insbesondere die Zertifikatdaten, zuzugreifen. Es bot sich wieder der Zugriff über ein WWW−Gateway an. Zum Zeitpunkt der Arbeit gab es bereits mehrere LDAP−WWW−Gateways:

- X.500−WWW−Gateway [Ric98]
  Eine sehr ausführliche, für einen globalen Einsatz geeignete Implementierung für den Zugriff auf X.500−Verzeichnisse, aber zu dem Zeitpunkt ohne Schreibzugriff auf die Verzeichnisse. Außerdem ist dieses Gateway ein unabhängiger Server−Prozeß mit direkter Socket−Programmierung, birgt u.U. also ein erhöhtes Sicherheitsrisiko.
- Netscape Directory Gateway
  Im Lieferumfang des Netscape Directory Servers ist auch ein WWW−Gateway enthalten, welches aber mangels freier Verfügbarkeit nicht getestet werden konnte. Außerdem ist sämtliche Netscape Server−Software derzeit nicht unter Linux verfügbar.

- Python Phonebook [LANL98]
  Simple CGI−BIN−Programm−Implementierung in Python, relativ unproblematisch zu installieren, aber ohne Schreibzugriff und recht inflexibel.

Vor allem wurde auch ein Schreibzugriff auf Verzeichnisinhalte benötigt. Daher wurde das CGI−BIN−Programm *ldap−client−cgi.py* entwickelt, welches auch externen Benutzern einen Zugang zu den internen LDAP−Servern ermöglichen soll. Abbildung 29 zeigt eine Übersicht über den Aufbau des Gateways. Ein externer Benutzer greift auf den WWW−Server zu, der das CGI−BIN−Programm *ldap−client−cgi.py* aufruft, welches dann die eigentliche Anfrage an den LDAP−Server absetzt.

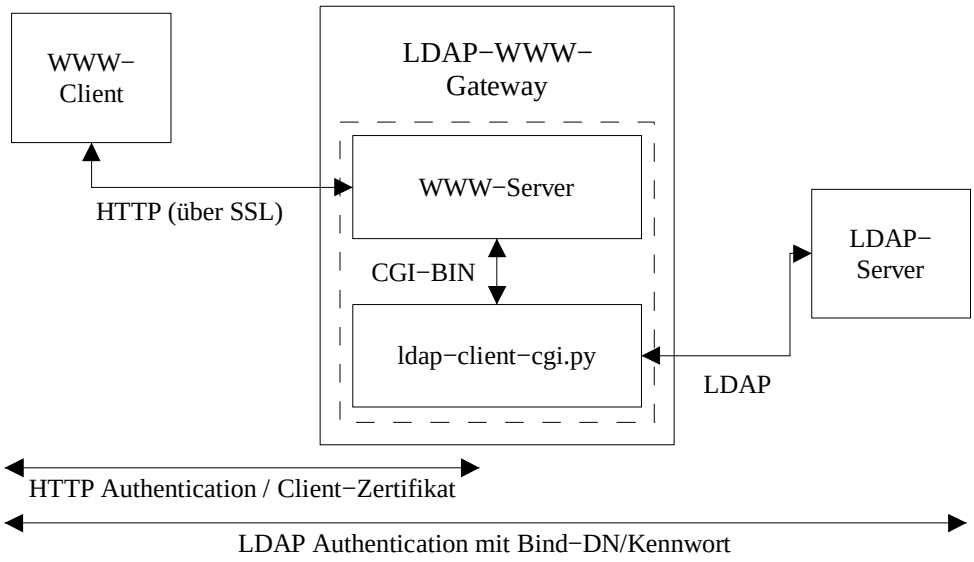


Abbildung 29: Übersicht des LDAP−WWW−Gateways

Die Ziele beim Entwurf waren:

- Erstellung eines vollwertigen LDAP−Clients mit Lese− und Schreibzugriff (add, modify, read, search).
- Leichte Bedienung ähnlich zum LDAP−Client des Netscape Communicator.
- Unabhängigkeit von bestimmten LDAP−Schemata, zumindest eine rohe Ausgabe aller Attribute und Eingabe beliebiger Einträge.
- Ausgabe von Binärdaten mit dem passenden MIME−Typ für HTTP (z.B. für die Zertifikate oder Fotos).
- Möglichst vertrauliche Handhabung aller Daten.
- Autorisierungsmechanismen basierend auf Client−Zertifikaten.
- Leicht konfigurierbare Benutzung von Verschlüsselungsinformationen.
- Flexibilität und Erweiterbarkeit durch Konfiguration.

Das CGI−BIN−Programm *ldap−client−cgi*.py wurde in Python entwickelt und setzt auf dem *ldapmodule* auf, welches ein Interface−Modul für die Bibliotheken des UMich LDAP−Servers ist [Leo98]. In den weiteren Abschnitten sollen vor allem die Sicherheitsaspekte beleuchtet werden. Eine nähere Beschreibung der benötigten Voraussetzungen zur Installation, der Funktionen und Konfigurationsmöglichkeiten von *ldap−client−cgi.py* findet man in [Strö98b].

7.4.1.1 Voraussetzungen

Um überhaupt eine verschlüsselte Sitzung zu erhalten, muß die WWW−Serversoftware SSL beherrschen und entsprechend konfiguriert sein (Beispiel Apache SSL, siehe Abschnitt 5.2.1). Insbesondere muß das CGI−BIN−Programm im richtigen Verzeichnis stehen, falls der WWW−Server eine Konfiguration für mehrere virtuelle Server beherbergt. Das Anfordern von Benutzerzertifikaten durch den Server muß ebenfalls in der Konfiguration des WWW−Servers eingeschaltet werden, das CGI−BIN−Programm hat auf den SSL−Verbindungsaufbau keinen Einfluß.
Das CGI−BIN−Programm *ldap−client−cgi.py* wurde nur unter ApacheSSL getestet, die Umgebungsvariablen mit den Verschlüsselungsdaten der SSL−Verbindung sind derzeit in keiner CGI−BIN−Spezifikation festgelegt. Die verwendeten Umgebungsvariablen sind also spezifisch für den Einsatz mit ApacheSSL (Abschnitt 5.2.1).

7.4.1.2 Sicherheitsstufen (Security level)

Es wurden verschiedene Sicherheitsstufen für den Zugriff auf das CGI−BIN−Programm definiert, um die Konfiguration einfacher zu gestalten. Die jeweilige Sicherheitsstufe wird bei jedem HTTP−Zugriff aus den von ApacheSSL mitgegebenen Umgebungsvariablen und der Skript−Konfiguration neu ermittelt.
Zu beachten ist, daß es bei der Sicherheitsstufe erst einmal nur um die Verbindung zwischen WWW−Client und WWW−Server geht, also nicht um die Authentifizierung und Autorisierung beim eigentlichen LDAP−Zugriff. Dies ist Sache der LDAP−Server−Konfiguration. Die Authentifizierung des CGI−BIN−Programms gegenüber einem LDAP−Server geschieht derzeit ausschließlich basierend auf einem Bind−DN (LDAP−Benutzerkennung) und einem Kennwort, welche der Benutzer über Eingabefelder angeben muß. Sind LDAP−Kennwörter erforderlich, so sollte also immer eine SSL−Verbindung benutzt werden. Zwischen WWW−Server und LDAP−Server wird eine sichere (interne oder verschlüsselte) Verbindung vorausgesetzt.

Zur Konfiguration des CGI−BIN−Programms können am Anfang des Quelltextes mehrere für die Sicherheitsstufe relevante Variablen gesetzt werden:

sec_sslminkeysize

Diese Variable gibt die erforderliche Mindestlänge für den symmetrischen SSL−Sitzungsschlüssel vor. SSL−Verbindungen, deren symmetrischer Sitzungsschlüssel kürzer als diese Mindestlänge ist, werden als nicht verschlüsselt angesehen. Hier sollte also eine Schlüssellänge eingetragen werden, die nach dem momentanen Stand der Technik als nur mit unvertretbar hohem Aufwand brechbar gilt.

sec_valid_dn* und *sec_valid_idn

sec_valid_dn ist ein regulärer Ausdruck für gültige DN zertifizierter Objekte in Benutzerzertifikaten (Client certificate), und *sec_valid_idn* ist ein regulärer Ausdruck für gültige DN von Zertifikatherausgebern. Dies ermöglicht eine Einschränkung der CGI−BIN−Programm−Benutzung auf bestimmte Zertifikate, ohne in die WWW−Server−Konfiguration einzugreifen. Natürlich kann man in der Konfiguration der WWW−Server auch die Autorisierung bis auf die URL genau konfigurieren. Meist wird dies aber schlicht und ergreifend vergessen, weswegen hier ein zusätzlicher Mechanismus eingebaut wurde.

In Abhängigkeit von den o.g. Konfigurationsvariablen wird die Sicherheitsstufe wie folgt ermittelt:

Sicher−heitsstufe	***Beschreibung***	***Vertrau−lichkeit***	***Sichere Authenti−kation***
0	Keine Verschlüsselung oder die symmetrische Schlüssellänge der SSL−Verbindung war kleiner als die erforderliche Mindestschlüssellänge in *sec_sslminkeysize.*	nein	nein
1	Die HTTP−Sitzung ist verschlüsselt über SSL und die Schlüssellänge ist größer oder gleich der konfigurierten Mindestlänge für die verwendeten symmetrische Schlüssel in *sec_sslminkeysize.*	ja	nein
2	Wie Sicherheitsstufe 1, jedoch hat zusätzlich die Client−Software ein gültiges Zertifikat gesendet, der DN des zertifizierten Objektes entsprach dem regulären Ausdruck in *sec_valid_dn*, und der DN des Zertifikatherausgebers entsprach dem regulären Ausdruck in *sec_valid_idn*.	ja	ja

Tabelle 3: Sicherheitsstufen in ldap−client−cgi.py

Die jeweilige Sicherheitsstufe, die aktuellen SSL−Verbindungsdaten und das vom Benutzer verwendete Zertifikat können vom CGI−BIN−Programm mit dem Kommando *secinfo* dem Benutzer angezeigt werden (z.B. wie in Abbildung 30). In diesem Beispiel wurde eine SSL−Verbindung mit RC4 als symmetrischem Algorithmus mit Schlüsselänge 128 Bit und dem Hash−Algorithmus MD5 ausgehandelt, und der WWW−Browser hat ein Benutzerzertifikat vorgelegt, welches den (in diesem Beispiel schwachen) Vorgaben entsprach. Dies ergibt die angezeigte Sicherheitsstufe 2.

CGI LDAP Client − Security info

Current security level is: **2**

You connected with cipher **RC4−MD5**, key size **128 Bit**, secret key size **128 Bit**.

Your client sent the following certificate:

C=DE ST=Baden−Württemberg L=Karlsruhe O=Universität Karlsruhe OU=Informatik CN=Michael Ströder Email=michael@ms.my	C=DE L=Karlsruhe O=Propack Data GmbH OU=Security CN=TestCA Email=ca−admin@propack−data.de

0	no encryption at all
1	Session is encrypted with SSL and keysize of actual symmetric cipher >= 40 Bit
2	Client presented valid certificate, the DN of the certified object matches " `/C=[a−zA−Z][a−zA−Z]/.*`" and the DN of the certifier matches " `/C=[a−zA−Z][a−zA−Z]/.*`"

Abbildung 30: Anzeige der SSL−Verbindungsdaten in ldap−client−cgi.py

Für jedes Kommando wird mit *seq_reqlevel* (Python−Dictionary) eine mindestens erforderliche Sicherheitsstufe für jedes Kommando vorgeben. Beispiel:

```
sec_reqlevel = {
                 'add':2,
                 'delete':2,
                 'modify':2,
                 'passwd':2,
                 'read':1,
                 'search':1,
                 'help':0,
                 'secinfo':0,
                 'searchform':0,
                 'input':2
               }
```

In diesem Beispiel erfordern alle Modifikationskommandos (add, delete, modify, passwd) die Vorlage eines gültigen und akzeptierten Benutzerzertifikates, während Lesevorgänge (search, read) nur gegen Mitlesen gesichert sein müssen und informelle Ausgaben (help, searchform, secinfo) in jedem Fall ausgegeben werden.

