Internetdienste

PEARSON Education

ADDISON-WESLEY

Peter Gergen

Internetdienste

Aufbau von Mail, Directory, WWW, Certificate Authority und Co.

ADDISON-WESLEY

An imprint of Pearson Education

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

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

**Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich.**

10 9 8 7 6 5 4 3 2 1

05 04 03 02

ISBN 3-8273-1926-9

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

Einbandgestaltung: atelier für gestaltung, niesner & huber, Wuppertal
Lektorat: Rolf Pakendorf, rpakendorf@pearson.de
Korrektorat: Petra Kienle, Fürstenfeldbruck
Herstellung: Anna Plenk, aplenk@pearson.de
Satz: reemers publishing services gmbh, Krefeld, www.reemers.de
Druck und Verarbeitung: Bercker, Kevelaer
Printed in Germany

Inhaltsverzeichnis

Für Klaudia

Vorwort

Seit ich zum ersten Mal eine Webseite auf meinem Monitor gesehen habe, bin ich von der Idee fasziniert, mit einem simplen Klick mit der Maus eine Informationen vom anderen Ende der Welt herunterzuladen. Ein einziger Tastendruck löst eine Aktion aus, die viele tausend Kilometer entfernt einen Rechner veranlasst, mir zu antworten. Ein Brief wird fast unmittelbar nach dem Betätigen des »Send«-Kommandos zum Adressaten transportiert. Ein Gesprächspartner, den man noch nie persönlich getroffen hat, sitzt am anderen Ende der Leitung und übermittelt seine Meinungen, Anschauungen und Gefühle.

Der vertraute Umgang mit dem Internet lässt im Laufe der Zeit eine Art Scheinwelt entstehen, in der alles von jedem Ort aus zu erreichen ist. Mit einigen Mausklicks erhält man Daten aus allen Teilen der Welt, ohne sich Gedanken darüber zu machen, wie eigentlich diese vielen Informationen und Bilder hinterlegt wurden, welche Intention der Autor mit der Veröffentlichung verfolgt, welche Technik zur Datenübertragung erforderlich ist, wer alles bei der Kommunikation mithören mag, welchen Weg die Daten gehen und von welchem Winkel der Erde sie stammen. Mich haben diese Fragen immer fasziniert und seit dem ersten Mausklick auf eine Webseite versuche ich herauszufinden, was eigentlich »dahinter steckt«.

Mit dem vorliegenden Buch möchte ich meine Erfahrungen mit dem Medium Internet weitergeben. Einige der dabei vorgestellten Techniken werden unter Umständen dazu verleiten, Dinge auszuprobieren, die über das tägliche Bedienen des Browsers hinausgehen. Mit Sicherheit ist dies eine sehr gute Möglichkeit, unter die »Motorhaube« des Internets zu blicken. Sie können dieses Medium neu erfahren und einiges über dessen Mechanismen lernen. Die angebotenen Informationen sind keine Geheimnisse und werden sowohl im Internet als auch in vielen Fachpublikationen detailliert beschrieben. Wenn die Kommunikationsprotokolle es erlauben, dass man sie über Telnet-Verbindungen nachstellt (in den Listings durch Fettdruck hervorgehoben), bekommt man einen guten Einblick in die Interaktion zwischen Client und Server. Je nach Hersteller sind die Reaktionen der Server unterschiedlich, die Abläufe und Verfahren sind größtenteils gleich.

Es sei jedoch betont, dass die vorgestellten Techniken unter Umständen Schaden anrichten, wenn sie unbedacht oder vorsätzlich falsch angewandt werden. In diesem Zusammenhang kann weder der Autor noch der Verlag eine Haftung für Schäden aufgrund fehlerhafter oder missbräuchlicher Anwendung der vorgestellten Techniken oder Beispiele übernehmen.

Peter Gergen
peter.gergen@addison-wesley.de
München

1 Einführung

Netzwerktopologien, wie sie seit etwa Mitte der neunziger Jahre eingesetzt werden, erfuhren einen grundlegenden Wandel von proprietären Datenübertragungsprotokollen hin zu standardisierten Kommunikationsmodellen. Die Entwicklung des Internets als populäre Plattform für den Dokumentenaustausch trug hierzu maßgeblich bei. Das Aufkommen von Webseiten, die Informationen nicht nur rein textbasiert übermitteln, sondern ein Netz von Informationen über »Links« knüpfen, machte den Siegeszug der Standardisierung erst möglich. Weitere Antriebskräfte waren die Darstellbarkeit der Information über freie Font-Wahl bis hin zur Multimediaunterstützung mittels Bildern und Grafiken. Multimediale Inhalte machen mittlerweile einen Großteil der übertragenen Informationen aus, von Fotos bis hin zu Musikvideos ist jede Form der digitalen Kommunikation möglich. Das Internet hat jedoch sehr viel ältere Wurzeln, seine Entstehungsgeschichte reicht bis weit in die sechziger Jahre zurück. Das wird oft übersehen, da der eigentliche Hype erst durch das Aufkommen der Webseiten seinen Anfang nahm.

2 Überblick

Das Wachstum des Internets verläuft nahezu ungebremst. Während der letzten acht Jahre ist die Anzahl der registrierten Hosts fast exponentiell gestiegen. Etwa alle eineinhalb Jahre hat sich deren Zahl verdoppelt. Kein anderes Medium erfuhr bisher eine vergleichbare Entwicklung. Die Präsenz der neuen Informationsverteilung und Kommunikationsform scheint allgegenwärtig: E-Mail-Adressen, die noch vor fünfzehn Jahren nur von einem Kreis von »Eingeweihten« verwendet wurden, gehören heute zur Kontaktinformation wie Telefonnummer und Postanschrift. Kaum ein Unternehmen kann es sich noch leisten, keinen Internetauftritt zu haben. Selbst in die Justiz und Behörden hat dieses Medium Einzug gehalten, sei es in Form von Prozessen über Domain-Namen oder der elektronischen Steuererklärung. Produktwerbungen oder Nachrichtenmagazine warten mit einer URL genauso auf wie Fernsehsendungen und Kinowerbungen.

Die Möglichkeiten des neuen Massenmediums Internet erkannte man sehr schnell. Eine breite Palette an Anwendungen ermöglicht es dem Einzelnen, zu kommunizieren, Daten auszutauschen, einzukaufen oder sich einfach nur zu informieren. Fast scheint es, als wären die vergangenen Jahre dadurch geprägt, dass diese neue Form der Kommunikation und des Informationsaustauschs entwickelt wurde. Nur wenigen ist bekannt, dass dieses weltweite Netz bereits seit den sechziger Jahren existiert, in denen die ersten Mails ausgetauscht wurden.

2.1 Historische Entwicklung des Internets

Die Entstehung des Internets lässt sich zum großen Teil auf die Situation zu Zeiten des kalten Krieges zurückführen. Der erste sowjetische Satellit im All löste in den USA tiefe Besorgnis aus und führte zu einer Reihe von wissenschaftlichen Tätigkeiten in vielen Bereichen. Unter anderem entstand eine Forschungsabteilung des Verteidigungsministeriums, die Advanced Research Projects Agency (ARPA).

Im militärischen Bereich kamen bereits recht früh Computer zum Einsatz, die miteinander vernetzt waren. Die Kommunikation geschah über Knotenrechner, deren Ausfall den Rest des Netzes lahm legte. Im Gegensatz zu heute wurde ein solches Netz zentral geführt, ein Ausfall dieser Zentrale führte meist zum Erliegen der Kommunikation aller beteiligten Rechner. Man erkannte, dass ein ausfallsicheres Netz auf einer dezentralen und paket-

orientierten Architektur mit alternativen Datenübertragungswegen basiert. Als mit dem ARPA-Netzwerk 1969 die ersten vier Computer unter diesen Rahmenbedingungen miteinander verbunden wurden, war der Vorläufer des Internets geboren.

1970 wurde das ARPA-Netzwerk sukzessive ausgebaut, es umfasste insgesamt 23 Rechner, die über die gesamten USA verteilt waren. Etwa zeitgleich wurden Anwendungen wie »telnet« und »ftp« entwickelt – Applikationen, die den Zugriff auf entfernte Rechnersysteme ermöglichen.

Im Jahre 1973 wurden die ersten Rechner aus Europa ans ARPA-Netz angeschlossen. 1974 umfasste das Netzwerk bereits 62 Rechner, allesamt in einem wissenschaftlich-militärischen Forschungsverbund zusammengeschlossen. Etwa 1978 war die Entwicklung des TCP/IP-Protokolls, also die »Netzwerksprache« der Computer untereinander, weitgehend abgeschlossen und es präsentierte sich bereits ungefähr in der Form, wie wir es heute kennen.

Je mehr Gruppierungen sich für die elektronische Datenübertragung interessierten, desto mehr entstanden unterschiedliche Ausprägungen des ursprünglichen ARPA-Netzwerks: Wissenschaftler verwendeten das CSNET (Computer Science Network), Studenten das USENET (USErs' NETwork), IBM Mainframes das BITNET (»Because It's Time«-Network – was zeigt, dass nicht jedes Akronym eine ernsthafte Bezeichnung sein muss), die europäische Gemeinde der Unix-Benutzer das EUNET (European Unix Network), das Militär das MILNET, die japanische Unix-Benutzergruppe das JUNET. Und damit auch die Benutzer von DOS-basierten Computern den Anschluss nicht verpassen, wurde das FIDONET gegründet (diese Bezeichnung beruht dabei nicht etwa auf einem Akronym: Tom Jennings entwarf 1983 eine Architektur, die es DOS-Rechnern gestattete, Mails, Diskussionen und Dateien über das Netzwerk miteinander auszutauschen. »Fido« war der Name von Jennings' Hund). Im Jahre Orwells, 1984, waren in all diesen – auf fast gleicher Technologie beruhenden – Netzwerken über 1000 Rechner miteinander verbunden.

Die National Science Foundation (NSF) fasste 1985 alle beschriebenen Netzwerke zu einem übergeordneten Netzwerk NFSNET zusammen und legte den Grundstein zu einem Internetbackbone (Backbone: engl. Rückgrat). Innerhalb von vier Jahren sollte sich die Zahl der beteiligten Rechner verachtzigfachen. Das ARPANET wurde aufgelöst und dessen Aufgaben wurden vom NFSNET übernommen, was immer mehr dem Begriff »Internet« gleichgesetzt wurde.

Ein solch großes Netzwerk, in dem man sich auf gemeinsame Standards einigen sollte, konnte im Laufe der Zeit nicht ohne steuernde Bürokratien auskommen. 1989 wurden demzufolge die Internet Engineering Task Force (IETF) und die Internet Research Task Force (IRTF) gebildet. Während sich die IETF mit der Einführung und Weiterentwicklung von Netzwerktechnologien, insbesondere der Spezifikation und Ausarbeitung von Internetprotokollen beschäftigt, plant und organisiert die IRTF Neuentwicklungen und Technologien im Rahmen des Internets.

Obwohl die Zahl der beteiligten Rechner immer schneller wuchs, wurde diese Technologie bis Anfang der neunziger Jahre fast nur im Bereich der Wissenschaft wahrgenommen. Die rechnerische Ausstattung fand sich vornehmlich an den Universitäten und Forschungseinrichtungen und der Datenaustausch beinhaltete in den meisten Fällen keine Informationen, die für den Massenmarkt von besonderem Interesse waren.

Und doch wurde hier im wissenschaftlichen Bereich der Grundstein für eine weltweite Revolution im Bereich der Telekommunikation und Datenübermittlung gelegt. Ausgangspunkt waren die Überlegungen eines Mitarbeiters am Europäischen Zentrum für Teilchenphysik, Tim Berners-Lee. Dieser befasste sich mit der Aufgabe, die vielen Datenmengen und Ergebnisse der verschiedenen physikalischen Experimente so zu archivieren, dass die Abhängigkeiten untereinander über Verweise auf die verschiedenen Dokumente umgesetzt wurden. Das sollte einen leichteren und direkteren Zugang zu den Daten ermöglichen. Das System sollte

- computergestützt und sowohl in Bezug auf die Hardware als auch auf das Betriebssystem plattformunabhängig sein,
- netzwerkbasiert ausgelegt sein, um den entfernten Datenzugriff zu unterstützen,
- erweiterbar sein, um jedermann die Möglichkeit zu geben, weitere Verweise auf eigene oder fremde Daten hinzuzufügen,
- textbasiert sein, um einen schnellen und direkten Zugriff auf die Daten zu erhalten, zugleich aber auch Möglichkeiten der Erweiterung bieten, um in einer späteren Version Grafiken zur Darstellung von Messergebnissen zu unterstützen,
- dezentral organisiert sein, um nicht einen einzigen monolithischen Wissensspeicher zu haben, sondern um die Informationen möglichst breit auf das gesamte Netz zu verteilen.