7.4.1.3 Weitere Sicherheitsoptionen

Zusätzlich kann man in der Python−Variable *ldap_clientdn2binddn* ein Muster (Python−Stringformatierung) vorgeben, wie der Objekt−DN in einem Benutzerzertifikat in einen Bind−DN für den LDAP−Zugriff umgewandelt wird. Dies dient vor allem der Benutzerfreundlichkeit, da die Zeichenkette eines LDAP−Bind−DN vergleichsweise lang ist. Der so gebildete Bind−DN wird dann als Vorgabewert in das Eingabefeld für den Bind−DN geschrieben. Beispiel:

```
ldap_clientdn2binddn ='cn=%s,ou=%s,o=%s'
```

und ein Zertifikat−DN (T.61−kodiert)

```
/C=DE/L=Karlsruhe/O=Universität Karlsruhe/OU=Informatik /CN=Michael
Ströder/Email=s_stroed@ira.uka.de
```

würde als Bind−DN für den LDAP−Zugriff (UTF−8−kodiert)

```
cn=Michael Ströder,ou=Informatik,o=Universität Karlsruhe
```

ergeben. Dies liefert aber nur bei sorgfältig abgestimmter Zertifikat−Namensvergabe und LDAP−Konfiguration sinnvolle Ergebnisse. Ist *ldap_clientdn2binddn* leer, so wird dieses Verhalten abgeschaltet.

Desweiteren legt die Python−Variable *sec_expire* fest, wie viele Sekunden eine ausgegebene HTML−Seite gültig ist. Dies soll vermeiden, daß vertrauliche Daten über lange Zeiträume in Zwischenspeichern auf Proxy−Servern und bei WWW−Browsern abgelegt werden. Es wird im HTTP−Header ein entsprechend errechneter 'expires:'−Eintrag (momentane Zeit plus in *sec_expire* angegebene Zeitdauer in Sekunden im Zulu−Zeitformat) an den WWW−Browser gesendet. Gibt man 0 Sekunden vor, so sendet das CGI−BIN−Programm immer 'pragma: no−cache' im HTTP−Header (siehe auch Abschnitt 2.4.4).
Ein spezielles Sicherheitsproblem bei *ldap−client−cgi.py* sind vor allem die u.U. benötigten LDAP−Kennwörter, welche als versteckte Eingabefelder im HTML−Text untergebracht werden, um dem Benutzer eine immer neue Kennworteingabe zu ersparen. Das CGI−BIN−Programm muß den Zustand beim Client halten. Die Verwendung von Cookies für diesen Zweck schied wegen geringer Akzeptanz bei den meisten Benutzern aus. Das CGI−BIN−Programm *ldap−client−cgi.py* sollte demnach ausschließlich über SSL−gesicherte Verbindungen verwendet werden, falls die Kennworteingabe verwendet wird.

7.4.1.4 Zukünftige Verbesserungen der Sicherheit

Wünschenswert wäre vor allem eine ebenfalls über SSL gesicherte Verbindung vom WWW−Gateway zum LDAP−Server, damit die Übermittlung des Bind−DN und des Kennworts nicht im Klartext geschieht. Eine SSL−Verbindung setzt einen SSL−fähigen LDAP−Server und eine entsprechende SSL−Socket−Bibliothek unter Python voraus.
Eine Ende−zu−Ende−Authentifizierung mittels Client−Zertifikaten ist bei einer solchen Gateway−Lösung nicht möglich, da zur Verwendung der Authentifizierung der private Schlüssel des Anwenders zum Einsatz kommt, das Gateway aber eine neue SSL−Verbindung zum LDAP−Server aufbaut. Eine sichere Ende−zu−Ende−Authentifizierung ermöglicht nur der direkte, über SSL gesicherte LDAP−Zugriff vom Client zum Server.

7.4.2 Datei−Server−WWW−Gateway

Ein weiterer Wunsch der Anwender war ein Zugriff auf Dateibestände auf Datei−Servern (Novell Netware und Windows NT). Solche Datenbestände direkt über den Datei−Servern eigene Mechanismen (NCP[14] und SMB[15]) zugänglich zu machen, birgt, neben technischen Schwierigkeiten (z.B. IPX−Routing, Installation der Client−Software etc.), auch ernste Sicherheitsrisiken. Diese Überlegungen legen wieder die Einrichtung eines Zugangs über ein WWW−Gateway nahe.

Eine Möglichkeit wäre gewesen, alle Datei−Server mit allen Rechten dem WWW−Server zugänglich zu machen und die Autorisierungsschemata zu den einzelnen Verzeichnissen auf dem WWW−Server einzurichten. Der WWW−Server würde letztlich als ein sehr mächtiger Benutzer auf die Datei−Server zugreifen. Dieser Ansatz wurde verworfen, da die vorhandene Rechtevergabe unter Netware sehr umfangreich ist und die Autorisierungsschemata sich nicht so leicht automatisiert aus Netware zum WWW−Server exportieren lassen. Außerdem wäre dann das WWW−Gateway ein sehr lohnender Angriffspunkt geworden, da es auf allen Servern alle Rechte haben müßte. Zudem wäre auch die benutzerbezogene Protokollierung unter Netware nicht mehr möglich gewesen.

Daher wurde das CGI−BIN−Programm *mount.py* entworfen, welches es Anwendern ermöglicht, sich als Benutzer mit der eigenen Benutzerkennung auf einem Datei−Server anzumelden und dort speziell für sie freigegebene Verzeichnisse in das Dokumentenverzeichnis des WWW−Servers einzubinden (mount). Es wirken also letztlich die auf den Datei−Servern eingerichteten Freigabe−Schemata, und es muß auf dem Datei−Server−WWW−Gateway lediglich sichergestellt werden, daß jeder Benutzer des Gateways sein eigenes Mount−Verzeichnis hat und die Benutzer nicht gegenseitig auf die Mount−Verzeichnisse anderer Benutzer zugreifen können. Da es eher sehr wenige Benutzer sind, welche von außerhalb zugreifen, ist ein solches Vorgehen letztlich übersichtlicher und damit sicherer zu handhaben als direkte Zugriffe auf die Datei−Server.

7.4.2.1 Voraussetzungen

Das Datei−Server−WWW−Gateway wurde speziell für die firmeninternen Belange zum Einsatz mit ApacheSSL unter Linux entwickelt. Unter Linux sind entsprechende Software−Pakete vorhanden, die es ermöglichen, auf Netware− und Windows NT−Server zuzugreifen. Der Linux−Kernel enthält eine

[14]NCP: Netware Core Protocol, Netware−Protokoll für Datei− und Druckdienste
[15]SMB: Server Message Block, von IBM und Microsoft entwickeltes Protokoll für Datei− und Druckdienste

Implementierung der IPX[16]−und NCP−Protokolle sowie auch die Unterstützung für den Dateisystem−Zugriff über SMB. Die frei verfügbaren Shell−Programme *ncpmount* und *smbmount* binden Netware− bzw. Windows NT−Verzeichnisse in den lokalen Verzeichnisbaum ein und werden vom CGI−BIN−Programm *mount.py* für den eigentlichen Mount−Vorgang benutzt.

7.4.2.2 Sicherheit

Der Benutzer kommt über ein unsicheres Netz an das Gateway, welches selbst die Server über ein logisch oder physikalisch getrenntes Netz erreicht (Abbildung 31). Diese Trennung macht die Konfiguration übersichtlicher, direkte Pakete vom Client können die Datei−Server nicht erreichen.

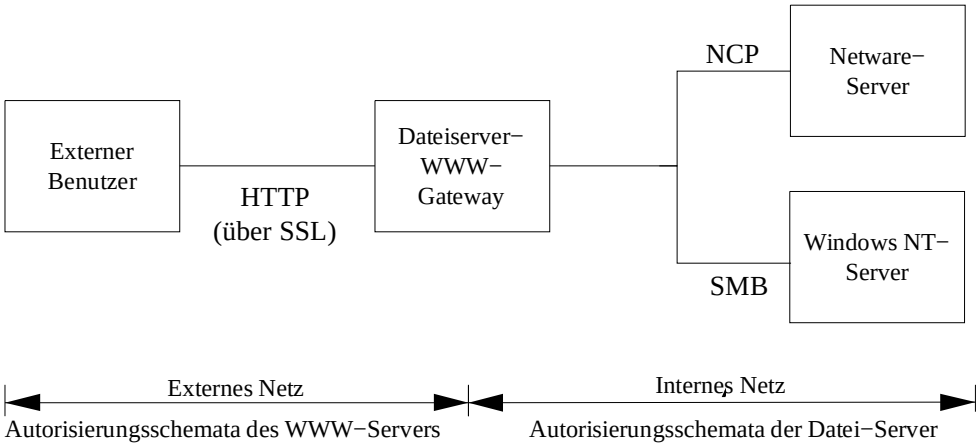


Abbildung 31: Datei−Server−WWW−Gateway

Es sollte auf dem Gateway−Rechner als einziger Dienst möglichst nur der WWW−Server laufen. Für einen Angreifer ist somit der leicht zu überwachende WWW−Server der einzige Angriffspunkt. Die HTTP−Verbindung zwischen Gateway und WWW−Browser wird mit SSL gesichert und kann demnach auch sicher über öffentliche Netze abgewickelt werden. Der eigentliche Zugriff auf die Dateien der Datei−Server geschieht in der derzeitigen Ausführung ausschließlich über die WWW−Server−Software (ApacheSSL) selbst.
Es wurde das Prinzip der konfigurierbaren SSL−Sicherheitsoptionen wie in *ldap−client−cgi.py* (Abschnitt 7.4.1.2) verwendet. Es wird in der Variable *sec_sslminlevel* die mindestens erforderliche Sicherheitsstufe für den Zugriff auf das CGI−BIN−Programm definiert, die Sicherheitsstufe 2 (SSL−Verbindung mit Mindestschlüssellänge und vorhandenem Benutzerzertifikat) wird für den Betrieb dringend empfohlen.
Entsprechend den Mindestanforderungen muß auch der ApacheSSL−Server konfiguriert werden, um zu gewährleisten, daß ausschließlich hinreichend lange Schlüssellängen bei der SSL−Verbindung benutzt werden können (ApacheSSL−Direktiven *SSLBanCipher*, *SSLRequireCipher* oder *SSLRequiredCiphers*). Auch muß jeder auf dem Gateway zugelassene Benutzer mit seinem Benutzerzertifikat−DN auf dem ApacheSSL−Server und einem Mount−Verzeichnis eingerichtet werden.

[16]IPX: Von Novell für Netware entwickeltes Protokoll der Netzwerkschicht

Beispiel einer Konfiguration für einen virtuellen Server mit ApacheSSL:

```
<VirtualHost mount-gw.ms.my:443>

# Verwendung von SSL einschalten
SSLEnable

ServerAdmin www@ms.my
DocumentRoot /usr/local/apache/data.mount-gw
ServerName mount-gw.ms.my
ErrorLog /var/log/mount-gw.error_log
TransferLog /var/log/mount-gw.access_log

# Pfad mit CA-Zertifikaten vertrauenswuerdiger Zertifizierungsstellen
SSLCACertificatePath /usr/local/ssl/TestCA/cacerts

# Server-Zertifikat
SSLCertificateFile /usr/local/apache/conf/pd-server-cert.pem

# Privater Schluessel zu Server-Zertifikat
SSLCertificateKeyFile /usr/local/apache/conf/pd-server-key.pem

# Client-Zertifikat erforderlich!
SSLVerifyClient 2

# Simulation der HTTP-Authentication mit
# Client-Zertifikaten aus dem SSL-Verbindungsaufbau
SSLFakeBasicAuth

ScriptAlias /cgi-bin/ /usr/local/apache/cgi-bin/

<Directory /usr/local/apache/cgi-bin>
AllowOverride None
Options None

AuthUserFile /usr/local/apache/conf/mntpasswd
AuthGroupFile /usr/local/apache/conf/mntgroup
AuthType Basic
AuthName mount-gateway

require valid-user

order deny,allow
allow from 10
deny from all
</Directory>

# Hier kommen die mount-Verzeichnisse

# Das mount-Verzeichnis ist nur fuer Admins zu lesen, damit
# niemand die anderen angemeldeten Benutzer sehen kann.
<Directory /usr/local/apache/data.mount-gw
AllowOverride None
Options Indexes

AuthUserFile /usr/local/apache/conf/mntpasswd
AuthGroupFile /usr/local/apache/conf/mntgroup
AuthType Basic
AuthName mount-gateway

require group admin

order deny,allow
allow from 10
deny from all
</Directory>

# Hier die einzelnen Mountpunkte

# MST

<Directory /usr/local/apache/data.mount-gw/mnt/ncp/mst.server1>
require user "/C=DE/L=Karlsruhe/O=Propack Data GmbH/OU=Information und Kommunikation/CN=Michael
Stroeder/Email=michael.stroeder@ms.my"
</Directory>
<Directory /usr/local/apache/data.mount-gw/mnt/ncp/mst.server2>
require user "/C=DE/L=Karlsruhe/O=Propack Data GmbH/OU=Information und Kommunikation/CN=Michael
Stroeder/Email=michael.stroeder@ms.my"
</Directory>

</VirtualHost>
```

Jedes Mount-Verzeichnis unterhalb der Mount-Wurzel (im o.g. Beispiel /usr/local/apache/data.mount-gw/mnt) folgt einem einheitlichen Namensschema, bestehend aus dem Protokoll (ncp oder smb), der auf allen Servern einheitlichen Benutzerkennung und dem vom Benutzer eingegebenen Servernamen. Ist ein

Mount−Verzeichnis nicht vorhanden, so wird es angelegt. Das Wurzelverzeichnis aller Mount−Verzeichnisse muß dazu für den Benutzer *wwwrun*, unter dem der WWW−Server läuft, schreibbar sein. Ist der Schreibzugriff durch das CGI−BIN−Programm nicht erwünscht, so müssen alle für einen Benutzer erforderlichen Mount−Verzeichnisse beim Einrichten des Benutzers auf dem Gateway manuell angelegt werden, was allerdings die Gefahr einer fehlerhaften Konfiguration birgt.