Dieses System, so stellte es sich Berners-Lee vor, sollte allen Forschungseinrichtungen zur Verfügung stehen und somit ein Informationsnetz (Web: engl. Netz) bilden, das stetig wächst und die Forschungsarbeiten unterstützt. Im Oktober 1990 wurde ein entsprechendes Proposal am CERN veröffentlicht [1].

Um diese Anforderungen zu unterstützen, musste(n)

- die Informationen im Klartext vorliegen, um nicht eine bestimmte Darstellungssoftware zu erzwingen,
- ein Zugangsprotokoll entwickelt werden, mit dem Daten von entfernten Rechnern über eine einfache Adressierung angesprochen werden konnten,
- eine Darstellungssoftware geschrieben werden, die die Darstellung der Links auf weitere Dokumente interpretieren und ausführen konnte.

Die Umsetzung sollte (wie bei anderen Architekturen bereits erfolgreich realisiert) auf Client/Server-Basis erfolgen, wobei der Anwender mit seiner Client-Software (Browser) auf die Dokumente des (Web-)Servers zugreifen kann.

Noch im gleichen Jahr wurde der erste Browser innerhalb des CERN getestet und Anfang 1991 stand dieser im Internet zum freien Download bereit. 1992 wurde der Browser Viola vorgestellt, der

- die meisten Betriebssysteme unterstützte,
- verschiedene Schriftarten interpretieren und darstellen konnte,
- Links erkannte und umsetzen konnte und
- eine Liste von Lesezeichen (Bookmarks) verwaltete.

Die Arbeit des CERN fand auf Forscherebene internationale Beachtung und die Entwicklungen in diese Richtung wurden vorangetrieben. So kam etwa zeitgleich mit Viola der Midas-Browser des Stanford Linear Accelerator Laboratory zum Einsatz, der etwa die gleiche Funktionalität wie der Viola-Browser hatte. Der eigentliche Internethype aber wurde mit dem Mosaic-Browser des National Center for Supercomputer Applications (NCSA) initiiert.

Der HTML-Standard wurde in der Version 1.0 verabschiedet und weiterentwickelt, so dass schon bald Tabellen, Eingabemasken und Dateiversionskontrollen unterstützt werden konnten. Ende Oktober 1993 zählte das Web etwa 200 Server, ein halbes Jahr später bereits über 600.

Als 1994 Mark Andreesen zusammen mit einer Reihe von weiteren Mitarbeitern das NCSA verließ, um mit einem eigenen Browser Netscape zu begründen, wurde das Internet zum neuen kommerziellen Kommunikationsmedium im ausgehenden zwanzigsten Jahrhundert.

2.2 Einfluss des Internets

Während das Internet in der dritten Welt noch eher eine untergeordnete Rolle spielt, entstehen in den modernen Industrieländern angesichts der neuen Informationsflut zunehmend Irritationen im Umgang mit diesen vielen Informationen. Manchen Staaten und Regierungen bereitet der Umstand, dass das Internet einerseits niemandem gehört und andererseits grenzenlos ist, große Kopfzerbrechen, scheitern sie doch an der scheinbar anarchistischen Struktur, die es jedem erlaubt, alles zu veröffentlichen, ohne Beschränkung durch ethische oder moralische Regeln. Unternehmen kämpfen gegen die Flut von Raubkopien ihrer Produkte und mancher Popstar sieht seine Existenz bereits durch Musik-Tauschbörsen wie Napster, Audiogalaxy oder Morpheus bedroht.

Auf der anderen Seite ist es aber gerade diese Form der Anarchie, die als Katalysator für Weiterentwicklungen in fast allen Bereichen wirkt:

- Die Produktpiraterie ähnelt einem Hase-und-Igel-Spiel zwischen Unternehmen und Anwendern. Das Ergebnis sind immer bessere und verfeinerte Schutzmechanismen auf der einen, effizientere Kompressions- und Verteilungsmethoden auf der anderen Seite, allesamt Entwicklungen, die der Weiterentwicklung der Medien zugute kommen.
- Hackerangriffe auf Unternehmen initiierten eine rasante Weiterentwicklung in den Bereichen Netzwerktechnologie und Kryptografie. Die daraus resultierenden Technologien können wiederum Banken für die Absicherung ihres Datenverkehrs untereinander wie mit dem Kunden einsetzen.
- Der plötzlich grenzenlose Vertrieb von Waren aus aller Welt führte zur Entstehung des elektronischen Kommerz mit vielen Facetten, von »Electronic Bill-Presentment« über »Online-Ordering« bis hin zu »eProcurement«.
- Um den Internetuser zu bewerben, bieten Portale Dienste an, die zu einer vollkommen neuen Form von Serviceleistungen führen.

Sicherlich wird diese Technologie eine Fülle von neuen Ideen auf den Markt bringen, die sich erst einmal bewähren müssen. So lässt sich darüber spekulieren, welchen Sinn es macht, wenn sich der Toaster mit dem Herd unterhält. Oder aber, ob ein Auto den Ausfall eines Teils automatisch der Vertragswerkstatt mitteilt, damit man dort ein Ersatzteil bestellen und einen Termin vereinbaren kann. Ob sich jeder wünschen würde, dass der Kühlschrank automatisch einen Bestellzettel an den Supermarkt mailt, mag dahingestellt sein. Könnte dieser aber aus den vorhandenen Lebensmitteln in Abhängigkeit des Verfallsdatums einen Menüvorschlag mit Rezepten aus dem Internet präsentieren, wäre dies vielleicht eine Killerapplikation wie seinerseits der erste Internetbrowser.

3 Directory-Services

Informationen sind nur so interessant, wie sich der Aufwand für ihre Beschaffung in Grenzen hält. Das kann Zeit, Geld, Tätigkeiten oder auch eine Kombination aus allem betreffen.

Zur Minimierung des Aufwands versucht man, die Daten zu strukturieren und sie in unkomplizierter Form dem Anwender anzubieten. Ein einfaches Beispiel ist ein Versandhauskatalog. Würden alle Artikel in ungeordneter Weise dargestellt werden, würden sich Kunden sehr schwer tun, die gewünschten Artikel zu finden und zu bestellen.

Um Daten strukturiert darzustellen, werden sie in Form eines ordnenden Systems oder einer Hierarchie hinterlegt. Diese Darstellung führt zur Entstehung von so genannten Verzeichnissen, die uns in unserem täglichen Leben – bewusst oder unbewusst – ständig begleiten.

- Telefonverzeichnis: eine Liste von Namen und Telefonnummern. Der erste Ordnungsfaktor ist die geografische Lage: Telefonbücher werden nach Städten oder Regionen ausgegeben. Weiterer ordnender Faktor sind Stadtteile und schließlich die alphabetische Anordnung der Nachnamen der Personen und Familien, die sich mit der zugeordneten Telefonnummer erreichen lassen.
- Fernsehzeitung: eine Übersicht über die Filme und Dokumentationen, die in naher Zukunft auf den einzelnen Fernsehkanälen gesendet werden. Der erste Ordnungsfaktor ist die Chronologie: Die Zeitschriften erscheinen jede Woche neu. Jede Zeitung ist nach Tagen geordnet, es folgt eine Gliederung nach Programmkanälen, diese wiederum ordnen sich chronologisch von früh morgens bis spät in die Nacht.
- Das Domain Name System des Internets ist ein geordnetes Verzeichnis, das den einzelnen Rechnern in einer Domain eine mnemonische Beschreibung und die entsprechende IP-Adresse zuordnet. Es folgt einer hierarchischen Ordnung, die mit ».« beginnt. Darunter finden sich die Toplevel-Domains wie ».com«, ».edu«, ».org« und alle länderspezifischen Domänen, gefolgt von den Unterdomänen »muenchen« oder »meinefirma«. Weitere Unterdomänen können folgen, den Abschluss bildet der Rechnername – »www«, »help« oder andere. Die Kontaktierung eines vernetzten Rechners erfolgt über die IP-Adresse, eine vierstellige Zahlenfolge, separiert durch den Punkt. Die Zuordnung zwischen Name und Adresse ist Aufgabe des DNS.

- Versandhauskataloge sind nach Produkten sortiert, die ihrerseits nach Farbe, Modestil oder Einsatzgebiet geordnet sind. Ein Inhaltsverzeichnis (nach Seitenzahlen geordnet) und ein Register (alphabetisch geordnet) helfen, die gesuchten Daten schnell zu finden.
- Suchmaschinen im Internet sind oft die einzige Möglichkeit, in der Fülle der Webseiten eine bestimmte Information zu finden. In diesem Sinne sind sie ebenfalls Verzeichnisse, die riesige Datenbanken beherbergen und mittels einer Suchmaske abgefragt werden können.

Allen Verzeichnissen gemeinsam ist, dass sie die Informationen sammeln, aufbereiten und in Form von Informationsdiensten Interessenten – Menschen oder Applikationen – zur Verfügung stellen. Für die Anwender ist es wichtig, dass die Benutzung einfach ist und der Dienst ständig zur Verfügung steht. Weiteres wichtiges Merkmal ist, dass der Zugriff keine aufwändigen oder proprietären Werkzeuge erfordert: Für eine Fernsehzeitung, Kataloge und die Telefonbücher muss man lesen können, das DNS ist standardisiert und für nahezu alle Betriebssysteme lesbar und die Suchmaschinen haben einfache *html*-Eingabemasken zum Formulieren der Suchanfrage. Die Aufbereitung der Daten in Form von hierarchischen Listen in Zusammenhang mit leichten Zugriffsmöglichkeiten trägt damit sehr zur Datenverfügbarkeit bei.

Directorys gehen einen Schritt weiter. Verzeichnisse haben meist einen sehr speziellen Anwendungsbereich: Telefonbücher dienen der Suche nach Telefonnummern, Fernsehzeitungen der Übersicht von TV-Programmen, Kataloge der Präsentation von Produkten. Weitere Verwendungen sind für diese Daten nicht vorgesehen. Directorys zielen dagegen auf mehr als eine Anwendergruppe. Betrachten wir wieder das Beispiel der Fernsehzeitung: Dem Directory-Ansatz zufolge würde dieses Verzeichnis als Applikation in Form eines elektronischen Services nicht nur die Namen, sondern auch die Frequenzen der Sender, Programmierdaten der Sendungen und die Beitragsart anbieten. So könnte sich ein Fernseher über diesen Dienst automatisch auf die Kanäle justieren und die Namen der Sender zuordnen. Der Videorecorder besorgt sich die Sendedaten über das »Video Programming System« (VPS), die Stereoanlage orientiert sich anhand des Attributs »Beitragsart«, ob es sich bei der Sendung um ein Sportevent, eine Seifenoper oder ein Rockkonzert handelt, und optimiert jeweils die Tonqualität. Und der Anwender kann sich über ein so genanntes »User-Interface« über den Programmverlauf der einzelnen Kanäle informieren.

Natürlich geht die Implementierung eines solchen TV-Directory weit über die Bedürfnisse vieler Menschen hinaus. In anderen Bereichen ist der Ansatz eines Directory aber sehr hilfreich, gerade wenn man feststellt, dass Daten redundant vorgehalten werden. Wir betrachten im Folgenden die Benutzerdaten und deren Verwendung im Netzwerk.

3.1 Szenarios verteilter Benutzerdaten

In vielen Unternehmen benötigen die Mitarbeiter für den Zutritt zu bestimmten Büroräumen eine Karte, um sich auszuweisen – eine so genannte »Badge«. Über ein Lesegerät erhält man mittels dieser Badge die Zutrittsautorisierung. Automatische Türöffner erkennen die Daten auf der Karte und entscheiden darüber, welchen Personen der Zutritt in die geschützten Räume gewährt wird. Ein neuer Mitarbeiter ist möglicherweise zwar aufgrund seiner Unternehmenszugehörigkeit berechtigt, die Räume zu betreten, kann sich aber unter Umständen aufgrund der noch fehlenden Badge dem Türöffnersystem gegenüber nicht authentifizieren. Und selbst mit vorhandener Ausweiskarte kann es zu Problemen kommen, wenn das System den neuen Mitarbeiter noch nicht im Datenstamm eingetragen hat. Ein Administrator hat dafür zu sorgen, dass dies in korrekter Form geschieht. Er gibt die Benutzerdaten wie beispielsweise Namen, Gruppenzugehörigkeit oder Karten-ID in den Datenstamm des Lesegeräts ein und erst dann gelangt der Mitarbeiter in seine neuen Räumlichkeiten.

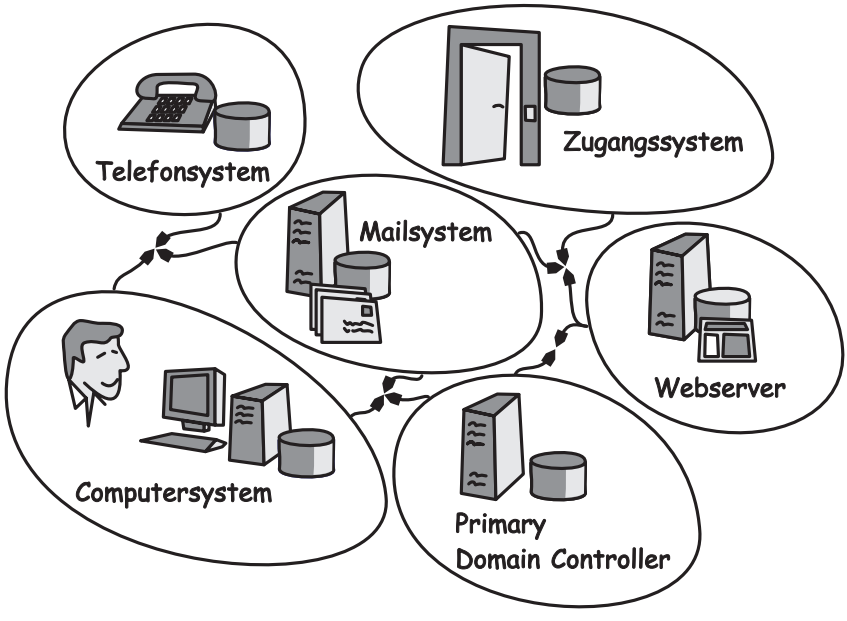


Abbildung 3.1: Verteilte Benutzerdaten-Haltung

Hier trifft er vielleicht auf die nächste Hürde: Die Badge wird bei der Zeiterfassung noch nicht erkannt. Auf seinem Schreibtisch steht ein neuer Rechner, der eine Authentifizierung mittels User-ID und Passwort verlangt. Daneben liegt eine ausgedruckte Telefonliste, die aber seinen Namen noch nicht enthält. Ein Kollege lässt ihn nun über einen allgemeinen Gast-Account in das System, ein Blick auf das webbasierte Telefonbuch verrät ihm aber, dass er auch hier noch nicht eingetragen ist. Auf seinem Tisch steht ein Telefon, dessen Display noch den Namen des Vorgängers anzeigt. Eine Mail an

den Administrator scheitert bereits im Ansatz, da noch kein Mailaccount angelegt ist. Das Netzwerk-Filesystem rückt keine Daten an Gast-Accounts heraus und gegen Mittag muss sich unser neuer Mitarbeiter mit einem Brötchen begnügen, weil ihm die Kantine noch keine Essenskarte ausgestellt hat.

Sicher, das ist ein etwas überspitztes Szenario, es macht aber auch ein grundlegendes Problem deutlich, das gerade in mittleren und großen Unternehmen häufig auftritt. Jede Applikation besitzt ihre eigenen Benutzerdaten: Das Telefonbuch muss aktualisiert werden, das Badge-System ebenfalls und bis hin zur Kantine werden die gleichen oder verwandte Informationen übermittelt, um einen geregelten Arbeitsablauf zu ermöglichen.

3.2 Datenredundanzen

Viele Applikationen, die in ein Umfeld neu eingegliedert werden, benötigen zur Verrichtung ihrer Aufgaben Benutzerdaten. Mailserver, Telefonverzeichnisse, Adress- und Nummernlisten auf Handys und PDAs, Computer und Türsysteme, all das sind Beispiele für Dienste, die auf Basis von Benutzerdaten funktionieren. Entsprechend hat jedes System einen eigenen Datenspeicher, der diese Informationen aufnimmt.

Doch gerade diese vielen verschiedenen Anwendungen bzw. Dienste und die erforderliche Datenhaltung bedeuten einen nicht unerheblichen administrativen Aufwand. Namen und Adressen, Telefonnummern und Passwörter werden immer wieder neu eingetragen und müssen gepflegt werden. Für einen neuen Mitarbeiter werden bei n verschiedenen Diensten die gleichen Benutzerdaten eingepflegt, kommt eine neue Applikation hinzu, müssen $n+1$ Datenstämme administriert werden.

Das Problem der Redundanz betrifft dabei nicht nur die Neuanlage von Benutzerdaten: Ändert sich beispielsweise ein Datum wie die Telefonnummer, so müssen $m \leq n$ betroffene Dienste verwaltet werden. Mit etwas Glück werden alle zu aktualisierenden Applikationen mittelfristig die neuen Daten erhalten, normalerweise sind aber nur ein Teil der Dienste sofort überarbeitet und alle anderen arbeiten mit veralteten Daten weiter. Dies erhöht nicht gerade die Verlässlichkeit der Benutzerinformationen. Sicherlich hat man sich im normalen Arbeitsumfeld in manchen Bereichen mit diesen Situationen arrangiert. Wird eine Mitarbeiterin, nachdem sie geheiratet und einen anderen Namen angenommen hat, für einige Zeit noch mit ihrem alten Namen geführt, so werden zunächst die Telefonlisten per Kugelschreiber korrigiert. Dies mag für eine Weile nicht unternehmenskritisch sein. Doch wo hört eine fehlertolerante Benutzerdatenverteilung auf und wo fängt eine konsistente Datenhaltung an, die verlässliche Daten unter allen unkritischen und kritischen Umständen liefert?

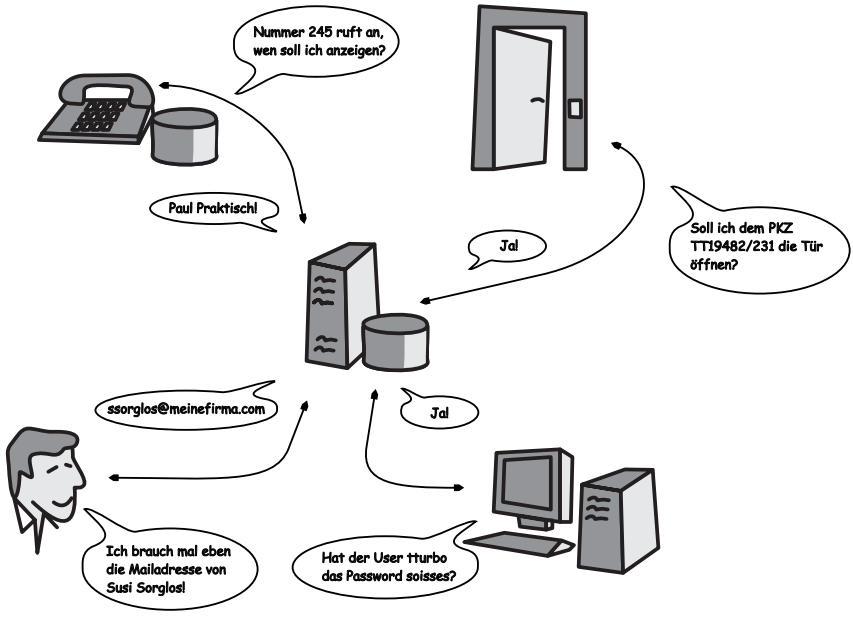


Abbildung 3.2: Zentralisiertes Benutzerverzeichnis

Datenredundanzen sind nicht die einzige Problematik in einem technologischen Umfeld mit vielen Anwendern. Schwierigkeiten bereitet häufig auch die Verwaltung der Datenvielfalt: Benutzer haben eine Reihe von Accounts auf den unterschiedlichen Applikationen. Der einzelne Anwender kann sich die vielen verschiedenen Mailadressen, Telefonnummern, User-IDs und insbesondere Passwörter oft nicht mehr merken. Spätestens hier betritt man den Bereich der Sicherheitsproblematik: Angesichts der Vielzahl von wichtigen und unwichtigen Daten fühlt sich mancher Anwender überfordert. Als Ausweg wird das Passwort für den Computerzugang in der Brieftasche mitgeführt, der Zugriffsschutz für den Mailaccount findet sich auf einem PostIt unter der Tastatur und der Zugriffscode auf die Files im Netzwerk ist auf dem Notizblock notiert. Hinzu kommen die mangelnde Bereitschaft der Benutzer, kryptische Passwörter zu wählen, und der Aufwand, die Passwörter auf allen relevanten Systemen zu aktualisieren. Ein solcher Umgang mit zum Teil unternehmenskritischen Daten stellt eine massive Gefährdung der Sicherheit des Gesamtsystems dar.

Ein zentraler Dienst, der diese Daten verwaltet und bei Bedarf verteilt, kann diese Problematik der verteilten Daten entschärfen. Der Name einer Mitarbeiterin, die nach den Flitterwochen wieder zum Dienst erscheint, wird an dieser zentralen Stelle ein einziges Mal aktualisiert, der Verteilungsmechanismus sorgt dafür, dass das Telefon auf dem Display automatisch den richtigen Namen anzeigt, das Badge-System den korrekten Eintrag in die Protokolldateien vornimmt, der Mailaccount automatisch mit dem richtigen Namen verknüpft wird und das Zeiterfassungssystem ordnungsgemäß Arbeitsbeginn und -ende der Mitarbeiterin protokolliert.

Das Ganze hat auch Vorteile bezüglich der Sicherheitsproblematik: Der Benutzer verwendet für sämtliche Applikationen nur ein einziges Passwort. Er muss sich nicht mehr viele verschiedene Passwörter auf Hilfsmitteln notieren, die in falsche Hände gelangen können.

Verteilte Benutzerdaten sind schwerer zu administrieren als zentralisierte. Dies ergibt sich aus dem Aufwand, alle Daten zu aktualisieren und keine zu vergessen.

3.3 Historische Ansätze

So genannte Directorys (Verzeichnisse) existieren nicht erst seit dem digitalen Zeitalter, sondern bereits seit Erfindung der Schrift. Kataloge und Warenlisten sind seit Tausenden von Jahren Bestandteil der Kulturen und helfen, Informationen zu ordnen und im Bedarfsfall bereitzustellen.

Elektronische Verzeichnisse von Benutzerdaten entstanden zeitgleich mit der Entwicklung des Computers. Für die Authentifizierung gegenüber einem Rechnersystem wird beispielsweise eine Liste von User-IDs und Kennwörtern geführt. Mit der Entwicklung des Computers als Kommunikationsmittel wurden den Verzeichnissen weitere Benutzerdaten wie Namen, Adressen und Telefonnummern hinzugefügt. Je mehr Daten aber katalogisiert werden, desto häufiger entstehen Redundanzen in den verschiedenen Diensten. Die gleichen Informationen werden mehr als einmal gepflegt und erfordern mehr und mehr administrativen Aufwand. Mit dem Aufkommen von digitalen Netzwerken erkannte man den Bedarf eines zentralisierten Directory, das über ein standardisiertes Kommunikationsprotokoll häufig angeforderte Daten zur Verfügung stellt. Auf diese Weise lassen sich Redundanzen einschränken.

Innerhalb von Unix-Netzen entstanden die so genannten »Yellow Pages«, das Network Information System (NIS) von Sun Microsystems. Mit NIS werden Benutzer, Netzwerk und Dienste katalogisiert. Die Motivation war, Administratoren einen Dienst an die Hand zu geben, der Konfigurationsdaten zentral gesteuert über das Netzwerk verteilt. Zusammen mit dem Network File System (NFS) sollten die Grenzen des Rechners auf das Netzwerk erweitert werden. NIS katalogisiert Metadaten, wie Benutzer-, Netzwerk- und Dienstezuordnungen, ist aber auf Unix-Systeme beschränkt und birgt eine Reihe von Sicherheitsproblemen.

Mit NDS (NetWare Directory Services) brachte Novell einen der ersten Nameserver auf den Markt, der die Zugriffe auf ein Novell-Netware-Netzwerk regelt. Das NDS ist ein Namensdienst, der einen globalen Zugriff auf alle Netzwerkressourcen unabhängig von deren physischen Standort ermöglicht. Die im NDS eingepflegten Benutzer melden sich nur einmal in einem Novell-Netware-System an und sehen das gesamte Netzwerk als ein einziges Informations- und Datensystem.

Mit diesem Dienst gelang ein weiterer Schritt in Richtung zentralisiertes Datenmanagement. Allerdings bestand immer noch eine Fokussierung auf eine bestimmte Applikation, die ihre Informationen zwar innerhalb des vom Hersteller vorgegeben Rahmens verteilt, aber über diese proprietären Grenzen hinaus keine Daten mit anderen Anwendungen teilte. Der Erfolg eines zentralisierten Verzeichnisses basiert jedoch sowohl auf der Plattform- als auch auf der Applikationsunabhängigkeit.

Mitte der 80er Jahre begannen unabhängig voneinander die »International Telecommunications Union« (ITU) und die »International Organization for Standardization« (ISO) mit der Entwicklung eines applikations- und plattformübergreifenden Directory. Beider Bestrebungen mündeten in die Entwicklung der X.500-Spezifikation, die bis heute immer noch fortwährend aktualisiert und verbessert wird. Eine umfangreiche Liste an Dokumentationen legt fest, wie die Implementierung und die Kommunikation mit einem solchen zentralen Benutzerverzeichnis auszusehen hat.