7.4.2.3 Benutzer einrichten

In diesem Abschnitt folgt noch einmal eine übersichtliche Aufstellung, welche Voraussetzungen erfüllt sein müssen, bevor ein Benutzer das Gateway benutzen kann:

- Der Benutzer muß ein für Server−Zugriffe geeignetes Client−Zertifikat einer internen Zertifizierungsstelle haben.
- Der Benutzer muß auf den Datei−Servern mit seiner Aufgabe entsprechenden Zugriffsrechten eingerichtet sein.
- Der Benutzer muß in eine Benutzerliste (ApacheSSL−Direktive *AuthUserFile*) eingetragen werden. Dies geschieht mit dem kompletten Zertifikat−DN als Benutzername und einem Pseudopasswort.
- U.U. muß der Benutzer mit dem kompletten Zertifikat−DN als Benutzername in eine Gruppe (ApacheSSL−Direktive *AuthGroupFile*) eingetragen werden.
- Für jedes mögliche Mount−Verzeichnis des Benutzers (Verzeichnisname enthält Benutzer− und Server−Name) muß eine *Directory*−Direktive mit Einschränkung der Zugriffsrechte auf exakt den Zertifikat−DN des Benutzers (*require user [Zertifikat−DN]*) eingerichtet werden.

Dieses Verfahren kann für viele Benutzer und Server schnell sehr aufwendig und unübersichtlich werden und ist deshalb nur für wenige Benutzer geeignet. Abschnitt 7.4.2.6.2 skizziert automatisierte Vorgehensweisen.

7.4.2.4 Anmeldung

Die Anmeldung geschieht direkt über das CGI−BIN−Programm, die erforderliche Anmeldeinformation (Benutzerkennung, Kennwort) wird dabei verschlüsselt übertragen, falls die Software entsprechend eingerichtet ist. Die Anmeldemaske für z.B. Netware−Server ist in Abbildung 32 dargestellt.

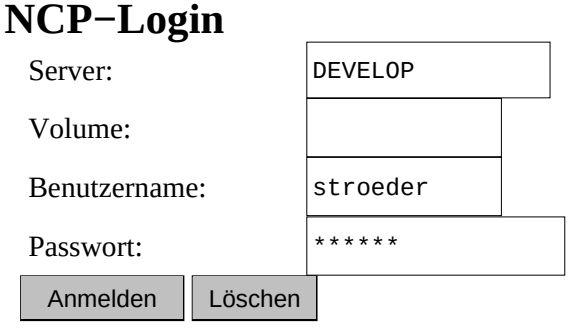


Abbildung 32: Eingabemaske zur Anmeldung an Netware−Server

Das CGI−BIN−Programm zeigt die Ausgabe des jeweiligen Mount−Befehls (*smbmount* oder *ncpmount*) und nach dem erfolgreichem Einbinden des gewünschten Datei−Servers die URL−Adresse des Mount−Verzeichnisses an.

7.4.2.5 Abmeldung

Nach der Anmeldung über das CGI−BIN−Programm *mount.py* hat das Gateway keine Informationen mehr über den Anmeldevorgang und weitere Zugriffe durch den Benutzer. Insbesondere stellt sich die Frage, wie sich der Benutzer wieder abmeldet. Die Möglichkeit, sich darauf zu verlassen, daß der Benutzer manuell Verzeichnisse abmeldet, scheidet als unpraktikabel und unzuverlässig aus. Zuerst einmal wurde die einfachste Methode einer harten Zeitbeschränkung gewählt: Ein regelmäßig (z.B. über CRON[17]) ablaufendes Skript meldet einfach die Benutzer der Mount−Verzeichnisse ab, deren Dateisystem−Zeitstempel älter als eine Stunde ist. Danach wird das Mount−Verzeichnis gelöscht. Diese Methode ist natürlich sehr einfach und unterbricht sogar einen noch laufenden Benutzerzugriff. Da z. Zt. nur sehr wenige Benutzer nur kurzzeitige Zugriffe auf das Datei−Server−WWW−Gateway tätigen, ist dies aber als akzeptabel hinzunehmen. Abschnitt 7.4.2.6.2 skizziert elegantere, aber auch aufwendiger zu realisierende Möglichkeiten.

7.4.2.6 Zukünftige Verbesserungen

Für ein vollständiges Datei−Server−WWW−Gateway wären die nun folgenden Erweiterungen wünschenswert.

7.4.2.6.1 Schreibzugriff mit PUT

Der Schreibzugriff auf Verzeichnisse des WWW−Servers kann über die HTTP−Methode PUT erfolgen. Im speziellen Fall des Apache−Servers ist zur Behandlung der PUT−Requests ein spezielles CGI−BIN−Programm erforderlich [ApWe97.4]. Eine Implementierung als Apache−Modul ist ebenfalls denkbar. Dieses CGI−BIN−Programm muß mit höchster Sorgfalt erstellt werden. Es sind folgende sicherheitsrelevante Aspekte zu beachten:

- Die sichere Authentifizierung der Benutzer muß erfolgen. Dies läßt sich leicht mit dem in einer Umgebungsvariable an das CGI−BIN−Programm übergebenen Zertifikat−DN bewerkstelligen.
- Das CGI−BIN−Programm läuft mit den Zugriffsrechten des WWW−Servers. Dieser muß auf alle Verzeichnisse schreibend zugreifen können, d.h. alle Autorisierungsschemata aus der Server−Konfiguration müssen vom CGI−BIN−Programm unter Benutzung der Authentifizierung beachtet werden.
- Kann ein Benutzer aus dem CGI−BIN−Programm ausbrechen und betriebssystemeigene Programme ausführen (z.B. eine Shell), so kann er auf alle Datei−Server−Inhalte lesend und schreibend zugreifen.

Das Risiko bei einem Schreibzugriff ist noch um einiges höher als beim reinen Lese−Zugriff, so daß man sich die Notwendigkeit dafür sehr genau überlegen muß, bevor man dies in Betrieb nimmt.

7.4.2.6.2 Gateway−Benutzerverwaltung

Zwei Aspekte bei dem bisherigen Vorgehen sind stark verbesserungswürdig:

- Jeder Benutzer muß auf dem Datei−Server−WWW−Gateway eingerichtet werden. Dies bedeutet einen zusätzlichen administrativen Aufwand und ist u.U. eine sicherheitskritische Fehlerquelle. Wünschenswert wäre eine automatische Autorisierung basierend auf den Benutzerzertifikaten.
- Für einen eleganteren Abmeldemechanismus wäre es wünschenswert, daß der Zeitpunkt des letzten Zugriffs eines Benutzers auf sein Mount−Verzeichnis erfaßt würde, um ein zeitgesteuertes Abmelden nach einer bestimmten Zeitspanne der Inaktivität zu ermöglichen.

[17] Ein unter vielen Betriebssystemen verfügbarer Systemdienst, der Programme (meist zyklisch) zeitgesteuert aufruft.

Hierzu könnten die Anmeldung und die Zugriffe auf die jeweiligen Mount−Verzeichnisse über eine persistenten Prozeß geschehen, welcher die Anmeldeinformationen (Benutzerkennung, Benutzerzertifikat, Mount−Verzeichnis) und den Zeitpunkt des letzten Zugriffs speichern kann. Basierend auf diesen Informationen könnte dieser zentrale Prozeß die Autorisierung und Abmeldung automatisch realisieren. Dieser Prozeß könnte in verschiedener Weise ausgeführt sein:

- Ein normales CGI−BIN−Programm, welches die Anmelde− und Zugriffsinformationen in einer Datei speichert (eine Datenbank mit Zertifikat−DN als Index).
- Ein persistentes CGI−BIN−Programm gemäß der Fast−CGI− oder PCGI−Spezifikation [PCGI].
- Ein persistentes Java−Servlet.
- Ein Apache−Modul, welches über spezielle URL−Adressen angesprochen wird [Apa98a].

Die automatische Autorisierung und Abmeldung birgt je nach Implementierung und Konfiguration verschiedene Sicherheitsrisiken, eine sichere Realisierung ist demnach recht aufwendig und würde den Rahmen dieser Arbeit sprengen.

7.5 Bestellsystem

Um ein Bestellsystem über Internet zu realisieren, welches Verbindlichkeit garantieren kann, müssen die Bestellangaben eines Benutzers von diesem selbst signiert an ein Bestellsystem übermittelt werden. Die Übermittlung der Bestelldaten kann z.B. als signierte E−Mail−Nachricht (Abschnitt 7.5.1) oder als signiertes WWW−Formular (Abschnitt 7.5.2) erfolgen. Diese beiden Möglichkeiten wurden im Rahmen dieser Arbeit untersucht.
Eine weitere wichtige Voraussetzung für die Verbindlichkeit ist, z.B. in den allgemeinen Geschäftsbedingungen (AGB) für das Bestellsystem, vertraglich festzuhalten: Private Schlüssel müssen von Teilnehmern geheim gehalten werden. Mit einem privaten Schlüssel signierte Bestellungen werden solange als gültig angesehen, bis ein Widerruf des zugehörigen Benutzerzertifikats erfolgt oder die Gültigkeitsdauer des Zertifikats abgelaufen ist. Dies erlegt dem Benutzer eine gewisse Sorgfaltspflicht auf. Allerdings ist die rechtliche Lage bezüglich der Wirksamkeit solcher Abmachungen derzeit noch unklar.

Wesentlich einfacher und sicherer wird ein Bestellsystem, wenn vor der Online−Bestellung schon eine Geschäftsbeziehung zwischen dem Kunden und dem Lieferanten besteht. Eine vorher vergebene Kundennummer kann dann mithilfe des Benutzerzertifikats auf Plausibilität geprüft werden und die eigentliche Bearbeitung der Bestellung enthält weitere Plausibilitätsprüfungen, welche den Mißbrauch erschweren.

7.5.1 Kombinierter WWW− und Mail−Dialog

Eine einfache Variante der Online−Bestellung ist, den Benutzer eine signierte E−Mail an das Bestellsystem schicken zu lassen, welche die Angaben über gewünschte Bestellungen enthält. Soll die E−Mail−Nachricht automatisch bearbeitet werden, so ist ein leicht von einem Programm zu behandelndes Format erforderlich. Strenge Formatvorgaben für E−Mail−Nachrichten sind aber für Benutzer meist nur schwer einzuhalten. Daher bietet es sich an, den Benutzer erst ein HTML−Formular ausfüllen zu lassen, dessen Formulardaten an ein CGI−BIN−Programm übergeben werden, welches die vom Benutzer gemachten Angaben als geeignet formatierte E−Mail an den Benutzer zurückschickt, mit der

Aufforderung, diese Angaben innerhalb eines relativ kurzen Zeitraums signiert an das Bestellsystem zu schicken (Abbildung 33).