Eine der wesentlichen Innovationen, die X.500 mit sich brachte, war die Bereitstellung eines ersten, wirklich standardisierten, öffentlich verfügbaren Directory, das nicht an eine bestimmte Applikation gebunden war. Der Ansprüche an die Architektur waren enorm hoch: Es sollte ein weltweites Verzeichnis eingeführt werden, in dem man netzwerkbasiert alles und jeden katalogisieren kann. Die Datenstruktur ist hierarchisch aufgebaut und zahlreiche Vorschriften und Regeln legen die Kommunikation mit dem Verzeichnis fest.

Der globalisierte Ansatz und das damit verbundene umfangreiche Regelwerk zusammen mit der ausschließlichen Implementierung auf dem OSI-Netzwerkprotokoll machten X.500 allerdings von Beginn an recht unflexibel, was dem Erfolg dieser Architektur von Anfang an im Weg stand.

Mitte der neunziger Jahre, etwa zeitgleich mit dem Aufkommen kommerzieller Websites, kam eine neue Form des zentralen Benutzerdienstes zum Einsatz: der *ldap*-basierte Directory Server (*ldap*: »Lightweight Directory Access Protocol«). Die Grundarchitektur von X.500 bildete die Basis dieses Dienstes. Allerdings verzichtete man auf viele Einschränkungen und umfangreichen Protokoll-Overhead, um eine »leichtgewichtige« Form dieses Service zu definieren. Der *ldap*-Directory-Dienst trat seinen Dienst an. Heute werden immer noch X.500-basierte Verzeichnisse eingesetzt, die sich mittlerweile auch auf TCP/IP-Netzwerken implementieren lassen, doch *ldap* hat dem Vorläufer deutlich den Rang abgelaufen.

Mit der Entwicklung eines *ldap*-basierten Verzeichnisdienstes verfolgte man das Ziel, einen Dienst bereitzustellen, der über TCP/IP mit einem schlanken Directory-Access-Protokoll erreichbar ist. Die ersten Versionen von *ldap*-Verzeichnissen waren lediglich »Aufsätze« auf ein bestehendes X.500-Directory. Es gab eine Schnittstelle, die über dieses schlanke Verzeichnisprotokoll das dahinter liegende Directory abfragen konnte. Den Clients gegenüber wurde *ldap* »gesprochen«, gegenüber der X.500-Datenbank kam dagegen *dap* (»Directory Access Protocol«) zum Einsatz. Über die reine Gateway-Funktio-

nalität des neuen Protokolls hinaus war der Kern der Architektur immer noch mit dem Overhead des X.500 behaftet.

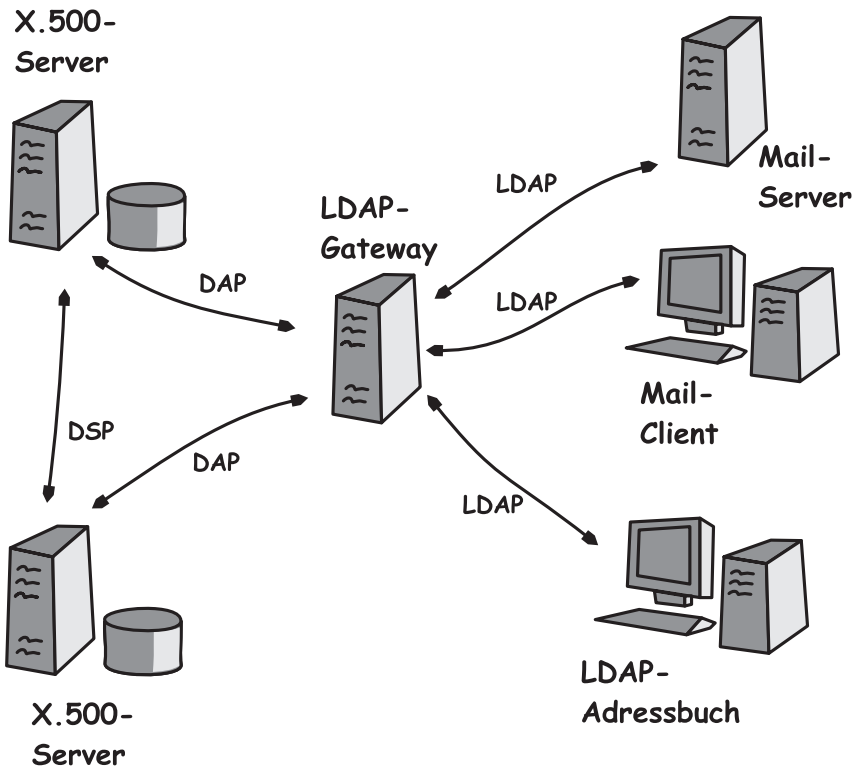


Abbildung 3.3: X.500-Implementierung mit LDAP-Gateway

Eine Mitarbeitergruppe von der Universität Michigan befasste sich intensiv mit der Weiterentwicklung des Dienstes zu einem eigenständigen Service. Ihr gelang Ende 1995 die Implementierung des ersten »Standalone *ldap* Daemon« (slapd), der auf einer eigenen Server-Architektur basierte.

Weitere Entwicklungen an eigenständigen *ldap*-Servern sind unter anderem:

- Novell Directory Service (NDS)
- Netscape/iPlanet Directory Server
- Innosoft Distributed Directory Server
- Sun Microsystems Directory Server
- Lucent Technologys Internet Directory Server
- Microsoft Active Directory

und natürlich die Weiterentwicklung des ersten slapd-Servers der University of Michigan, der in das Open *ldap*-Projekt mündete.

ldap bezeichnet das Protokoll, mit dem man Daten aus dem Directory liest. Das heißt, Clients erhalten eine »Sprache«, um den Inhalt eines Directory Servers abfragen zu können. Dieses Protokoll bietet query- und update-Funktionen zur Abfrage und Veränderung des Inhalts einer Directory-Datenbank. Service- und Replikationsfunktionen zwischen Servern untereinander sind in der originären *ldap*-Spezifikation nicht vorgesehen, vielmehr handelt es sich um ein reines Client/Server-Protokoll.

Für die Kommunikation zwischen den verschiedenen Directory-Servern sind in der X.500-Spezifikation klar definierte Protokolle vereinbart. Bei *ldap* existieren derzeit noch keine solche Standards. Beispielsweise sind die Kommunikationsprotokolle, die Replikationen von einem zentralen Server hin zu »gespiegelten« Servern erlauben, vielfach noch proprietäre Lösungen. Es existieren aber bereits die ersten Vorschläge für eine Übereinkunft, der Draft eines neuen Synchronisationsprotokolls mit dem Namen *ldup*. Das Akronym steht für »*ldap* Replication Update Protocol« und bezeichnet eine Kommunikation zwischen verschiedenen Directory Servern für den Datenaustausch im Sinne einer Replikation. Die *ldup*-Spezifikation liegt zurzeit (Stand 2002) nur als Standardisierungsvorschlag vor und ist noch nicht von der IETF in Form eines RFC freigegeben [5].

3.4 Abgrenzung gegen Datenbanken

Klassische Ansätze für eine geordnete Datenhaltung sind seit jeher Datenbanken. Warum hat man nun das Rad neu erfunden? Die Antwort ergibt sich aus neuen Anforderungen. Eine Datenbank ist transaktionsorientiert, das heißt, ein Datum wird erst dann für gültig erklärt, wenn alle damit verbundenen Datenänderungen abgeschlossen sind. Bei einem Directory Server werden die Änderungen sequenziell abgearbeitet. Eine einzelne Datenänderung wird entweder abgebrochen (die atomare Änderung ist damit nicht gültig) oder zu Ende durchgeführt (womit die atomare Änderung umgesetzt wird). Größere Transaktionen, wie sie bei korrelierten Daten in einer relationalen Datenbank vorkommen, sind im Directory Server dagegen nicht vorgesehen. Im Gegensatz zur transaktionsorientierten Arbeitsweise bezeichnet man diese Form des Datenänderungsmanagements beim Directory Server als operationsorientiert.

Eine weitere Einschränkung einer Datenbank ergibt sich aus dem Kommunikationsprotokoll. Eine Abfrage einer Datenbank wird meist in SQL formuliert. Diese »Query Language« ist allerdings nicht netzwerkfähig, insbesondere nicht über TCP/IP. Gerade dies aber wird von einem Verzeichnis erwartet. (Natürlich gibt es Client/Server-Architekturen, die SQL-Queries über das Netz austauschen. Diese setzen aber nicht auf ein natives Kommunikations- bzw. Übertragungsprotokoll auf, sondern stützen sich auf übergeordnete Applikationen, die die Datenübertragung umsetzen.)

Eine Datenbank ist weiterhin für lesenden und schreibenden Zugriff optimiert. Dies ergibt sich aus der Charakteristik der Daten, die sich relativ häufig ändern (volativ). Directory-Daten dagegen sind eher persistent, was in der Natur der Daten liegt, die in das Directory eingepflegt werden. Benutzerdaten sind überwiegend statisch (non volativ). Namen ändern sich bestenfalls einige Male im Leben, Adressen meistens nur alle paar Jahre. Gleiches gilt für Telefonnummern und E-Mail-Adressen. Am häufigsten ändern sich noch Passwörter oder Abwesenheitsmeldungen für den Autoreply auf Mails im Urlaub. Damit unterscheiden sich diese Daten deutlich von denen in einer relationalen Datenbank. Statistische Auswertungen ergaben, dass der lesende Zugriff im Vergleich zum schreibenden Zugriff in etwa im Verhältnis von 1000:1 steht. Pro tausend Abfragen auf den Directory Server erfolgt also etwa eine Datenänderung. Für viele Datenbanken gilt dagegen, dass etwa gleich viel lesende wie schreibende Zugriffe erfolgen.

Die Entscheidung für eine relationale Datenbank oder einen Directory Server hängt von der Datencharakteristik und den Applikationen ab, die auf diese Daten zugreifen:

- Webseiten, die einen Zugriff auf geschützte Daten nur nach einer Autorisierung durch User-ID und Passwort zulassen: Werden diese Daten über einen *ldap*-Server verwaltet und muss der Webserver eine nicht unerhebliche Zahl an Usern gleichzeitig bewältigen, so ist der Einsatz eines Directory Servers zu favorisieren, da Account-Daten in einem solchen Umfeld eher abgefragt als geändert werden.
- Kalender-Server, die Terminangaben sowohl liefern als auch entgegennehmen müssen, hinterlegen ihre Daten besser in einer Datenbank, da sich lesende und schreibende Zugriff wohl annähernd die Waage halten. Für solche Einsätze bietet ein Directory Server im schreibenden Zugriff keine ausreichende Performance.

Für Einsätze in einem Umfeld, in dem es auf die Bereitstellung von Daten ankommt, sind Directory Server von ihrer Architektur her optimiert. Sie sind in der Lage, die geforderten Daten schnell über ein netzbasiertes Kommunikationsprotokoll zu liefern. Schreibzugriffe sind möglich, stehen aber von der Priorität her hinter Lesezugriffen. Mit dieser Fokussierung sind Directory Server herkömmlichen Datenbanken im lesenden Zugriff etwa vier- bis fünffach überlegen.

3.5 Verzeichnis-Anwendungen

Ein rein directorybasierter Dienst sieht vor, dass über ein Directory-Access-Protokoll benutzerrelevante Informationen verteilt werden. Eine clientseitige Adressbuchapplikation beispielsweise sucht direkt am Directory Server nach Namen oder Adressen. Loggt sich ein User an seiner Mailbox ein, um seine E-Mails zu lesen, werden seine Authentifizierungsdaten vom Client-

Programm abgefragt und vom Mailserver entgegengenommen, der seinerseits diese Daten am Directory Server verifiziert. Das Zugangssystem zu Räumlichkeiten oder Gebäuden, das die Daten der Mitarbeiter über Badges überprüft, fragt an gleicher Stelle die Berechtigungsdaten ab. In allen Fällen verwenden die Applikationen gegenüber dem Directory das gleiche Kommunikationsprotokoll.

Mit Hilfe dieses Protokolls nimmt ein Directory Server Requests entgegen und liefert die Ergebnismenge. Der Anwender verwendet eine entsprechende Client-Software, um seine Abfrage zu formulieren und an den Server zu übergeben. Die Ergebnisse werden vom Client entgegengenommen, aufbereitet und dem Anwender dargestellt. Dabei werden die verwendeten *ldap*-Kommandos nicht im Klartext übermittelt. Es ist also beispielsweise nicht möglich, über Telnet auf den entsprechenden Serverport einen *ldap*-Befehl zu formulieren.

Für den direkten Zugriff auf *ldap*-Server gibt es eine Reihe von Client-Programmen, die mit mehr oder weniger komfortablen Oberflächen die Formulierung einer Abfrage ermöglichen und die Ergebnisse aufbereitet darstellen. Tools wie Outlook von Microsoft oder der Messenger von Netscape, beides Client-Programme, die auf die Verwaltung von E-Mails ausgerichtet sind, bieten die Möglichkeit, ein Directory als Quelle für Benutzerdaten anzugeben. Manche Produkte enthalten Eingabefelder mit so genannter Pinpoint-Funktionalität. Hierbei gibt man in einer Eingabemaske den vollständigen oder einen Teil des Namens der gesuchten Person ein. Der Client startet nach einer festgelegten Zeit von einigen Millisekunden automatisch eine Abfrage über das Directory und liefert das Ergebnis zurück an den Client. Der Benutzer sucht beispielsweise die Daten von Joe User und gibt »Joe« ein. Die Client-Software beginnt nach etwa einer Sekunde selbstständig mit der Durchführung einer Suche nach allen Einträgen, die den Begriff »Joe« enthalten.