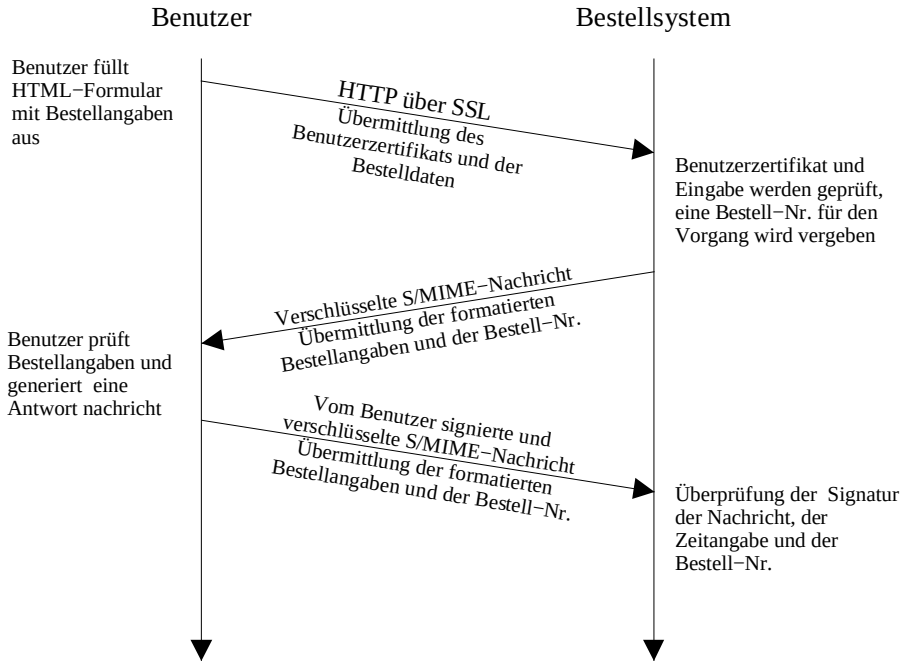


Abbildung 33: Ablauf einer verbindlichen Bestellung beim kombinierten WWW−E−Mail−Dialog

Um die Vertraulichkeit der Bestellangaben zu gewährleisten, muß der gesamte Datenverkehr verschlüsselt abgewickelt werden:

- Die Verbindung von WWW−Browser zum WWW−Server muß über SSL gesichert werden.
- Die formatierte E−Mail vom Bestellsystem zum Benutzer muß mit dem öffentlichen Schlüssel aus dem Benutzerzertifikat verschlüsselt werden.
- Die formatierte E−Mail vom Benutzer zum Bestellsystem muß mit dem öffentlichen Schlüssel aus dem Zertifikat des Bestellsystems verschlüsselt werden.

Um die Verbindlichkeit zu gewährleisten, sind folgende Maßnahmen erforderlich:

- Der Benutzer muß ein gültiges Zertifikat einer anerkannten Zertifizierungsstelle haben.
- Die als E−Mail übermittelte Bestellinformation muß vom Benutzer signiert werden.
- Der Zeitstempel der signierten Bestellinformation muß vom Bestellsystem auf Plausibilität überprüft werden.

Zum Einsatz kamen für die Behandlung der S/MIME−Nachrichten wieder die S/MIME−Programme von Dr. Henson (vergleiche auch Abschnitt 7.2).

7.5.2 Signierte Formulare

Die Client−Software Netscape Communicator der Version 4.04 und höher bietet die Möglichkeit, auf Seite des Clients über einen JavaScript−Aufruf beliebige Textinhalte vom Benutzer mit dessen privaten Schlüssel signieren zu lassen. Dies kann benutzt werden, um HTML−Formulardaten signiert an ein CGI−

BIN−Programm zu übermitteln [Nets98b]. Die signierten Texte und die Signatur müssen dann auf Server−Seite, z.B. von einem CGI−BIN−Programm, überprüft und gespeichert werden. Der Ablauf besteht beispielsweise aus folgenden Schritten:

1. Das Bestellsystem stellt einen HTML−Text mit einem HTML−Formular und einer eigenen JavaScript−Funktion *signForm* bereit.
2. Der Benutzer füllt das HTML−Formular aus.
3. Beim Betätigen des Submit−Buttons wird die im ONSUBMIT−Parameter des <FORM>−Tags angegebene JavaScript−Funktion *signForm* aufgerufen. Die JavaScript−Funktion *signForm* muß im selben HTML−Quelltext bereitgestellt werden.
4. Die JavaScript−Funktion *signForm* baut die Parameternamen/−Inhalt−Paare aller Formularfelder zu einem einzigen Text zusammen und ruft die JavaScript−Methode *crypto.signText* mit geeigneten Parametern auf [Nets98d].
5. Der Benutzer bekommt ein modales Dialogfenster mit dem zu signierenden Text angezeigt. Mit Druck auf den *OK*−Button signiert er diesen Text.
6. Der signierte Text und die Signatur werden den Formulardaten hinzugefügt.
7. Die Formulardaten werden an das CGI−BIN−Programm übermittelt.
8. Das CGI−BIN−Programm prüft die Signatur und, soweit möglich, die Plausibilität der übermittelten Daten.

Um die Vertraulichkeit der Bestellangaben zu gewährleisten, müssen die WWW−Zugriffe verschlüsselt, also über eine SSL−Verbindung, abgewickelt werden.

Anzumerken ist hierbei, daß nicht die eigentlichen Formulardaten signiert werden, sondern von der JavaScript−Funktion dem Benutzer ein beliebiger Text zum Signieren vorgelegt wird. Idealerweise ist dieser Text aus den eingegebenen Formulardaten zusammengebaut, aber es obliegt dem Benutzer, dies sorgfältig zu prüfen. An folgendem Beispiel wird diese Diskrepanz deutlich (verkürzter Quelltext aus [Plum98]):

```
<HTML>
<HEAD>
<TITLE>signText Example</TITLE>

<SCRIPT LANGUAGE="javascript1.2">

function formatElement(name, value) {
  return "<LI>"+"<BOLD>"+name+":</BOLD> "+value+"\n";
}

function signForm(theForm, theWindow, validation) {
  var formSize = theForm.elements.length;
  var text = "I affirm the following information:\n<UL>";
  var elem;

  for( var i = 0; i < formSize; i++) {
    elem = theForm.elements[i];
    switch (elem.type) {

      case "hidden":
      case "button":
      case "submit":
      case "reset":
      case "image":
      case "password":
      case "file":
```

```
        // Don't include these elements in the validation.
        break;

      case "select-one":
        var selectValue = elem.options[elem.selectedIndex].text;
        text += formatElement(elem.name, selectValue);
        break;

      case "select-multiple":
        for(var op = 0; op < elem.length; op++) {
          if(elem.options[op].selected) {
            text += formatElement(elem.name, elem.options[op].text);
          }
        }
        break;

      case "radio":           // Radio buttons and checkboxes can be
      case "checkbox":        // treated the same way.
        if(elem.checked) {
          text += formatElement(elem.name, elem.value);
        }
        break;

      default:
        text += formatElement(elem.name, elem.value);
    }
  }
  text += "</UL>"
  var signedText = theWindow.crypto.signText(text, "ask");
  validation.value = signedText;
  return true;
}

</SCRIPT>

</HEAD>
<FORM ACTION="http://hoohoo.ncsa.uiuc.edu/cgi-bin/test-cgi" METHOD="get"
ONSUBMIT="return signForm(this, window, this.validation);">

Text Input:
<INPUT TYPE="text" NAME="textInput" VALUE="Joe Schmoe">
<INPUT TYPE="submit"><INPUT TYPE="reset">

</FORM>
</BODY>
</HTML>
```

Die JavaScript-Routine *signForm* baut zwar aus den Formulardaten einen für den Benutzer gut lesbar formatierten Text zusammen, welchen der Benutzer signiert. Dieser Text wird aber im o.g. Beispiel überhaupt nicht an das CGI-BIN-Programm übermittet, sondern nur seine Signatur im Parameter *validation*. Der Text ist aber der eigentlich verbindliche Inhalt. In diesem Beispiel müßte das CGI-BIN-Programm diesen Text wieder selbst genauso rekonstruieren, was eine konzeptionelle Schwäche darstellt.

Im Rahmen der Arbeit wurde das prototypische CGI-BIN-Programm *signed-forms-demo.py* erstellt, welches ein leeres Eingabeformular und die JavaScript-Funktion *signForm* ausgibt, wenn es ohne Parameter aufgerufen wird. Im Gegensatz zum obigen Beispiel fügt hier die Funktion *signForm* aber auch den eigentlich zu signierenden Text und die Signatur zu den an das CGI-BIN-Programm übermittelten Formulardaten hinzu (die versteckten Eingabefelder *__pkcs7data__* und *__pkcs7signature__*). Zur Überprüfung der mittels *crypto.signText* signierten Daten stellt Netscape das Programm *signver* zur Verfügung [Nets98e]. Der signierte Text und die Signatur müssen also vom CGI-BIN-Programm gespeichert werden. Danach ruft das CGI-BIN-Programm das Netscape-Programm *signver* mit den entsprechenden Parametern als externes Programm auf [Nets98e]. Ist die Überprüfung erfolgreich, so werden die Formulardaten als durch den Zertifikatbesitzer signiert angesehen. Bei einem Bestellsystem ist die Bestellung dann als verbindlich akzeptiert.

Hier noch einmal zusammenfassend die Nachteile dieser Methode:

- Das hier vorgestellte *Form Signing* basiert auf einer proprietären JavaScript−Methode, d.h. das Verfahren ist nicht für den Einsatz mit WWW−Browsern geeignet, welche entweder kein JavaScript beherrschen oder auch nur die *crypto.signText*−Methode nicht unterstützen. Zudem wiesen die JavaScript−Implementierungen der beiden großen Browser−Hersteller in der Vergangenheit immer wieder massive Sicherheitsprobleme auf, so daß man den Anwendern nur empfehlen konnte, die JavaScript−Funktionalität abzuschalten.
- Die Kombination aus JavaScript−Code und dem die Formulardaten bearbeitenden CGI−BIN−Programm muß den signierten Text und die Signatur konsistent handhaben. Das CGI−BIN−Programm muß den signierten Text als relevante Eingabe annehmen und nicht, wie eigentlich üblich, die Formulardaten der Eingabefelder. Der signierte Text muß also sowohl für den Anwender verständlich lesbar, als auch für das CGI−BIN−Programm einfach zu behandeln sein.

Um die konsistente Handhabung der *Signed Forms* zu erleichtern, könnte die erforderliche Funktionalität in das Modul *cgiforms.py* (Abschnitt 2.4.5) integriert bzw. in einer von der Klasse *formClass* abgeleiteten Klasse *signedformClass* untergebracht werden. Eine Methode *signedformClass.JSsignForm* würde den erforderlichen JavaScript−Code für die bereits deklarierten Eingabefelder speziell erzeugen. Die Methode *signedformClass.GetParams()* würde um eine entsprechende Prüfsequenz erweitert, welche die Gültigkeit der Signatur der übermittelten Formulardaten überprüft. Ist die Signatur gültig, so wird die Übereinstimmung des signierten, benutzerfreundlich formatierten Texts und der eigentlichen Formularparameter überprüft. Tritt bei den Überprüfungen ein Fehler auf, so wird eine Ausnahme der entsprechenden Fehler−Klasse erzeugt. Das CGI−BIN−Programm müßte den signierten Text zwar zu Dokumentationszwecken speichern, könnte aber wie gewohnt direkt mit den von *cgiforms.py* bereitgestellten Formulardaten arbeiten.

7.6 Zusammenfassung

In diesem Kapitel wurden verschiedene Lösungsansätze vorgestellt, die darlegen, wie man kryptographische Techniken zur Sicherung von Internet−Diensten einsetzen kann, ohne daß alle beteiligten Komponenten die relevanten kryptographischen Protokolle unterstützen. Insbesondere sind dabei folgende Aspekte interessant:

- Speziell die Verteilung von Client−Software an viele Benutzer gestaltet sich oft zeitaufwendig. Eine Interimslösung (z.B. Mail−Weiterleitung, siehe Abschnitt 7.2) erreicht nur eine kryptographische Sicherung von Teilstrecken, ist aber in jedem Fall der gänzlich unverschlüsselten Übertragung vorzuziehen.
- Man kann Server−Software gegen Angriffe schützen, wenn man die beteiligte Software als unsicher einstuft und demnach der direkte Client−Zugriff ein zu hohes Risiko darstellen würde (z.B. LDAP−WWW−Gateway, siehe Abschnitt 7.4.1).
- Man kann mit kryptographischen Techniken schon vorhandene, aber unzureichende Sicherheitsmechanismen unterstützen, wie z.B. bei der Verschlüsselung von zu übertragenden Kennwörtern (z.B. Datei−Server−WWW−Gateway, siehe Abschnitt 7.4.2).

Desweiteren wurde mit einem Entwurf für ein Online−Bestellsystem (Abschnitt 7.5) eine weitere Anwendung zum Einsatz kryptographischer Techniken skizziert, wobei die Methode mit signierten

Nachrichten (Abschnitt 7.5.1) für den Benutzer vergleichsweise umständlich ist, während die Verwendung von Signed Forms (Abschnitt 7.5.2) noch konzeptionelle Schwächen hat und eine eher proprietäre Lösung für Netscape−Browser darstellt.

8 Zusammenfassung und Ausblick

Es wurde in dieser Arbeit untersucht, wie sich die Risiken bei der praktischen Nutzung von Internet−Diensten mithilfe von kryptographischen Techniken zur Verschlüsselung und Authentifizierung effektiv verringern lassen.
Eine Analyse der gewünschten Kommunikationsarten ließ den *WWW*−Dienst mit sicherer Authentifizierung von Benutzern und den vertraulichen und verbindlichen *E−Mail*−Austausch als wichtigste Basisanwendungen erkennen. Dies führte zu einer Auswahl und näheren Betrachtung benötigter bzw. bereits verwendeter Software, in deren Verlauf *Netscape Communicator* als Client−Software und *ApacheSSL* als Server−Software ausgewählt wurden. Zudem wurde als Basis−Software zum Aufbau einer Zertifizierungsstelle *SSLeay* ausgesucht.
Zur Benutzung kryptographischer Techniken wurde eine Zertifizierungsstelle aufgesetzt, die Zertifikate für verschiedene Anwendungszwecke ausstellt. Dabei wurden nach der Entscheidung für einen X.509−basierten PKI−Ansatz und die darauf aufbauenden Protokolle SSL für Server−Zugriffe und S/MIME für E−Mail−Nachrichten die für den sicheren Betrieb einer Zertifizierungsstelle notwendigen Verfahrensweisen und Voraussetzungen geschaffen. Zum Entwurf der Zertifizierungsstelle wurde eine für die gewünschten Zertifikattypen geeignete CA−Topologie festgelegt. Der Zertifizierungsablauf, die Zertifikatbereitstellung, die Zertifikatprüfung und der Zertifikatwiderruf wurden konzipiert bzw. realisiert. Im Testbetrieb wurden die Zertifizierungsstelle und die organisatorischen Abläufe auf Praxistauglichkeit untersucht. Dies schloß auch eine komfortable, aber simple Benutzerführung ein, da sich vor allem die Benutzerakzeptanz im Testbetrieb als eine der wichtigsten Voraussetzungen einer sinnvollen Verwendung kryptographischer Techniken herausstellte.
Es wurde ein System in Betrieb genommen und ausgiebig getestet, welches eine konsistente und komfortable Sicherung von E−Mail−Diensten mit dem S/MIME−Protokoll ermöglicht. Neben der schon genannten Zertifizierungsstelle umfaßte dies vor allem auch Analysen der verwendeten Client−Software, welche konkrete Benutzeranweisungen zur Konfiguration der Client−Software zum Ergebnis hatten. Eine vertrauliche Weiterleitung interner, ursprünglich unverschlüsselter Nachrichten an externe Benutzer wurde konzipiert, realisiert und die Sicherheitsnachteile analysiert.
Desweiteren wurde der kryptographisch gesicherte WWW−Zugriff mit sicherer Benutzerauthentifizierung realisiert, welchem eine besondere Rolle zukommt, da dieser auch in Form von WWW−Gateways zur Sicherung anderer Server−Dienste benutzt wurde (z.B. WWW−Server ohne SSL, Datei−Server, LDAP−Server). Verschiedene unübliche Beispielanwendungen zeigen die Verwendung von kryptographischen Techniken zur Teilsicherung von Datenkommunikation bzw. der Einbettung von konventionellen Sicherheitsmechanismen (z.B. Kennwörter) in verschlüsselte Verbindungen, auch und gerade um Anregungen für weitere derartige Anwendungen zu geben.
Des weiteren wurde ein Verzeichnisdienst in Betrieb genommen, welcher, neben der Funktion als Personaldatenbank, primär in dieser Arbeit der Verteilung von Zertifikatdaten dient, aber auch Autorisierungsschemata aufnehmen kann.

Als zentrale Erkenntnisse kann man festhalten: Absolute Sicherheit erreicht man nicht, auch nicht durch den Einsatz kryptographischer Techniken. Ein wesentliches Sicherheitsrisiko bei der Benutzung einer kryptographischen Infrastruktur bleibt der Mensch als Benutzer dieser Infrastruktur, da die erforderliche Sorgfalt im Umgang mit den kryptographischen Anwendungen von diesem oft als lästig und unproduktiv empfunden wird.
Des weiteren wurde erkannt, daß mit zunehmenden Kommunikationsbeziehungen eine saubere Unterscheidung zwischen internen und externen Systemen immer schwerer möglich ist. Konzepte, welche auf einer solchen Trennung aufbauen, sind entweder als unsicher anzusehen oder bedürfen einer sorgfältigen Einschränkung der Zugriffe.
Die in dieser Arbeit gewonnenen Erfahrungen können ohne weiteres auf andere Firmen übertragen werden, da die Konzepte keinerlei für die Propack Data GmbH spezifische Einschränkungen aufweisen.

Man sollte den eingeschlagenen Weg mit folgenden Zielen weiterverfolgen:

- Konsequente Sicherung aller Internet−Dienste gegen unbefugtes Mitlesen mittels SSL−Protokoll, wobei möglichst externe und interne Dienste gleich behandelt werden sollten.
- Die ausschließliche Verwendung nicht proprietärer, kryptographischer Standards sollte beibehalten werden. Die Weiterentwicklung dieser Standards, besonders im Hinblick auf PKIX, muß kontinuierlich beobachtet und ggf. übernommen werden.
- Es sollte nur Software zum Einsatz kommen, welche starke kryptographische Techniken beinhaltet, also keine Kryptosysteme mit verkürzter Schlüssellänge.
- Die zentrale, firmenweite Benutzerverwaltung mit Rechtevergabe auf einem Verzeichnisdienst sollte gegenüber anderen, herstellerabhängigen Lösungen favorisiert werden.
- Alle verwendeten Systeme sollten zur Authentifizierung bzw. Verwaltung der Autorisierungsschemata ihrer Benutzer den Verzeichnisdienst und Benutzerzertifikate benutzen.
- Private Schlüsseldaten sollten nicht mehr von Client−Software auf Arbeitsplatzrechnern verwaltet werden. Vielmehr sollte jeder Benutzer eine Krypto−Chipkarte erhalten, welche den privaten Schlüssel enthält. Man sollte darauf achten, daß die entsprechende Software und Hardware den PKCS#11−Standard unterstützen, um auch hier ausschließlich herstellerunabhängig kompatible Systeme zu verwenden.
- Der eigentliche Zertifizierungsprozeß muß konsequent automatisiert werden, um Fehler seitens der CA−Betreuer auszuschließen und eine lückenlose Dokumentation des Zertifizierungsbetriebs zu ermöglichen. Im Zuge dieser Automatisierung ist auch die Auswahl der verwendeten Software zu überdenken − ggf. sollte ein anderes Softwarepaket eingekauft werden.
- Die Benutzerführung muß weiter verbessert werden. Fehlerhafte Eingaben seitens des Benutzers sollten an manchen Stellen des Zertifizierungsprozesses früher mit entsprechenden Klartexthinweisen abgewiesen werden. Spezielle Eingabeformulare für eingeschränkte Spezialfälle und eine Lokalisierung der CGI−BIN−Programme für verschiedene Sprachen sind ebenfalls hilfreich für den problemlosen Einsatz der PKI durch den Benutzer.
- Vor allem muß eine systematische Schulung der Mitarbeiter zur Nutzung kryptographischer Techniken erfolgen, insbesondere, um überhaupt bei den Benutzern eine Sensibilität für sicherheitsrelevante Fragen zu schaffen.

Zur Verwirklichung dieser Ziele ist vor allem eine herstellerübergreifende Standardisierung vonnöten, d.h. man muß stets darauf achten, daß eine technische Detail−Lösung eine breite Zustimmung und Unterstützung seitens verschiedener Softwarehersteller erfährt. Gerade bei dem globalen Einsatz von Verschlüsselungstechniken gerät man mit proprietären Lösungen zu schnell in eine Sackgasse und schränkt die Einsatzmöglichkeiten unnötig ein.

A Literaturverzeichnis

[2600] Hacked Pages; http://www.2600.com/hacked_pages/

[Ada98] Internet X.509 Public Key Infrastructure − Certificate Management Protocols; C. Adams (Entrust Technologies), S. Farrell (SSE); Nov. 1998; PKIX Working Group Internet Draft; http://www.ietf.org/internet−drafts/draft−ietf−pkix−ipki3cmp−09.txt

[Apa98a] Apache 1.3 User's Guide; http://www.apache.org/docs/

[Apa98b] Apache Server Frequently Asked Questions; http://www.apache.org/docs/misc/FAQ.html

[ApWe97.4] Publishing Pages with PUT; Apache Week; Issue 59: 4th April 1997; http://www.palmira.net/apache/apache_put.html

[Boey98] Internet X.509 Public Key Infrastructure − LDAPv2 Schema; Sharon Boeyen (Entrust), Tim Howes (Netscape), Pat Richard (Xcert); September 1998; http://www.ietf.org/internet−drafts/draft−ietf−pkix−ldapv2−schema−02.txt

[Bra97] A Survey Of Public−Key Infrastructures; Marc Branchaud; März 1997; http://www.xcert.com/~marcnarc/PKI/

[BrPr98] List of Browser Security Problems (Netscape, MSIE, Lynx); ; http://www.columbia.edu./acis/rad/browser−bugs.html; November 29, 1998

[BSI97a] Ende−zu−Ende−Sicherheit für elektronischen Dokumentenaustausch − Infrastruktur und Leitlinien für die Bundesverwaltung; F. Bauspieß, R. Rudeloff, J.−U. Aden; Version 0.12, 15.06.1997; http://www.bsi.bund.de/literat/sphinx/endeende.htm

[BSI97b] MailTrusT Ergänzungsspezifikation; 1998 http://www.bsi.bund.de/literat/sphinx/mailtrus.htm

[BSI97c] DNS−Spoofing; Das Computer Emergency Response Team im Bundesamt für Sicherheit in der Informationstechnik (cert@bsi.de); http://www.bsi.de/bsi−cert/vulner/dnsspoof.htm

[BSI98a] IT−Grundschutzhandbuch; Bundesamt für Sicherheit in der Informationstechnik; http://www.bsi.de/gshb/

[BSI98b] Sicherheit beim Betrieb von Web−Servern; Das Computer Emergency Response Team im Bundesamt für Sicherheit in der Informationstechnik (cert@bsi.de); http://www.bsi.de/bsi−cert/webserv.htm

[CERT97.25] CERT* Advisory CA−97.25.CGI_metachar; Original issue date: Nov. 10, 1997, Last revised: February 13, 1998; http://www.cert.org/advisories/CA−97.25.CGI_metachar.html

[Ches96] Firewalls und Sicherheit im Internet; William R. Cheswick, Steven M. Bellovin; 1996, Addison−Wesley−Verlag

[Dah96] Internet Commerce; Andrew Dahl & Leslie Lesnik; 1996, New Riders Publishing

[DFN97] Die Policies der DFN−PCA; DFN; 24.04.1997, http://www.pca.dfn.de/dfnpca/policy/

[Dwy97] Hyperlink Spoofing: An attack on SSL Server Authentication; Frank O'Dwyer; January 3, 1997; http://www.iol.ie/~fod/sslpaper/sslpaper.htm

[Eng98a] Apache interface to SSLeay (mod_ssl); Ralf Engelschall; http://www.engelschall.com/sw/mod_ssl/

[Eng98b] Global−Server−IDs (Verisign) or International Step−Up (Netscape) or Server Gated Cryptography (SGC) (Microsoft); Ralf Engelschall; http://www.engelschall.com/sw/mod_ssl/distrib/mod_ssl−SNAP/README.GlobalID

[FIPS140−1] FIPS PUB 140−1: Security Requirements for Cryptographic Modules; U.S. Department of Commerce, Ronald H. Brown, Secretary; 1994 January 11; http://csrc.nist.gov/fips/fips1401.htm

[Gar91] Practical Unix Security; Simson Garfinkel and Gene Spafford; 1991, O'Reilly & Associates, Inc.