Ein anderes Tool ist beispielsweise ein webbasiertes Telefonbuch, das der Benutzer verwenden kann, um Informationen aus dem Directory Server zu erfragen. Auf einer Intranetseite befindet sich eine Eingabemaske, in der der Anwender den Namen der Person einträgt, deren Telefonnummer, Adresse oder Faxnummer er benötigt. Die zugrunde liegende Applikation übernimmt den Eintrag, generiert einen *ldap*-Query und erfragt die gewünschten Daten im Directory.

In den Unternehmen reicht die Bandbreite der eingesetzten *ldap*-Clients von fertig konfigurierten und mit grafischen Benutzeroberflächen versehenen Programmen bis hin zu kommandozeilenbasierten Befehlen wie beispielsweise »ldapsearch«. Allen gemeinsam ist, dass sie dem Benutzer eine Möglichkeit geben, eine Anfrage zu definieren, die die Anwendung in einen verzeichnislesbaren Request übersetzt. Die Ergebnismenge wird mehr oder weniger aufbereitet dem Benutzer dargestellt.

Aber nicht nur User verwenden einen *ldap*-Client zur Abfrage von Benutzerdaten. Viele Prozesse oder andere Dienste greifen auf diese User-Daten zurück. Beispiel: Ein Mailserver erhält eine Mail. Eine der ersten Aufgaben

dieses Dienstes ist es herauszufinden, ob der Adressat der Mail in dem vom Mailserver verwalteten Netzwerk vorhanden ist. Beim Eintreffen einer Mail an »joe.user@meinefirma.com« muss also überprüft werden, ob Joe User überhaupt einen Mailaccount im Netzwerk besitzt. Diese Information liefert der Directory Server. Existiert dieser Benutzer nun unter der angegebenen Mailadresse, so überprüft der Mailserver weiter, ob er die Nachricht für den Benutzer lokal bereitstellen oder an einen anderen Server weiterleiten soll. Auch diese Daten sind im Benutzerverzeichnis hinterlegt und werden über *ldap* vom Mailserver erfragt. Loggt sich der Benutzer an seiner Mailbox ein, so werden seine Authentifizierungsdaten mit den Daten am Directory Server verglichen. Hat der Benutzer die korrekten Daten, also seine User-ID und das entsprechende Passwort, richtig eingegeben, so kann er seine Nachrichten lesen.

Directory-Clients können benutzer- oder applikationsgesteuert sein. Sie verwenden das gleiche Kommunikationsprotokoll.

Andere Dienste steuern ihren Betrieb gleichermaßen über die Daten des Directory Servers: Webserver erlauben oder verwiegern den Zugriff auf geschützte Seiten, Proxyserver kontrollieren den Zugang einzelner User oder ganzer Gruppen auf das Internet, Radiusserver überprüfen die Einwahldaten von Telefonverbindungen und stellen über den Remote Access Service (RAS) die Einwahldienste bereit.

Eine ganze Reihe von Client-Prozessen lassen sich mit Hilfe eines Directory Servers mit User-Daten versorgen. Dadurch erübrigt sich die Notwendigkeit, jeder Applikation ihren eigenen Datensatz bereitzustellen. Der administrative Aufwand für die Pflege dieser Daten reduziert sich auf einen einzelnen Dienst. Voraussetzung ist, dass diese Applikationen in der Lage sind, mit dem Directory zu kommunizieren.

3.6 X.500-Verzeichnisse

3.6.1 Basisarchitekturen

Mit X.500 ist eine Architektur entstanden, die in ihren Grundzügen bis heute die Struktur von Directorys beeinflusst. Selbst wenn sich *ldap* mittlerweile weitgehend von der X.500-Basis emanzipiert hat und als eigenes System existiert, so sind immer noch Prinzipien, Bezeichnungen und Strukturen die gleichen wie bei X.500. Was aber genau bietet ein X.500-Dienst im Netz?

Wie ein *ldap*-Directory hat ein X.500-Verzeichnis die Aufgabe, Informationen in Form von Objekten zu speichern. Es repräsentiert einen Dienst, der in Form von »Gelben Seiten« die Daten von Benutzern, Dingen und Lokalisationen auf Netzwerkebene bereitstellt. Über Zugriffskontrollen werden die Informationen geschützt.

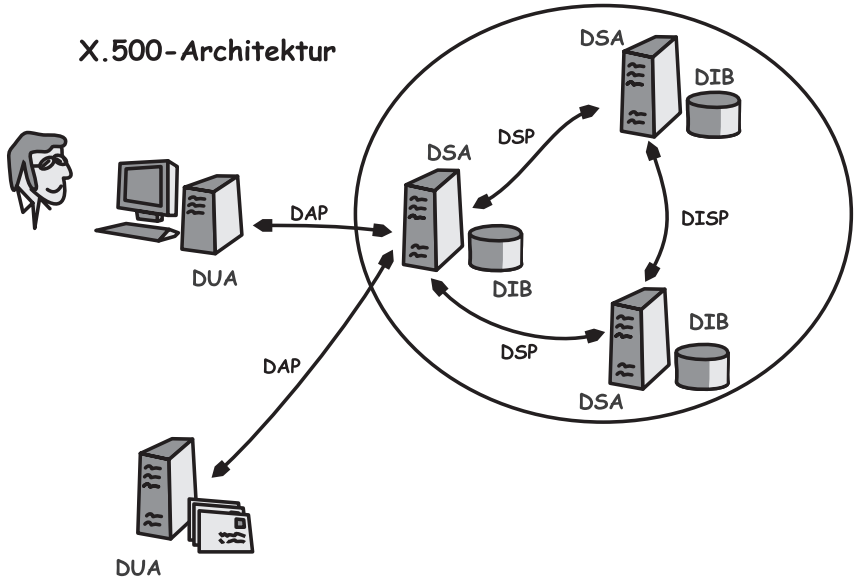


Abbildung 3.4: X.500-Architektur – Protokolle und Agents

Zum X.500-Wortschatz gehören folgende Bezeichnungen und Akronyme:

- Der Directory System Agent (DSA), der die Informationen im Verzeichnis verwaltet, Abfragen annimmt, bearbeitet und den Zugriff auf diese Daten steuert.
- Der Directory User Agent (DUA), der den Client-Zugriff auf das Verzeichnis regelt. Dabei handelt es sich um einen Client-Prozess, über den Directory-Anfragen formuliert und abgesetzt werden und der die Ergebnisse der Abfrage aufbereitet und dem Prozess oder Anwender zur Verfügung stellt.
- Die Directory Information Base (DIB), ein Dienst, der die Directory-Datenbank enthält und durch je einen DSA angesprochen wird.

Für die Kommunikation der Dienste untereinander und der Clients mit den Servern kommen die folgenden Protokolle zum Einsatz:

- Das Directory Access Protocol (*dap*), das die Kommunikation der Clients (DUAs) mit den Directory-Services (DSAs) bereitstellt.
- Das Directory Information Shadowing Protocol (*disp*), über das die DSAs der einzelnen Server untereinander Daten austauschen können, was eine verteilte replizierte Datenhaltung ermöglicht.
- Das Directory System Protocol (*dsp*), das verteilte Directory-Operationen ermöglicht. Dieses Protokoll ermöglicht den Zugriff auf Informationen in unterschiedlichen DSAs, indem die einzelnen Repositorys miteinander verlinkt werden.

- Das Directory Operations Protocol (*dop*), das die operationalen Beziehungen zwischen zusammenhängenden Verzeichnissen unterstützt.

Eine Zusammenfassung der beschriebenen Dienste und Protokolle ist in Abbildung 3.4 dargestellt.

Ein X.500-basiertes Verzeichnis kann in zwei Modi in einem Netzwerk von verschiedenen Servern integriert sein: Distribution und Replikation.

Bei der Distribution kann eine nationale Organisation ein unabhängiges Verzeichnis besitzen, das den zentralen Punkt für Directory-Abfragen darstellt. Ein privates Unternehmen besitzt und administriert daneben ein eigenes internes Verzeichnis, das als untergeordnetes Directory in das nationale Repository gelinkt wird. Bei dieser Architektur ist die Datenhaltung verteilt. Die Unternehmen behalten ihren Teil der DIB, sowohl die Administration als auch der »Besitz« der Daten sind dezentral.

Für den Fall, dass Daten über eine schmalbandige und teure Leitung transportiert werden müssen, ist eine lokale »Kopie« der Verzeichnisdaten erforderlich (Replikation). Diese Replikate werden über das *disp*-Protokoll verteilt. Replikate erfordern jedoch einen zusätzlichen Aufwand für Administration und Konsistenzprüfung. In der X.500-Terminologie unterscheidet man zwischen »shadowed« und »replicated« Directorys. Auf Ersteres darf nur lesend, auf das replizierte sowohl lesend als auch schreibend zugegriffen werden.

3.6.2 Protokolle im X.500-Umfeld

Der vorherige Abschnitt stellte bereits die verwendeten Protokolle vor. Diese sollen nun hier etwas detaillierter beschrieben werden.

- Das Directory Access Protocol (*dap*) wird ausschließlich zwischen dem Directory User Agent (DUA) und dem Directory System Agent (DSA) »gesprochen«. Der DUA ist immer der Erste, der diese Kommunikation initiiert, der DSA startet nie von sich aus die Verbindung. Über dieses Protokoll werden die Requests der DUAs wie die Ergebnisse der Anfrage übermittelt.

  Die Anfragen erlauben

  - die Suche nach Informationen, basierend auf einem bestimmten Filter – search. (Ich suche den Mitarbeiter Joe User. Wenn ein solcher Eintrag im Verzeichnis gefunden wurde, möchte ich wissen, wie dessen Telefonnummer lautet.)
  - den Vergleich von Informationen – compare. (Lautet die Mailadresse von Joe User »joe.user@meinefirma.com«?)
  - das Hinzufügen eines Eintrags – add entry. (Joe User hat ein Mobiltelefon bekommen, die neue Nummer soll ins Verzeichnis eingetragen werden.)

- die Änderung eines Eintrags – modify entry. (Joe User ist umgezogen und hat eine neue Adresse.)
- das Löschen eines Eintrags – delete entry. (Joe User verlässt die Firma und sein Eintrag wird im Unternehmensverzeichnis gelöscht.)
- das Lesen eines Eintrags – read entry. (Wie lautet die Adresse von Joe User?)

Dieses Protokoll ist das einzige Kommunikationsprotokoll zwischen Verzeichnis-Client und -Server.

- Das Directory Information Shadowing Protocol (*disp*). Zur Durchführung von Replikationen einzelner DIBs wird das *disp* verwendet. Damit werden die Daten von einem Server zum anderen übermittelt, das heißt, ganze Datensätze werden über dieses Protokoll auf andere Server kopiert.
- Das Directory System Protocol (*dsp*) ist das Protokoll, das zwischen den einzelnen Servern verwendet wird. Es entspricht im Wesentlichen dem *dap*, ist allerdings um einige Befehle erweitert, um die Verlinkungen der Server untereinander zu ermöglichen. Wird beispielsweise ein Request an ein lokales Directory geleitet, das nicht über die gewünschte Information verfügt, erfolgt mit dem *dsp* eine Weiterleitung der Anfrage an einen übergeordneten Server.
- Das Directory Operations Protocol (*dop*) dient zum Umsetzen von Vereinbarungen, beispielsweise Zugriffsrechte, Replikationsübereinkünfte und Informationen über die übermittelten Daten. Es ist ein rein administratives Protokoll, über das keine DIB-internen Daten ausgetauscht werden.

Die X.500-Verzeichnisarchitektur ist der Vorläufer der heute weiter verbreiteten *ldap*-Architektur. Sie hat eine mächtige und umfangreiche Konfigurations- und Befehlssammlung, die Implementierung der Daten erfolgt nach einem streng vorgegebenen Schema. Die Kommunikation der beteiligten Prozesse findet über fest vorgegebene Protokolle statt.

3.7 Datenhaltung

In heutigen Netzwerken hat die *ldap*-Architektur X.500 den Rang abgelaufen. Daher werden wir uns im Folgenden eng an die Beschreibung der *ldap*-Systeme halten, zumal eine große Ähnlichkeit in den Konzepten beider Architekturen besteht. X.500-relevante Unterschiede werden dabei herausgestellt.