[Gar95] PGP Pretty Good Privacy; Simson Garfinkel; 1995, O'Reilly & Associates, Inc.

[Gar97] Web Security & Commerce; Simson Garfinkel with Gene Spafford; 1997, O'Reilly & Associates, Inc.

[Geh94] Eine Sicherheitsarchitektur für kooperative, offene Umgebungen (GMD−Bericht Nr. 239); Michael Gehrke; 1994, R. Oldenbourg Verlag

[Ger98] Overview of Certification Systems: X.509, PGP and SKIP; E. Gerck; Version 1.8, 01.08.1998; http://www.mcg.org.br/cert.htm

[Ger98a] Why Is Certification Harder Than It Looks?; Ed Gerck; http://www.mcg.org.br/whycert.htm
[Ger98b] Towards Real−World Models of Trust: Reliance on Received Information; Ed Gerck; http://www.mcg.org.br/trustdef.htm
[Glö96] X.509v3 Certificate; Petra Glöckner; GMD−TKT; Juli 1996
[Gri94] Sicherheit für offene Kommunikation (Sicherheit in der Informations− und Kommunikationstechnik Band 4); Rüdiger Grimm; 1994, B−I−Wissenschaftsverlag
[Gund96] CGI Programmierung im World Wide Web; Shishir Gundavaram (deutsche Übersetzung von Peter Klicman); 1. Auflage 1996, O'Reilly/International Thomson Verlag GmbH & Co KG
[Gut98a] X.509 Style Guide; Peter Gutmann, pgut001@cs.auckland.ac.nz; August 1998; http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt
[Gut98b] How to recover private keys for Microsoft Internet Explorer, Internet Information Server, Outlook Express, and many others; Peter Gutmann, pgut001@cs.auckland.ac.nz; http://www.cs.auckland.ac.nz/~pgut001/pubs/breakms.txt
[Gut98c] PFX − How Not to Design a Crypto Protocol/Standard; Peter Gutmann, pgut001@cs.auckland.ac.nz; http://www.cs.auckland.ac.nz/~pgut001/pubs/pfx.html
[Hei96] Linux−Companion zur Systemadministration; Jochen Hein; 1996, Addison−Wesley−Verlag
[Hens98a] SSLeay PKCS#12 patch FAQ; Dr. Stephen Henson; http://www.drh−consultancy.demon.co.uk/pkcs12faq.html
[Hens98b] Netscape Certificate Database Information; Dr. Stephen Henson; http://www.drh−consultancy.demon.co.uk/cert7.html
[Hens98c] Certificate patcher program description; ; http://www.drh−consultancy.demon.co.uk/ca−fix.html
[Hir97] Introducing SSL and Certificates using SSLeay; Frederick J. Hirsch; 1997; http://www.camb.opengroup.org/RI/www/prism/wwwj/
[Hub98] Projektbericht: AS400 sicher ins Internet (iX 1/98 S. 104); Johannes Hubertz; 1998, Heinz Heise Verlag
[Huds98] SSLeay and SSLapps FAQ; T. J. Hudson; 10. Jan. 1998; http://www.psy.uq.oz.au/~ftp/Crypto/
[Hun95] TCP/IP Netzwerk Administration; Craig Hunt; 1995, O'Reilly Verlag
[IBM4986] Understanding LDAP; IBM Redbook 4986; http://www.redbooks.ibm.com/SG244986/4986fm.htm
[IuKDG] Informations− und Kommunikationsdienste−Gesetz (IuKDG); Beschluß des Deutschen Bundestages vom 13. Juni 1997; http://www.iid.de/rahmen/iukdgbt.html
[Kal93c] A Layman's Guide to a Subset of ASN.1, BER and DER; B. Kaliski; RSA Laboratories Technical Note; November 1993
[Ker91] Einführung in die Computersicherheit; Heinrich Kersten (Bundesamt für Sicherheit in der Informationstechnik − BSI, Bonn); 1991, R. Oldenbourg Verlag
[Koc97] JavaScript; Stefan Koch; 1997; dpunkt−Verlag Heidelberg
[Koops] Crypto Law Survey; Bert−Jaap Koops; http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm
[Kop98] Rechtsfragen der Kryptographie und der digitalen Signatur; Wolfgang Kopp; 13.01.1998; http://www.stud.uni−muenchen.de/~wolfgang.kopp/krypto.html
[Kos97] Charlie Kaufman − Security Status; Interview by Betsy Kosheff; Notes.net; 1. Okt. 1997; http://notes.net/today.nsf/cbb328e5c12843a9852563dc006721c7/5c32705b58a3bd6d85256522005eaede?OpenDocument
[Kuba98] WWW Browser Security & Privacy Flaws; Ed Kubaitis − ejk@uiuc.edu; http://www.ews.uiuc.edu/~ejk/browser−security.html
[Kuhn98] In die Röhre geguckt − Unerwünschte Abstrahlung erlaubt Lauschangriff; Markus Kuhn; c't 24/98; 1998, Heinz Heise Verlag
[LANL98] LDAP at LANL; http://www.nic.lanl.gov/~neale/doc/ldap/
[Lau98] Apache−SSL Documentation; Ben Laurie, Adam Laurie; October 12, 1998; http://www.apache−ssl.org/docs.html
[Laur96] A Supplementary Analysis of the Royal Holloway TTP−based Key Escrow Scheme; Ben Laurie <ben@algroup.co.uk>; 16 Nov 1996; http://www.algroup.co.uk/crypto/rh.html
[Leo98] ldapmodule for Python; Leonhard; http://www.csee.uq.edu.au/~leonard/dc−prj/ldapmodule/
[Löw97] Das Python−Buch; Martin von Löwis, Nils Fischbeck; 1. Auflage 1997; Addison−Wesley−Verlag;

[Luck98] Privatissimo − PGPs europäische Version 6.0 und PGP for Business Security (c't 23/98 S. 102); Nobert Luckhardt; 1998, Heinz Heise Verlag
[Mit96] The Royal Holloway TTP−based key escrow scheme; Chris J. Mitchell, Information Security Group, Royal Holloway, University of London; 8th June 1996; ftp://ftp.dcs.rhbnc.ac.uk/pub/Chris.Mitchell/istr_a2.ps
[Mraz97] Falsch verbunden − Gefahr duch DNS−Spoofing (c't 10/97 S. 286); Viktor Mraz, Klaus Weidner; 1997, Heinz Heise Verlag
[Mün98] SELFHTML: HTML−Dateien selbst erstellen ; Stefan Münz; Version 7.0 vom 27.04.1998; http://www.teamone.de/selfaktuell/
[Nets96] Netscape Extensions for User Key Generation − Preliminary Navigator 3.0 Version; Netscape; 29.06.1996; http://www1.netscape.com/eng/security/ca−interface.html
[Nets97a] Netscape SSL 2.0 Certificate Format; Netscape; http://home.netscape.com/newsref/std/ssl_2.0_certificate.html
[Nets97b] SSL Version 3.0; Netscape; http://home.netscape.com/newsref/std/SSL.html
[Nets97c] Netscape Certificate Specifications; Netscape; http://www1.netscape.com/eng/security/certs.html
[Nets97d] Netscape Certificate Extensions Specifications; Netscape; 13.08.1997; http://www1.netscape.com/eng/security/comm4−cert−exts.html
[Nets97e] Netscape Certificate Download Specification − Communicator Version 4.0; Netscape; 13.08.1997; http://www1.netscape.com/eng/security/comm4−cert−download.html
[Nets97f] Netscape Extensions for User Key Generation − Communicator Version 4.0; Netscape; 13.08.1997; http://www1.netscape.com/eng/security/comm4−keygen.html
[Nets97g] LDIF−Format; NDS 3.0 Dokumentation; Netscape; http://home.netscape.com/eng/server/directory/3.0/ag/ldif.htm
[Nets97h] Object classes; NDS 3.0 Dokumentation; Netscape; http://home.netscape.com/eng/server/directory/3.0/ag/objclass.htm
[Nets97i] Attributes; NDS 3.0 Dokumentation; Netscape; http://home.netscape.com/eng/server/directory/3.0/ag/attribute.htm
[Nets97j] The SSL Protocol Version 3.0; Alan O. Freier, Philip Karlton, Paul C. Kocher; Internet Draft March 1996; http://home.netscape.com/eng/ssl3/ssl−toc.html
[Nets98a] Introduction to SSL; Netscape; http://developer.netscape.com/docs/manuals/security/sslin/index.htm
[Nets98b] Security − Signed Forms; Netscape; http://developer.netscape.com/tech/security/formsign/formsign.html
[Nets98c] Introduction to Public−Key Cryptography; Netscape; http://developer.netscape.com/docs/manuals/security/pkin/index.htm
[Nets98d] Signing Text from JavaScript; Netscape; http://developer.netscape.com/docs/manuals/security/sgntxt/
[Nets98e] Using the Signature Verification Tool; Netscape; http://developer.netscape.com/docs/manuals/security/signver/
[Nets98f] Object Signing Resources; Netscape; http://developer.netscape.com/docs/manuals/signedobj/overview.html
[PCGI] Persistent CGI; http://www.digicool.com/releases/pcgi/
[Pesch98] Key Recovery Utilities and Other Resources; Joe Peschel; http://members.aol.com/jpeschel/crack.htm
[PKCS10] PKCS#10: Certification Request Syntax Standard; RSA Laboratories Technical Note; Version 1.5; Revised November 1, 1993; http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs−10.html
[PKCS11] PKCS#11: Cryptographic Token Interface Standard; RSA Laboratories Technical Note; http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs−11.html
[PKCS12] PKCS#12: Personal Information Exchange Syntax Standard; RSA Laboratories Technical Note; http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs−12.html
[PKCS6] PKCS#6: Extended−Certificate Syntax Standard; RSA Laboratories Technical Note; Version 1.5; Revised November 1, 1993; http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs−6.html
[PKCS7] PKCS#7: Cryptographic Message Syntax Standard; RSA Laboratories Technical Note; Version 1.5; Revised November 1, 1993; http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs−7.html