3.7.1 Datendarstellung

Die Inbetriebnahme eines Directory Servers in einem Netzwerk erfordert neben der Installation eine Implementierung der bereitgestellten Daten. Gerade Letzteres stellt den Hauptaufwand bei der Einführung dieses Dienstes dar. Die späteren Directory-Daten müssen gesammelt, aufbereitet und dem Server zur Verfügung gestellt werden. Wie bei der Installation eines Betriebssystems sind nach einer Neuinstallation noch keine bzw. nur wenige Daten vorhanden. Je nach Hersteller einer solchen Software kann bereits ein Grundgerüst an Daten vorliegen, in denen die User-Informationen hinterlegt werden. Letztendlich ist aber die Implementierung der Datentopologie im Directory Server ein wichtiger Prozess der Installierung eines Directory-Services. Die Architektur der Daten beeinflusst die Möglichkeiten, Zugriffsrechte zu vergeben, die Performance zu optimieren und Änderungen durchzuführen. Eine ungünstige Topologie kann eine Skalierung auf große Datenmengen verhindern oder Replikationen erschweren.

Wie aber werden die Daten im Directory gespeichert? Eine allgemeingültige Antwort auf diese Frage gibt es nicht, da jeder Hersteller für die Hinterlegung der Daten in einer Datenbank ein eigenes Konzept verfolgt. Es existiert aber eine Übereinstimmung bezüglich der prinzipiellen Vorgehensweise. So sind Directory-Daten in einem so genannten Datenbaum hinterlegt. Dieser besitzt eine Wurzel, Zweige und Blätter. Damit wird eine Hierarchie eingerichtet, die die Verwaltung der Daten vereinfacht.

Einzelne Daten werden zu Einträgen zusammengefasst und in diesem Baum hinterlegt. Der einzelne Wert eines Datums wird in (Attribute/Wert-)Einheiten gespeichert, die Bezeichnung eines Attributs weist auf den Verwendungszweck hin. Wird ein neuer Benutzer mit seinem Namen eingetragen, erhält er beispielsweise ein Attribut mit der Bezeichnung »cn«, das den Wert <Name> zugewiesen bekommt:

```
cn: Joe User
```

»cn« ist der Attributbezeichner und steht für »common name«. (Laut Spezifikation ist der Wert des Attributs »cn« über »<Nachname> <Vorname>« definiert, in den meisten Fällen findet man aber die Reihenfolge »<Vorname> <Nachname>«.) Der Wert des Attributs ist »Joe User«. Der Sinngehalt eines solchen Eintrags liegt in der Verantwortung der Person, die die Daten einpflegt. So spielt es für den Server keine Rolle, ob das Attribut »cn« mit »Joe User« oder »eins-zwei-drei« belegt wird.

Die einzige Validierung, die ein Server durchführen kann, betrifft den Wertebereich. So kann der Server überprüfen, ob es sich bei dem eingetragenen Wert um einen alphanumerischen Wert handelt oder ob er ausschließlich Integer-Werte umfasst. Werden in der Datenvorbereitung Attribute mit alphanumerischen Werten belegt, für die laut Directory-Schema nur Zahlen vorgesehen sind, wird der Datenimport abgewiesen.

Die Implementierung der Daten folgt damit streng der Form

```
<Attribut>: <Wert>
```

Um nach allen eingetragenen Benutzern im Verzeichnis zu suchen, die mit dem Namen »Joe« anfangen, generiert man einen Request der Form

```
»Gib mir alle Benutzer aus, deren Name mit Joe beginnt«
```

oder für den Directory Server verständlich:

```
(cn=Joe*)
```

Für eine Abfrage lassen sich Verknüpfungen erzeugen, die einen Query präzisieren. Hierfür gibt es »UND«- bzw. »ODER«-Verknüpfungen.

Beispiel:

```
»Gib mir alle Benutzer, deren Name mit Joe beginnt und deren Adresse
den Substring ‚Starnberg' enthält«
```

oder:

```
(&(cn="*Joe*")(address="*starnberg*"))
```

Relationen dieser Art werden über »&« (für »UND«) und »|« (für »ODER«) umgesetzt.

Für die Namen der Attribute werden fest vorgegebene Bezeichnungen verwendet. Diese sind im so genannten Standardschema hinterlegt. So verwendet man für den Vor- und Zunamen »common name« oder »cn«, für die Adresse »address« usw. Man kann sich nun fragen, ob es Sinn machen würde, diese Bezeichner für eine lokalisierte Version des Directory Servers zu übersetzen. Die Konfiguration eines Servers lässt dies durchaus zu und erlaubt eine Erweiterung des Standardschemas, indem man seine eigenen Attribute einpflegt und diese für die Datenmodellierung verwendet. Dieses Vorgehen hat allerdings einen entscheidenden Nachteil: Fremde Clients, ob Anwendungen oder Personen, wissen zunächst nichts von den Schemaerweiterungen und würden zur Suche im Verzeichnis die Standardbezeichner verwenden. Die Wahrscheinlichkeit, dass sich der angepasste Attributname aus der Schemaerweiterung erraten lässt, ist eher gering. Bei Applikationen wie zum Beispiel Outlook oder Messenger sind die Suchkriterien im Programm unveränderbar vorgegeben. Ein solches Adressbuch-Programm verwendet beispielsweise folgende Befehlszeile

```
(|(mail=<Argument>)(cn=<Argument>))
```

zur Suche. Im Beispiel werden gemäß der *ldap*-Notation zwei Suchfilter mit einer »ODER«-Verknüpfung (»|«) kombiniert. Mit der folgenden Aufforderung sollen alle Einträge gefunden werden, deren Mailadresse bzw. Name mit »Joe« beginnt:

```
(|(mail=Joe*)(cn=Joe*))
```

Würde nun die Firma »meinefirma.com« für die Daten im Directory statt wie in der Standardimplementierung üblich

```
cn: Joe User
mail: joe.user@meinefirma.com
```

ausschließlich eine eigene Version der Art

```
vor_und_zuname: Joe User
elektronische_postadresse: joe.user@meinefirma.com
```

verwenden, so hätte dieses Adressbuch-Programm keine Möglichkeit, aus dem angepassten Verzeichnis die tatsächlich vorhandenen Daten zu lesen. (Einschränkung: eine Alias-Funktion innerhalb eines Directory, mit der man »cn« auf »vor_und_zuname« abbilden kann, was die »Lokalisierung« ermöglicht, ohne die standardisierte Schreibweise zu verlieren. Dies würde aber einen zusätzlichen Administrations-Overhead bedeuten.)

Um Konflikten mit Attributbezeichnern möglichst vorzubeugen, hat man sich daher im Directory-Standard auf festgelegte und oft verwendete Bezeichner geeinigt, die in vorgegebenen Objekten zum Einsatz kommen. Das entsprechende Regelwerk ist im Standardschema definiert und von der Internet Engineering Task Force festgelegt worden.

Mit dem Festlegen von Attributnamen allein ist die Datenorganisation in einem Directory noch nicht abgeschlossen. Vielmehr organisiert man die Benutzerdaten in Einträgen oder »Objekten«. Diese Objekte sind ebenfalls im Standardschema vordefiniert. Deren Umfang kann nach eigenen Anforderungen erweitert werden. Ein Directory-Eintrag bezeichnet eine logische Einheit, beispielsweise den Benutzerdatensatz einer Person oder einer Sache. Ein Benutzereintrag gruppiert somit alle personenbezogenen Daten einer Person. Zu diesem Zweck existieren vordefinierte Objekte im Directory, die diese Daten aufnehmen. So kann man beispielsweise ein vom Standardschema vorgegebenes Objekt »Person« verwenden, um den User »Joe User« zu beschreiben. Dieses Objekt nimmt dessen Daten auf, bekommt einen eindeutigen Schlüssel zugewiesen und wird in der Datenbank des Directory Servers hinterlegt.

Joe User könnte zum Beispiel wie folgt im Directory angelegt sein:

```
dn: uid=juser, ou=People, o=meinefirma.com
mail: joe.user@meinefirma.com
cn : Joe User
sn : User
givenname : Joe
telephonenumber: 089 1234567-123
ou: Professional Services
…
```

Der Schlüssel zum Datensatz von Joe User ist in der ersten Zeile hinterlegt, dem so genannten »Distinguished Name« (dn). Der Wert dieses Schlüssels ist »uid=juser, ou=People, o=meinefirma.com« und eindeutig im gesamten

Datenstamm des Directory. Die darauf folgenden Zeilen enthalten die im Directory hinterlegten Daten der Person Joe User.

Allgemein gesprochen liegen die Directory-Daten in Form von Einträgen vor, die einen Schlüsselwert in Form eines Distinguished Name beinhalten und darüber hinaus die Daten in Form von <Attribute>:<Value>-Paaren enthalten:

```
dn: <value distinguished name>
<attributename>: <value>
```

Der Distinguished Name ist eine organisatorische Information, die eindeutig über den gesamten Datenbestand zum jeweiligen Eintrag gehört. Dieser Wert dient der internen Verwaltung und wird nicht für die Informationsspeicherung verwendet. Damit ist es auch zunächst einmal unerheblich, ob man

```
dn: uid=juser, ou=People, o=meinefirma.com
```

oder

```
dn: mein erster Eintrag im Directory
```

verwendet, solange dieser Wert nur ein einziges Mal als Schlüsselinformation herangezogen wird. (Ein Directory Server wertet den Distinguished Name beim Einlesen in die Datenbank aus. Ein dn wie im letzten Beispiel hat keine gültige Struktur und wird abgewiesen. Dazu aber in Kürze mehr.)

Abbildung 3.5: Beispiel eines Directory-Eintrags

Für die weitere Organisation bedarf es einer Hierarchie, die es gestattet, aufgrund von geordneten Schlüsselinformationen schnell auf die Daten zuzugreifen. Diese Hierarchie wird in Form eines Baums (DIT: Directory Information Tree) realisiert.

Die Daten werden in Einträgen verwaltet. Jeder Eintrag hat einen Schlüssel, den so genannten Distinguished Name. Dieser Name ist eindeutig im gesamten Datenraum.

Die Daten werden in <Attribute>:<Value>-Paaren hinterlegt.

Die Datenwerte können vom Server auf Wertemengenbereiche festgelegt werden.

3.7.2 Directory Information Tree

Der Aufbau eines Directory Information Tree erinnert vom Aufbau her an ein Dateisystem mit Root, Ordnern und Dateien. Die Wurzel des Baums bezeichnet man beim DIT als Suffix (Dateisystem: »Root«), die Distinguished Names sind vergleichbar mit den absoluten Dateinamen (Dateisystem: kompletter Pfad und Dateiname) und der Inhalt eines Eintrags lässt sich mit dem einer Datei vergleichen.

Ein wesentlicher Unterschied zum Dateisystem ist, dass ein Ordner in einem Dateisystem selbst keine Daten enthalten kann, während in einem DIT alles, von der Wurzel angefangen über jeden Knoten bis hin zu den Blättern, Einträge der beschriebenen Form darstellt. Im vorliegenden Beispiel ist also der Eintrag mit dem dn »ou=People, o=meinefirma.com« gleichzeitig Container (für die User-Einträge darunter) und Datenträger. So kann man am Knoten »ou=People« die Mailadresse oder die zentrale Telefonnummer für alle Benutzer hinterlegen: mail: all@meinefirma.com, telephonenumber: 089 1234567-0.

Dateisystem und Directory Information Tree sind trotz zahlreicher Ähnlichkeiten zwei vollkommen unterschiedliche Systeme und haben – abgesehen von der vergleichbaren Topologie – nichts miteinander zu tun. Während das eine System zur Organisation von Daten auf einem Speichermedium dient (Dateisystem), beschreibt das andere die Organisation der Daten in einem Directory Server.

Schauen wir uns jedoch einmal die Struktur des Distinguished Name im vorangegangenen Beispiel etwas näher an.

Der Eintrag von Joe User enthält die Schlüsselinformation »uid=juser, ou=People, o=meinefirma.com«. Der allgemeinste Wert ist der der Organisation (o): »o=meinefirma.com«. Etwas spezieller ist »ou=People« (ou: Organization Unit). Die User-ID (»uid«) schließlich ist für den einzelnen User reserviert. Der Schlüssel für eine andere Person hätte die gleichen Werte für »o« und »ou« (»ou=Person, o=meinefirma.com«) und würde sich nur in der User-ID von den anderen Einträgen unterscheiden.

Weitere Beispiele:

- Angenommen, im Directory würden neben Personen auch Räume verwaltet werden, so würden diese gemäß unseres Beispiels unter »ou=rooms, o=meinefirma.com« hinterlegt werden. Der Schlüssel zum Konferenzraum mit dem Namen »Monterey« könnte somit lauten: »cn=monterey, ou=rooms, o=meinefirma.com«.
- Gruppen innerhalb der gleichen Organisation »o=meinefirma.com« werden unter dem Schlüssel »ou=groups, o=meinefirma.com« verwaltet. Die Gruppe der Volleyballer bekommt einen eigenen Eintrag mit dem Schlüssel »cn=volleyballer, ou=groups, o=meinefirma.com«.