[PKCS9] PKCS#9: Selected Attribute Types; RSA Laboratories Technical Note; Version 1.5; Revised November 1, 1993; http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs−9.html
[PKCSex] Some Examples of the PKCS Standards; B. Kaliski; RSA Laboratories Technical Note; November 1993
[PKCSov] An Overview of the PKCS Standards; B. Kaliski; RSA Laboratories Technical Note; November 1993; http://www.rsa.com/rsalabs/pubs/PKCS/
[PKIX] Public−Key Infrastructure (X.509) (pkix) Charter; IETF Working Group; http://www.ietf.org/html.charters/pkix−charter.html
[Plum98] Signed Form Example; Alec Plumb; http://developer.netscape.com/tech/security/formsign/stex.html
[Ranu98] Internet Attacks; Marcus J. Ranum; http://www.clark.net/pub/mjr/pubs/attck/index.htm
[Rei97] Herr der Zertifikate − Zertifikatsmanagment und Online−Beglaubigungsstellen mit SSLeay (iX 4/97); Holger Reif; 1997, Heinz Heise Verlag
[RFC1421] RFC1421 − Privacy Enhancement for Internet Eletronic Mail: Part I: Message Encryption and Authentication Procedures; John Linn; Februar 1993; ftp://ftp.isi.edu/in−notes/rfc1421.txt
[RFC1422] RFC1422 − Privacy Enhancement for Internet Eletronic Mail: Part II: Certificate−Based Key Managment; S. Kent; Februar 1993; ftp://ftp.isi.edu/in−notes/rfc1422.txt
[RFC1423] RFC1423 − Privacy Enhancement for Internet Eletronic Mail: Part III: Algorithms, Modes and Identifiers; D. Balenson; Februar 1993; ftp://ftp.isi.edu/in−notes/rfc1423.txt
[RFC1424] RFC1424 − Privacy Enhancement for Internet Eletronic Mail: Part IV: Key Certification and Related Services; B. Kaliski; Februar 1993; ftp://ftp.isi.edu/in−notes/rfc1424.txt
[RFC2045] RFC 2045 − Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. N. Freed & N. Borenstein. November 1996; ftp://ftp.isi.edu/in−notes/rfc2045.txt
[RFC2046] RFC 2046 − Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. N. Freed & N. Borenstein. November 1996; ftp://ftp.isi.edu/in−notes/rfc2046.txt
[RFC2047] RFC 2047 MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non−ASCII Text. K. Moore. November 1996; ftp://ftp.isi.edu/in−notes/rfc2047.txt
[RFC2048] RFC 2048 − Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures. N. Freed, J. Klensin & J. Postel. November 1996; ftp://ftp.isi.edu/in−notes/rfc2048.txt
[RFC2049] RFC 2049 − Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples. N. Freed & N. Borenstein. November 1996; ftp://ftp.isi.edu/in−notes/rfc2049.txt
[RFC2065] RFC2065 − Domain Name System Security Extensions; D. Eastlake, 3rd (Cybercash), C. Kaufman (Iris); Januar 1997; ftp://ftp.isi.edu/in−notes/rfc2065.txt
[RFC2256] RFC2256 − A Summary of the X.500(96) User Schema for use with LDAPv3; M. Wahl, Critical Angle Inc.; Dezember 1997; ftp://ftp.isi.edu/in−notes/rfc2256.txt
[RFC2311] RFC 2311 − S/MIME Version 2 Message Specification. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka. March 1998; ftp://ftp.isi.edu/in−notes/rfc2311.txt
[RFC2312] RFC 2312 − S/MIME Version 2 Certificate Handling. S. Dusse, P. Hoffman, B. Ramsdell, J. Weinstein. March 1998; ftp://ftp.isi.edu/in−notes/rfc2312.txt
[RFC2377] RFC 2377 − A Directory Naming Plan; A. Grimstad, R. Huber, S. Sataluri, M. Wahl; September 1998; ftp://ftp.isi.edu/in−notes/rfc2377.txt
[RFC2459] RFC 2459 − Internet X.509 Public Key Infrastructure Certificate and CRL Profile; R. Housley, W. Ford, W. Polk, D. Solo; January 1999; ftp://ftp.isi.edu/in−notes/rfc2459.txt
[Ric98] web500gw: A WWW/HTTP−LDAP/X.500 Gateway; Frank Richter; http://www.tu−chemnitz.de/~fri/web500gw/
[RSA95] S/Mime Implementation Guide − Interoperability Profile, Version 1; S/Mime Editor; RSA Data Security Inc.; August 1995
[RSA97] S/MIME Freeware Library − Fact Sheet; 7. November 1997; http://www.rsa.com/smime/html/vda_freeware.html
[Schm97a] Kidnapping im Netz (c't 10/97 S. 142); Jürgen Schmidt; 1997, Heinz Heise Verlag
[Schm97b] Falsche Fährten − Mißbrauchsmöglichkeiten von ARP und ICMP (c't 12/97 S. 246); Jürgen Schmidt; 1997, Heinz Heise Verlag

[Schn96] Angewandte Kryptographie; Bruce Schneier; 1996, Addison−Wesley−Verlag
[Schn97a] Why Cryptography Is Harder Than It Looks; Bruce Schneier; http://www.counterpane.com/whycrypto.html
[Schu99] Markt− oder Staatsmacht − Streit um digitale Signaturen; Christiane Schulzki−Haddouti; c't 1/99; 1998, Heinz Heise Verlag
[Schul96] Inside the Windows 95 Registration Wizard; Andrew Schulman; January 2, 1996; ftp://ftp.ora.com/pub/examples/windows/win95.update/regwiz.html
[SigV] Verordnung zur digitalen Signatur (SigV); Beschluß der Bundesregierung vom 08. Oktober 1997; http://www.iid.de/rahmen/sigv.html
[SMIME] S/MIME Working Group; http://www.imc.org/ietf−smime/
[Ste96] Transparente Netzwerktrennung zur Erhöhung der Sicherheit (Dissertation Univ. Karlsruhe); Steffen Stempel; 1996, Herbert Utz Verlag
[Strö98a] cgiforms.py; Michael Ströder; 1998; http://www.uni−karlsruhe.de/~un01/Python/cgi/
[Strö98b] ldap−client−cgi.py; Michael Ströder; 1998; http://www.uni−karlsruhe.de/~un01/ldap/
[Thaw98a] The Strong Extranet − A Managed Certificate Solution for secure web access control; Thawte; http://www.thawte.com/certs/strongextranet/
[TLS] Transport Layer Security (tls); IETF Working Group; http://www.ietf.org/html.charters/tls−charter.html
[Ver98a] Certification Practice Statement; Verisign; http://www.verisign.com/repository/CPS/
[Ver98b] Certification Practice Statement − Quick Summary of Important CPS Rights and Obligations; Verisign; http://www.verisign.com/repository/summary.html
[Wäch98] Überblick über das Chaos − LDAP, die neue Auskunft; Peter Wächtler; Linux−Magazin 9/98; http://www.linux−magazin.de/ausgabe.1998.09/LDAP/ldap.html
[X509] ITU−T Recommendation X.509; ITU−T; 11/9306/97; ftp://ftp.bull.com/pub/OSIdirectory/ITU/97x509final.doc
[IPSec] IP Security Protocol (ipsec) Charter; IETF Working Group; http://www.ietf.org/html.charters/ipsec−charter.html
[IBM98] eNetwork Whitepaper VPN; IBM Corporation; http://www.software.ibm.com/enetwork/library/whitepapers/vpn/

B URL−Verzeichnis

Dieses URL−Verzeichnis enthält allgemeine URL−Adressen zu bestimmten Sachgebieten, meist Software−Projekte oder Organisationen, aber keine Hinweise auf spezielle Literatur. Diese stehen im Anhang A.

Apache	http://www.apache.org/
ApacheSSL	http://www.apache−ssl.org/
CERT	http://www.cert.org/
Cryptozilla	http://www.cryptozilla.org/
Fast−CGI	http://www.fastcgi.org/
Fortify	http://www.fortify.net/
Mozilla	http://www.mozilla.org/
OpenLDAP	http://www.openldap.org/
OpenSSL	http://www.openssl.org/
Python	http://www.python.org/
Rootshell	http://www.rootshell.com/
SECUDE	http://www.darmstadt.gmd.de/secude/
SSLeay	http://www.ssleay.org/
stunnel	http://mike.daewoo.com.pl/computer/stunnel/stunnel.html
Thawte	http://www.thawte.com/
Verisign	http://www.verisign.com/
Opera	http://www.operasoftware.com/
Persistent CGI	http://www.digicool.com/releases/pcgi/

C Konfigurationsdatei ssleay.cnf

Dieser Anhang enthält die zentrale Konfigurationsdatei für den Einsatz von SSLeay. Hier ist allerdings nur eine zu lokalen Testzwecken verwendete Datei abgedruckt, die allerdings alle wesentlichen Parameter der verwendeten zweistufigen CA−Hierarchie und die relevanten Zertifikatattribute beinhaltet.

```
#
# SSLeay configuration file: Zweistufige CA−Hierarchie
# This is mostly being used for generation of certificate requests.
#

RANDFILE         = $ENV::HOME/.rnd
oid_file       = $ENV::HOME/.oid

######################################################################
[ ca ]

Persona_ca     = CA_Persona
Partner_ca     = CA_Partner
Autorisiert_ca = CA_Autorisiert
Mitarbeiter_ca = CA_Mitarbeiter
Objekt_ca      = CA_Objekt
Admin_ca       = CA_Admin
RA_ca   = CA_RA
Server_ca      = CA_Server
CA_ca   = CA_CA

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

[ CA_Persona ]
dir     = /usr/local/ssl/TestCA/Persona      # Where everything is kept
certs   = $dir/certs   # Where the issued certs are kept
crl_dir         = $dir/crl     # Where the issued crl are kept
database        = $dir/index.txt       # database index file.
new_certs_dir   = $dir/newcerts        # default place for new certs.
certificate     = $dir/cacert.pem      # The CA certificate
serial  = $dir/serial          # The current serial number
crl     = $dir/crl.pem         # The current CRL
private_key     = $dir/private/cakey−Persona.pem# The private key
RANDFILE        = $dir/private/.rand   # private random number file
default_days    = 7     # how long to certify for
default_crl_days= 2    # how long before next CRL
default_md      = md5   # which md to use.
preserve        = no    # keep passed DN ordering
policy  = policy_Persona
x509_extensions        = x509v3_ext_Persona

[ CA_Partner ]
dir     = /usr/local/ssl/TestCA/Partner      # Where everything is kept
certs   = $dir/certs   # Where the issued certs are kept
crl_dir         = $dir/crl     # Where the issued crl are kept
database        = $dir/index.txt       # database index file.
new_certs_dir   = $dir/newcerts        # default place for new certs.
certificate     = $dir/cacert.pem      # The CA certificate
serial          = $dir/serial  # The current serial number
crl             = $dir/crl.pem         # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand   # private random number file
default_days    = 200   # how long to certify for
default_crl_days= 2    # how long before next CRL
default_md      = md5   # which md to use.
preserve        = no    # keep passed DN ordering
policy  = policy_Partner
x509_extensions        = x509v3_ext_Partner

[ CA_Autorisiert ]
dir     = /usr/local/ssl/TestCA/Autorisiert   # Where everything is kept
certs   = $dir/certs   # Where the issued certs are kept
crl_dir         = $dir/crl     # Where the issued crl are kept
database        = $dir/index.txt       # database index file.
new_certs_dir   = $dir/newcerts        # default place for new certs.
certificate     = $dir/cacert.pem      # The CA certificate
serial  = $dir/serial          # The current serial number
crl     = $dir/crl.pem         # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand   # private random number file
default_days    = 200   # how long to certify for
default_crl_days= 2    # how long before next CRL
default_md      = md5   # which md to use.
preserve        = no                   # keep passed DN ordering
policy          = policy_Autorisiert
x509_extensions        = x509v3_ext_Autorisiert

[ CA_Mitarbeiter ]
dir             = /usr/local/ssl/TestCA/Mitarbeiter   # Where everything is kept
```

```
certs          = $dir/certs          # Where the issued certs are kept
crl_dir        = $dir/crl            # Where the issued crl are kept
database       = $dir/index.txt      # database index file.
new_certs_dir  = $dir/newcerts               # default place for new certs.
certificate    = $dir/cacert.pem     # The CA certificate
serial         = $dir/serial         # The current serial number
crl            = $dir/crl.pem               # The current CRL
private_key    = $dir/private/cakey.pem# The private key
RANDFILE       = $dir/private/.rand  # private random number file
default_days   = 200                 # how long to certify for
default_crl_days= 5                  # how long before next CRL
default_md     = md5                 # which md to use.
preserve       = no                  # keep passed DN ordering
policy         = policy_Mitarbeiter
x509_extensions       = x509v3_ext_Mitarbeiter

[ CA_Objekt ]
dir            = /usr/local/ssl/TestCA/Objekt       # Where everything is kept
certs          = $dir/certs          # Where the issued certs are kept
crl_dir        = $dir/crl            # Where the issued crl are kept
database       = $dir/index.txt      # database index file.
new_certs_dir  = $dir/newcerts               # default place for new certs.
certificate    = $dir/cacert.pem     # The CA certificate
serial         = $dir/serial         # The current serial number
crl            = $dir/crl.pem               # The current CRL
private_key    = $dir/private/cakey.pem# The private key
RANDFILE       = $dir/private/.rand  # private random number file
default_days   = 200                 # how long to certify for
default_crl_days= 5                  # how long before next CRL
default_md     = md5                 # which md to use.
preserve       = no                  # keep passed DN ordering
policy         = policy_Mitarbeiter
x509_extensions       = x509v3_ext_Objekt

[ CA_Admin ]
dir            = /usr/local/ssl/TestCA/Admin # Where everything is kept
certs  = $dir/certs   # Where the issued certs are kept
crl_dir        = $dir/crl     # Where the issued crl are kept
database       = $dir/index.txt      # database index file.
new_certs_dir  = $dir/newcerts               # default place for new certs.
certificate    = $dir/cacert.pem     # The CA certificate
serial         = $dir/serial  # The current serial number
crl            = $dir/crl.pem               # The current CRL
private_key    = $dir/private/cakey.pem# The private key
RANDFILE       = $dir/private/.rand  # private random number file
default_days   = 200   # how long to certify for
default_crl_days= 5    # how long before next CRL
default_md     = md5   # which md to use.
preserve       = no    # keep passed DN ordering
policy  = policy_Admin
x509_extensions       = x509v3_ext_Admin

[ CA_RA ]
dir    = /usr/local/ssl/TestCA/RA    # Where everything is kept
certs  = $dir/certs   # Where the issued certs are kept
crl_dir        = $dir/crl     # Where the issued crl are kept
database       = $dir/index.txt      # database index file.
new_certs_dir  = $dir/newcerts       # default place for new certs.
certificate    = $dir/cacert.pem     # The CA certificate
serial         = $dir/serial  # The current serial number
crl    = $dir/crl.pem # The current CRL
private_key    = $dir/private/cakey.pem# The private key
RANDFILE       = $dir/private/.rand  # private random number file
default_days   = 200   # how long to certify for
default_crl_days= 5    # how long before next CRL
default_md     = md5   # which md to use.
preserve       = no    # keep passed DN ordering
policy  = policy_Mitarbeiter
x509_extensions       = x509v3_ext_RA

[ CA_Server ]
dir    = /usr/local/ssl/TestCA/Server       # Where everything is kept
certs  = $dir/certs   # Where the issued certs are kept
crl_dir        = $dir/crl     # Where the issued crl are kept
database       = $dir/index.txt      # database index file.
new_certs_dir  = $dir/newcerts       # default place for new certs.
certificate    = $dir/cacert.pem     # The CA certificate
serial = $dir/serial  # The current serial number
crl    = $dir/crl.pem # The current CRL
private_key    = $dir/private/cakey.pem# The private key
RANDFILE       = $dir/private/.rand  # private random number file
default_days   = 30    # how long to certify for
default_crl_days= 2    # how long before next CRL
default_md     = md5   # which md to use.
preserve       = no    # keep passed DN ordering
policy  = policy_Server
x509_extensions       = x509v3_ext_Server

[ CA_CA ]
dir    = /usr/local/ssl/TestCA/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       = $dir/index.txt      # database index file.
new_certs_dir  = $dir/newcerts       # default place for new certs.
```

```
certificate    = $dir/cacert.pem     # The CA certificate
serial  = $dir/serial  # The current serial number
crl    = $dir/crl.pem # The current CRL
private_key    = $dir/private/cakey.pem# The private key
RANDFILE       = $dir/private/.rand   # private random number file
default_days   = 200   # how long to certify for
default_crl_days= 5    # how long before next CRL
default_md     = md5   # which md to use.
preserve       = no    # keep passed DN ordering
policy  = policy_CA
x509_extensions        = x509v3_ext_CA

############################ Policies ################################
[ policy_Persona ]
countryName    = optional
stateOrProvinceName    = optional
localityName   = optional
organizationName       = optional
organizationalUnitName         = optional
commonName     = supplied
initials       = optional
emailAddress   = supplied

[ policy_Partner ]
countryName    = supplied
stateOrProvinceName    = optional
localityName   = optional
organizationName       = supplied
organizationalUnitName         = optional
commonName     = supplied
initials       = optional
emailAddress   = supplied

[ policy_Autorisiert ]
countryName    = supplied
stateOrProvinceName    = optional
localityName   = optional
organizationName       = supplied
organizationalUnitName         = optional
commonName     = supplied
initials       = optional
#uid    = optional
emailAddress   = supplied

[ policy_Mitarbeiter ]
countryName    = supplied
stateOrProvinceName    = optional
localityName   = supplied
organizationName       = match
organizationalUnitName         = supplied
commonName     = supplied
initials       = optional
#uid    = supplied
emailAddress   = supplied

[ policy_RA ]
countryName    = supplied
stateOrProvinceName    = optional
localityName   = supplied
organizationName       = match
organizationalUnitName         = supplied
commonName     = supplied
initials       = supplied
emailAddress   = supplied

[ policy_Admin ]
countryName    = match
stateOrProvinceName    = optional
localityName   = supplied
organizationName       = match
organizationalUnitName         = supplied
commonName     = supplied
initials       = supplied
#uid    = supplied
emailAddress   = supplied

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

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