Die Schlüsselinformationen dienen im Datenmodell zum Aufbau einer Datenhierarchie. Man entwickelt einen Datenbaum, der die Einträge anhand der Schlüsselwerte ordnet. So existiert eine »Wurzel« (oder im Directory-Sprachgebrauch der »Suffix«) des Baums, in unserem Beispiel ist dies »o=meinefirma.com«. Unter dieser Wurzel werden die weiteren Daten angeordnet. Wir haben die Zweige »ou=People« und »ou=Groups« verwendet, um die Daten von Personen und Gruppen einzurichten (Abbildung 3.6).

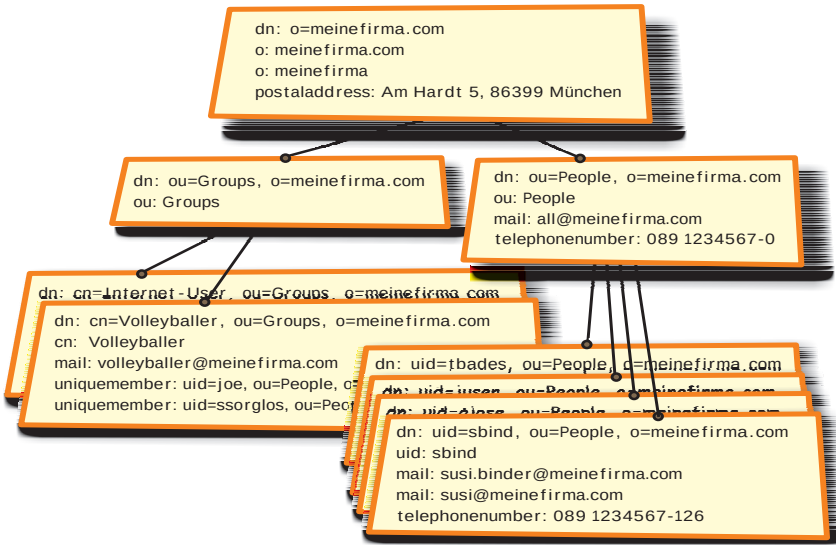


Abbildung 3.6: Beispiel eines DIT

Innerhalb des Werts für den Distinguished Name hat die Teilinformation, die unmittelbar hinter dem Bezeichner »dn:« steht, einen besonderen Namen. Dies ist der Wert des »Relative Distinguished Name« (rdn).

Beispiel: Im Schlüssel »uid=juser, ou=People, o=meinefirma.com« ist der Wert des rdn »uid=juser«.

Wird ein Directory Tree angelegt, so muss zunächst die Wurzel des Datenbaums vorhanden sein, bevor ein oder mehrere Zweige angelegt werden

können. Es kann immer nur eine Wurzel in einem Datenbaum existieren. Ein Blatt kann immer nur unter einem Zweig aufgehängt werden. Weiterhin ist das Anlegen von Blättern ohne den dafür vorgesehenen Zweig unzulässig, Importversuche von Einträgen dieser Art werden vom Directory Server abgewiesen.

Werden diese Implementierungsregeln für den Aufbau eines Directory-Datenstamms eingehalten, ist man unter *ldap* relativ frei in der Gestaltung des Baums. Eine Wurzel kann mit einer Organisation (o) beginnen, es können mehrere Organisationen untereinander angeordnet werden, Gleiches gilt beispielsweise für Organizational Units (ou). Somit ist es vollkommen legitim, eine Organisationsstruktur im Directory Server abzubilden, die einer realen Unternehmensstruktur ähnlich ist.

Beispiel:

Die Wurzel im Directory Server unserer Firma benennen wir mit »o=meinefirma.com«:

```
dn: o=meinefirma.com
```

Die verschiedenen Abteilungen werden als Organizational Units unter der Wurzel eingefügt:

```
dn: ou=IT-Abteilung, o=meinefirma.com
dn: ou=Marketing-Abteilung, o=meinefirma.com
```

Innerhalb dieser Abteilungen existieren Unterabteilungen, die sich im Directory wiederfinden können:

```
dn: ou=Professional Services, ou=IT-Abteilung,
    o=meinefirma.com
dn: ou=Sales, ou=IT-Abteilung, o=meinefirma.com
dn: ou=Suedeuropa, ou=Marketing-Abteilung,
    o=meinefirma.com
dn: ou=Nordeuropa, ou=Marketing-Abteilung,
    o=meinefirma.com
```

Wie beim Dateisystem muss die Einmaligkeit des Distinguished Name gewährleistet sein. Dies muss ein Directory Server beim Anlegen von neuen Daten stets überprüfen, da ansonsten die Datenbankeinträge nicht mehr konsistent sind. Wie bereits angesprochen, werden für die Konstruktion des Distinguished Name Attribute verwendet. Für das Erstellen des Verzeichnisses ist es unerheblich, welche Attribute man hierfür heranzieht. Es sollte aber ein einheitliches Verfahren angestrebt werden. Nimmt man beispielsweise den »cn« als Relative Distinguished Name, so ist dies vollkommen legitim:

```
dn: cn=joe user, ou=people, o=meinefirma.com
mail: joe.user@meinefirma.com
cn: Joe User
```

Stellt die Firma allerdings einen zweiten Mitarbeiter mit Namen »Joe User« ein, so kann dieser Wert des »cn« nicht mehr als Relative Distinguished

Name verwendet werden. Hier empfiehlt es sich, die User-ID (uid) zu verwenden, die im gesamten Netzwerk eindeutig ist.

Distinguished Names sind Schlüssel für Directory-Einträge. Ihre Struktur spiegelt die Position im Directory Information Tree wieder.

Ein Datenbaum in einem Directory Server baut sich über Einträge beginnend von der Wurzel bis hin zu den Blättern auf. Um ein Blatt einzufügen, bedarf es eines Zweigs, bevor ein Zweig implementiert wird, muss die Wurzel des Baums vorhanden sein.

Die für einen Eintrag verwendeten Attribute sollten dem Standardschema entnommen werden. Nur wenn sich die Directory-Information nicht über eines dieser Attribute beschreiben lässt, sollte man eine Schemaerweiterung in Erwägung ziehen.

Beispiel:

Angenommen, ein neuer Mitarbeiter, »Thomas Turbo«, soll mit einem neuen Eintrag im Directory aufgenommen werden.

Der Directory Server hat in seinen Konfigurationsdaten alle Attribute, die im Rahmen des *ldap*-Standards festgelegt wurden, implementiert. Sind vom einzelnen Unternehmen weitere Attribute definiert worden, sind diese im Rahmen der Schemaerweiterungen hinterlegt worden.

Für die Attribute des Namens möchten wir cn und sn verwenden, außerdem soll der Benutzer ein Passwort erhalten, mit dem er sich an den verschiedenen Applikationen anmelden kann. Nicht zu vergessen: das postaladdress-Attribut, in dem seine Privatadresse eingetragen werden soll.

Wir haben im Verzeichnis bereits einen Baum realisiert, der in Abbildung 3.6 dargestellt ist. Wir hängen demnach unseren Eintrag unter den Zweig »ou=people, o=meinefirma.com« und verwenden wieder die uid als Relative Distinguished Name:

```
dn: uid=tturbo, ou=people, o=meinefirma.com
postaladdress: Am Rotheck 4, 80995 München
cn: Thomas Turbo
sn: Turbo
userpassword: samba
```

(User-Passwörter werden im Directory Server üblicherweise nicht unverschlüsselt, sonder gehashed hinterlegt; dazu später mehr.)

Ein gültiger Directory-Eintrag besitzt demnach einen Schlüssel und die Datenfelder in Form von <Attribute>:<Wert>-Paaren. Diese Definition ist aber noch zu allgemein. Eine wesentliche Information fehlt, um einen gültigen Eintrag zu erzeugen: die Angabe der Objektklasse.

Objektklassen teilen dem Directory Server mit, um welche Art Eintrag es sich bei dem vorbereiteten Datensatz handelt, der in die Datenbank des Servers übernommen werden soll. So existiert beispielsweise eine Objektklasse »Person« mit einem Satz erlaubter Attribute, die man verwenden darf, wenn man eine Person über diese Objektklasse beschreiben möchte. Andere Attribute sind nicht erlaubt. Aus einem Eintrag wird ein gültiges Objekt, wenn dem Datensatz die Information über das Objekt mitgegeben wird. Jede Objektklasse hat einen bestimmten Satz von Attributen, die verwendet werden dürfen (oder die unbedingt erforderlich sind).

Directory-Objekte sind hierarchisch geordnet und leiten sich voneinander ab. Es existiert eine Superklasse, von der alle anderen Objekte (direkt oder indirekt) abgeleitet werden. Bei der Ableitung vererben die übergeordneten Klassen den darunter liegenden alle erlaubten Attribute. Die Superklasse hat den Namen »top«. Sie unterstützt lediglich zwei Attribute, »objectclass« und »aci«, und ist für die Bildung eines gültigen Directory-Eintrags unerlässlich.

Im Folgenden sehen Sie einen Auszug aus der *ldap*-Referenz des Standardschemas:

```
objectClass: top

Supported by ldap

Definition

Object class used as a superclass for all other object classes in the
directory. Reserved for use by the directory server.

superior Class: top

OID: 2.5.6.0.
```

Required Attributes	Description
ObjectClass	Defines the object classes for the entry.
Allowed Attributes	Description
aci	Stores the directory server access control information for this entry.

Für die Implementierung von personenorientierten Daten existiert eine vordefinierte Klasse »person«, die einen Satz von Attributen bereitstellt, mit deren Hilfe man User-Daten im Datenmodell hinterlegen kann. Sie leitet sich direkt von der Superklasse »top« ab.

Stellt man einen Eintrag einer Person für das Directory zusammen, müssen alle verwendeten Objekte im Eintrag angegeben werden.

Beispiel:

```
dn: uid=tturbo, ou=people, o=meinefirma.com
objectclass: top
objectclass: person
cn: Thomas Turbo
sn: Turbo
userpassword: samba
```

Mit der damit erlaubten Angabe »objectclass: person« wird ein Objekt erzeugt, das die Attribute »postaladdress«, »cn«, »sn«, »userpassword« usw. erlaubt. Reicht diese Attributmenge aus, um die Person ausreichend zu beschreiben, so sind keine weiteren Angaben zu den Objektklassen mehr notwendig.

Die Vorgaben der Objektklassen sind im Directory-Schema vorgegeben. Im Folgenden sehen Sie erneut einen Auszug aus der *ldap*-Referenz des Standardschemas:

```
objectclass: person

Supported by ldap

Definition

Defines entries that generically represent people. This object class
is the base class for the organizationalPerson object class.

superior Class: top

OID: 2.5.6.6
```

Required Attributes	**Description**
objectClass	Defines the object classes for the entry.
cn (commonName)	The person's common name.
sn (surName)	The person's surname, or last name.
Allowed Attributes	**Description**
description	Text description of the person.
seeAlso	URL to information relevant to the person.
telephoneNumber	The person's telephone number.
userPassword	Password with which the person can bind to the directory.

Diese Definition ist Bestandteil der Directory-Server-Konfiguration und darf im Sinne eines Standardschemas nicht geändert werden.

Muss den personenbezogenen Daten die postalische Adresse hinzugefügt werden, so lässt sich diese Anforderung mit der vorliegenden Objektdefinition nicht erfüllen. Vielmehr muss ein Objekt aus dem vorliegenden Konstrukt abgeleitet werden, das diese Angabe erlaubt. Im Standardschema ist dies bereits mit der Objektklasse »OrganizationalPerson« realisiert, die sich aus der Klasse »Person« ableitet. Verwendet man diese Objektklasse, so steht eine Vielzahl von weiteren Attributen zur Verfügung, auf die man für die Beschreibung der Daten zurückgreifen kann. (Anhang A fasst die im vorgestellten Beispiel verwendeten Objektklassen einschließlich der zugehörigen Attribute zusammen.)

Der Eintrag für den neuen Mitarbeiter sieht demnach folgendermaßen aus:

```
dn: uid=tturbo, ou=people, o=meinefirma.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
postaladdress: Am Rotheck 4, 80995 München
cn: Thomas Turbo
sn: Turbo
userPassword: samba
```

Betrachten Sie noch einmal die Struktur dieses Beispiels:

1. Der Distinguished Name ist der Schlüssel zu diesem Eintrag. Er umfasst im vorliegenden Beispiel drei Teilinformationen: »uid=tturbo«, »ou=people« und »o=meinefirma.com«. Diese Informationen wertet der Server bei der Hinterlegung dieses Eintrags in der Datenbank aus. Der Datensatz wird dabei unterhalb der Wurzel »o=meinefirma.com« und dem Zweig »ou=people« als »Datenblatt« mit dem Schlüssel »uid=tturbo, ou=people, o=meinefirma.com« im Datenbaum eingehängt.
2. Damit dies geschehen kann, muss der Eintrag als gültiger Datensatz validiert werden. Dazu werden die Objektklassen »top«, »person« und »organizationalperson« im Eintrag hinterlegt.
3. Anschließend ist die Angabe aller erlaubten Attribute im Rahmen der Objektklassendefinition möglich.