```
################### certification request params ####################

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

[ req_distinguished_name ]
countryName    = Country Name (2 letter code)
countryName_default    = DE
countryName_min        = 2
countryName_max        = 2

stateOrProvinceName    = State or Province Name (full name)
stateOrProvinceName_default   = Baden-Württemberg

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

0.organizationName             = Organization Name (eg, company)
0.organizationName_default     = Michael's

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

organizationalUnitName         = Organizational Unit Name (eg, section)
# organizationalUnitName_default      =

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

emailAddress   = Email Address
emailAddress_max       = 64

initials       = Initials of an individuals names
initials_max   = 5

uid     = User ID to logon
uid_max        = 8

[ req_attributes ]
challengePassword      = A challenge password
challengePassword_min  = 4
challengePassword_max  = 20

[ x509v3_ext_Persona ]
nsComment      = "Anonymous Mail (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Persona
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Persona?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Persona?
nsCaPolicyUrl  = TestCA/policy/persona-policy.html
nsCertType     = 0x20
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_Partner ]
nsComment      = "E-Mail Partner (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Partner
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Partner?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Partner?
nsCaPolicyUrl  = TestCA/policy/partner-policy.html
nsCertType     = 0x20
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_Autorisiert ]
nsComment      = "Further access (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Autorisiert
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Autorisiert?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Autorisiert?
nsCaPolicyUrl  = TestCA/policy/autorisiert-policy.html
nsCertType     = 0x80
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_Mitarbeiter ]
nsComment      = "Member of Michael's (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Mitarbeiter
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Mitarbeiter?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Mitarbeiter?
nsCaPolicyUrl  = TestCA/policy/mitarbeiter-policy.html
nsCertType     = 0xa0
```

```
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_Objekt ]
nsComment      = "Object Signer of Michael's (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Objekt
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Objekt?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Objekt?
nsCaPolicyUrl  = TestCA/policy/objekt-policy.html
nsCertType     = 0x10
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_Admin ]
nsComment      = "Administrator of Michael's (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Admin
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Admin?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Admin?
nsCaPolicyUrl  = TestCA/policy/admin-policy.html
nsCertType     = 0x80
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_RA ]
nsComment      = "Regional Authority of Michael's (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_RA
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_RA?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_RA?
nsCaPolicyUrl  = TestCA/policy/ra-policy.html
nsCertType     = 0x20
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_CA ]
nsComment      = "Certification Authority of Michael's (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_CA
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_CA?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_CA?
nsCaPolicyUrl  = TestCA/policy/ca-policy.html
nsCertType     = 0x07
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType

[ x509v3_ext_Server ]
nsComment      = "Server of Michael's (TestCA)"
nsBaseUrl      = https://www.ms.my/
nsCaRevocationUrl      = cgi-bin/TestCA/get-crl.py/CA_Server
nsRevocationUrl        = cgi-bin/TestCA/ns-check-rev.py/CA_Server?
nsRenewalUrl   = cgi-bin/TestCA/ns-renewal.py/CA_Server?
nsCaPolicyUrl  = TestCA/policy/server-policy.html
nsCertType     = 0x40
```

D Programme und Module

Dieser Teil des Anhangs gibt eine Übersicht über die im Rahmen der Arbeit erstellten Programme und Module. Alle Python−Quellcodes erfordern eine Python 1.5.1−Installation, werden zusätzliche Module oder Patches benötigt, so ist dies beim jeweiligen Quelltext angegeben.

D.1 Python−Module

Modul charset.py
Diverse Konvertierungen von Zeichensätzen.
Modul htmlbase.py
Rudimentäre HTML−Ausgaben, insbesondere für Kopf− und Ende−HTML−Tags.
Modul ipadr.py
Umrechnungen für IP−Adressen.
Modul cgiforms.py
Ein Modul zur sicheren Behandlung von Eingabedaten für CGI−BIN−Programme [Strö98a].
Modul cgissl.py
Ermittlung und Anzeige der aktuell bei einer CGI−BIN−Sitzung vorhandenen SSL−Daten. Derzeit werden nur die SSL−relevanten Umgebungsvariablen von ApacheSSL ausgewertet.
Modul relay.py
HTTP−Proxy−Funktion, welche über HTTP ein weiteres CGI−BIN−Programm mit Eingabedaten aus einem cgiforms.FormClass()−Formular aufrufen kann. Es wird ein Patch für Python 1.5.1 benötigt, welcher einen Bug in der File−Object−Funktion des Socket−Moduls behebt.
Modul ssleay
Dieses Modul ist als Verzeichnis mit mehreren Teilmodulen realisiert.

- **ssleay/__init__.py**
  Initialisierungsteil
- **ssleay/cnf.py**
  Dient dem Einlesen der SSLeay−Konfigurationsdatei (meist ssleay.cnf).
- **ssleay/db.py**
  Auslesen und Manipulieren der SSLeay−Zertifikatdatenbank (index.txt).
- **ssleay/cert.py**
  Auslesen von Daten aus Zertifikaten.

D.2 Programme

smime−forward.py
Weiterleitung von auf stdin angelieferten E−Mails als S/MIME−verschlüsselte E−Mail auf stdout. Aufruf über /etc/aliases (bei sendmail) oder .procmailrc (bei procmail) (siehe auch Abschnitt 7.2.2). Es werden spezielle, nicht frei verfügbare S/MIMEv2−Programme von Dr. Stephen Henson verwendet.

D.2.1 Zertifizierungsstellenbetrieb

ca−cert−mail−challenge.py
Dieses Modul behandelt die E−Mail−Antwort eines Benutzers beim E−Mail−Dialog während des Zertifizierungsantrags (siehe Abschnitt 6.7.2).
ca−expire.py
Abgelaufene Zertifikate in der SSLeay−DB als »expired« markieren. Wird üblicherweise zyklisch als CRON−Prozeß aufgerufen.

D.2.2 Rund um LDAP

Alle Skripten in diesem Abschnitt brauchen die LDAP−Programme bzw. die Bibliotheken *liblber* und *libldap* aus dem UMich−LDAP−Server bzw. von OpenLDAP.
passwd2ldif.py
Sehr auf die Konfiguration bei der Propack Data GmbH spezialisiertes Skript, welches aus einer /etc/passwd mit speziellem Format, einer SSLeay−DB, einem Verzeichnis mit JPEG−Bildern der Mitarbeiter und einer weiteren Liste mit Personaldaten eine LDIF−Datei mit allen Mitarbeiterinformationen generiert.
ssleayca2ldif.py
Mit diesem Skript wird eine LDIF−Datei mit sämtlichen CA−Daten aus der ssleay.cnf generiert.
ssleaycerts2ldif.py
Dieses Skript generiert eine LDIF−Datei mit sämtlichen Zertifikatdaten der Benutzer.
ssleaycerts2ldap.py
Dieses Skript sendet die Benutzerzertifikate online an einen LDAP−Server. Es wird das Python−Modul ldapmodule benötigt [Leo98].

D.3 CGI−BIN−Programme

D.3.1 WWW−Gateways

ldap−client−cgi.py
LDAP−WWW−Gateway, siehe Abschnitt 7.4.1 und [Strö98b]. Es wird das Python−Modul ldapmodule benötigt [Leo98].
mount.py
−WWW−Gateway, siehe Abschnitt 7.4.2.

D.3.2 Zertifizierungsstellenbetrieb

cert−query.py
Abfrage der SSLeay−DB mit Suchbegriffen.
get−crl.py
Abruf einer CRL.
get−cert.py
Abruf eines Zertifikates.
ns−check−rev.py
Online−Prüfung eines Zertifikates.
ns−enroll.py
Erzeugung einer Zertifikatanforderung mit Netscape Navigator.

ns−renewal.py
Erneuerungsantrag für ein schon existierendes Zertifikates.
ns−revoke.py
Zertifikatwiderruf durch Zertifikatinhaber

D.3.3 Prototypen

signed−forms−demo.py
Eine prototypische Implementierung der Verarbeitung von Signed Forms, siehe Abschnitt 7.5.2. Es wird das *signver*−Programm von Netscape benötigt.