Die Form der Datenrepräsentation, wie sie der obige Eintrag zeigt, hat eine besondere Bezeichnung: »*ldap* Data Interchange Format« (LDIF). Über dieses Format ist es möglich, Daten in den Directory Server zu importieren. Gleichzeitig stellt dieses Format die ASCII-Darstellung der Daten dar, wie sie in binärer Form in der Directory-Datenbank hinterlegt ist. Viele Directory Server lassen den Export der Daten in dieses Format zu, womit man eine lesbare Form der Datenbank als Datei erhält.

Für den Datenimport in das Directory kann man eine Reihe von solchen LDIF-Einträgen zusammenfassen. Die verschiedenen Datensätze sind lediglich durch eine Leerzeile voneinander getrennt:

```
dn: uid=tturbo, ou=people, o=meinefirma.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
postalAddress: Am Rotheck 4, 80995 München
cn: Thomas Turbo
sn: Turbo
userPassword: samba

dn: uid=juser, ou=people, o=meinefirma.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
postalAddress: Hauptstrasse 3, 81673 München
cn: Joe User
sn: User
userPassword: geheim
```

Im Allgemeinen werden Migrationen über LDIF durchgeführt. Ein einfaches Script liest zeilenbasierte strukturierte Benutzerdaten in beliebiger Form ein, spaltet die Benutzerdaten ab, versieht sie mit dem korrekten Attribut und generiert dadurch ein LDIF-File, das dann vom Directory Server eingelesen wird.

Einträge in einem Directory Server werden in einer hierarchischen Struktur hinterlegt. Diese Struktur spiegelt sich in der des Distinguished Name wieder.

Jeder Eintrag wird über Objektklassen definiert, die in der Schemaimplementierung die zulässigen Attribute festlegen.

Directory-Einträge werden nur als gültige Objekte vom Directory Server importiert.

Die Festlegung der Attributnamen in Verbindung mit den Zuordnungen in Objektklassen führt zur weltweit gültigen Definition der Datenimplementierung in einem *ldap*-Server. Diese wird – wie beschrieben – als Standardschema bezeichnet. Mit Hilfe dieser Definition kann jeder Client – ob hart programmiert oder nicht – einen Directory Server auf Informationen bezüglich der Standardattribute abfragen. Möglich wird dies durch die gemeinsame Bezeichnung der Attribute und die standardisierte Verteilung der Attribute in vordefinierte Objektklassen.

3.7.3 Einschub: Verzeichnisbaum eines X.500-Directory

Bei der Planung der *ldap*-Architektur orientierte man sich stark an der X.500-Struktur. Diese sieht jedoch ein sehr viel strafferes Regelwerk für die Hierarchie der Daten vor. Darüber hinaus sind die Bezeichnungen etwas unterschiedlich.

Im X.500-Verzeichnis wird der Directory Information Tree (DIT) unter der Directory Information Base (DIB) platziert. Typischerweise werden diese mit dem »c=<Staat>« (c, Country) Ländereintrag belegt. Es kann die Lokalisation (l, Locality) folgen, anschließend die Organisation (o, Organization), die Organisationseinheit (ou, Organization Unit) und die Person (Organizational Person).

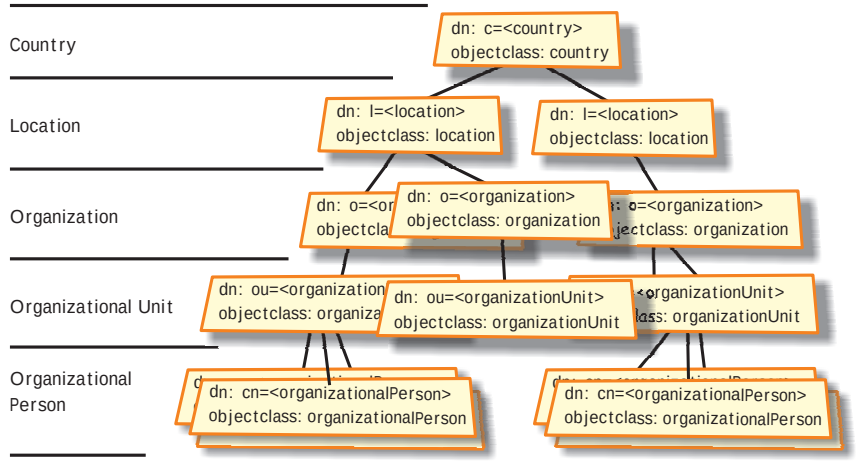


Abbildung 3.7: X.500-DIT

Diese Hierarchie der Einträge muss beim Design des DIT eingehalten werden. Es kann beispielsweise kein Country-Eintrag unter einem Organisationseintrag platziert werden.

3.7.4 Schemaerweiterungen in Directorys

Da sowohl in X.500 als auch in *ldap*-Verzeichnissen nicht alle Möglichkeiten der verschiedenen Organisationen und die Abdeckung in einem standardisierten Schema geboten werden können, gestattete man eine kundenspezifische Erweiterung des Standardschemas in der Implementierung der Server. Wir werden uns im Folgenden bei der Beschreibung der Architektur wieder auf den *ldap*-Standard beschränken, das Vorgehen bei X.500-Directorys ist aber weitgehend identisch.

Mit dem bisherigen Setup wurde erreicht, dass Benutzerinformationen, die in vielfältiger Form weltweit vorliegen, mit dem Standardschema in eine ein-

heitliche Struktur gebracht werden, was wie beschrieben die Grundlage einer system- und plattformübergreifende Kommunikation darstellt. Jede Applikation »kennt« die Liste der Attribute, die sie erwarten kann, wenn ein Directory Server die Objektklasse »organizationalPerson« für einen Eintrag verwendet.

Ein Directory Server hat das Standardschema implementiert und stellt damit eine Sammlung von Objektklassen zur Verfügung, die man beim Aufbau der Verzeichnisdatenbank verwendet. Es existieren Klassen für viele Formen von Einträgen, wie zum Beispiel für Personen, Geräte und Räume.

Gehen die eigenen Anforderungen aber über das vorgesehene Schema hinaus, muss ein Verzeichnisdienst dies berücksichtigen können. Realisierungen dieser Art werden als Schemaerweiterungen bezeichnet.

Angenommen, in einer Beispielfirma stellt die Schuhgröße der Mitarbeiter eine businessrelevante Information dar, die sich unbedingt im Directory wiederfinden muss. Leider ist in keiner einzigen personenbezogenen Objektklasse ein Attribut zu finden, das diese Daten sinnvoll aufnehmen kann. In diesem Fall muss eine Schemaerweiterung durchgeführt werden.

Die Vorgehensweise in Kurzform:

- Das fehlende Attribut wird im Directory Server in der Konfiguration als proprietäre Erweiterung angegeben.
- Bei der Implementierung der Daten orientiert man sich an der Objektklasse aus dem Standardschema, die die User-Daten zum größten Teil aufnehmen kann. Diese Objektklasse wird als Grundlage der Schemaerweiterung herangezogen.
- Eine neue, proprietäre Objektklasse wird von der vorherigen Objektklasse abgeleitet. Damit erbt die neue Klasse alle Attribute der Standardklasse.
- Das neue Attribut wird der neuen Objektklasse hinzugefügt.

Die einzelnen Schritte werden im Folgenden an diesem Beispiel erläutert.

Zunächst muss das neue Attribut definiert werden. Dies geschieht in der Konfiguration des Directory Servers. Das neu zu definierende Attribut soll »shoesize« heißen und in Form einer Schemaerweiterung eingeführt werden.

```
Shoesize
Schema extention
Definition
typically identifies the shoe size of a person.
Examples:
shoesize:     42
or
shoesize:     7A
syntax:       CIS - many
OID:          to be defined
```

Die Syntax dieses Attributs umfasst:

cis	»Case Ignore String«	Groß- und Kleinschreibung ist irrelevant
ces	»Case Exact String«	Groß- und Kleinschreibung ist relevant
bin	»Binary«	Binärdaten
tel	»Telephone Number String«	Leerzeichen, Bindestriche und Slashes irrelevant
dn	»Distinguished Name«	Leerzeichen irrelevant

Die Festlegung der Syntax eines Attributs schränkt somit den Wertebereich der Directory-Einträge ein. Damit erhöht sich die Datensicherheit und damit die Verlässlichkeit der Directory-Daten.

Neben der Syntax einer Information spielt die Wertigkeit eine große Rolle. Ein Attribut ist entweder »single-valued« oder »multi-valued«. Im ersten Fall darf es innerhalb eines Eintrags nur einmal benutzt werden, »multi-valued«-Attribute können auch mehrmals verwendet werden.

Beispiel:

Eine Personalstammnummer, die jeder Mitarbeiter des Unternehmens hat, wird als Information in das Directory aufgenommen. Diese Nummer ist eindeutig, jeder Mitarbeiter hat genau eine Nummer. Um zu verhindern, dass man im Betrieb des Servers irrtümlich eine zweite Personalstammnummer einpflegen kann, wird dieses Attribut als »single-valued« angegeben. Da der User aber mehrere Mailadressen haben kann, wird das »mail«-Attribut als »multi-valued« geführt.

Je nach Hersteller gibt es leicht unterschiedliche Umsetzungen der Attributdefinition in den Konfigurationsdaten. Das Prinzip ist allerdings das gleiche.

Im nächsten Schritt untersucht man die Liste der Objektklassen, die die Informationen der Mitarbeiter weitgehend aufnehmen kann. Beispiel ist die von Netscape eingeführte und mittlerweile standardisierte Objektklasse InetOrgPerson. Hier werden eine Vielzahl von Attributen zur Beschreibung eines Benutzers bereitgestellt.

Nachdem das Attribut »shoesize« nun definiert ist, erzeugt man eine neue Objektklasse – »meinefirmaperson« –, die von der Objektklasse »inetOrgPerson« abgeleitet ist und damit alle Attribute aus dieser Klasse erbt:

```
objectclass: meinefirmaperson

Schema extention

Definition

Defines entries that generically represent people at meinefirma. This
objectclass is a schemaextention and supports shoesize
```

```
superior Class:        inetorgperson
OID:                   to be defined
Required Attributes    Description
(attributes inherited by parental class)
Allowed Attributes     Description
(attributes inherited by parental class)
shoesize               Shoe size of a person
```

Diese Darstellung ist als Beispiel zu verstehen, die Implementierung variiert bei den unterschiedlichen Herstellern von Directory Services.

Mit der neuen Objektklasse können die User-Einträge um die Schuhgröße ergänzt werden. Dabei ist zu beachten, dass in einem Directory-Eintrag die Angabe der neuen Objektklasse erfolgt, bevor das neue Attribut verwendet wird. Andernfalls kann der Eintrag nicht ergänzt werden:

```
dn: uid=tturbo, ou=People, o=meinefirma.com
objectclass: top
objectclass: person
objectclass: organizationalperson
objectclass: inetorgperson
objectclass: meinefirmaperson
postaladdress: Am Rotheck 4, 80995 München
cn: Thomas Turbo
sn: Turbo
userpassword: samba
shoesize: 42
```

Abbildung 3.8: Directory-Eintrag einer Person mit einer Schemaerweiterung

Für das vorliegende Beispiel müssen die jeweiligen Benutzereinträge nicht neu geschrieben werden, sondern nur um die neue Objektklasse und das gewünschte Attribut (mit dessen Wert) ergänzt werden.

Die Schemaerweiterung geht über das Standardschema hinaus und ist nicht Bestandteil der internationalen Attributbezeichnungen. Damit kann nicht erwartet werden, dass Clients, die keine Kenntnis von der Schemaerweiterung haben, diese Attribute finden. Da aber, wie im Beispiel, die neue Objektklasse aus einer Standardklasse abgeleitet wurde, die bereits einen Großteil der User-Daten beinhaltet, sind zumindest diese Standarddaten »sichtbar«.

Mit Hilfe der Schemaerweiterung kann man jedes beliebige Attribut im Verzeichnis platzieren. Eine Applikation, die unsere shoesize nicht »kennt«, ist dennoch in der Lage, die übrigen Attribute abzufragen, und muss nicht eigens für unseren Datensatz angepasst werden.

3.7.5 Referrals

Wir haben Replikation bereits als Möglichkeit vorgestellt, wie man eine verteilte Directory-Topologie auf verschiedenen Directory-Servern realisieren kann. Dabei werden Daten von einem Server auf einen anderen kopiert. Eine andere Form der Verteilung ist über die Verwendung von Verweisen, so genannten Referrals, möglich. Hierbei handelt es sich um eine Verlinkung der Daten, eine Technik, wie man sie bereits aus dem Web-Umfeld kennt. Innerhalb eines DIT wird an einem bestimmten Knoten auf einen anderen Server verwiesen, der diese Daten (unterhalb des Knotens) bereitstellt. Im Gegensatz zur Replikation werden keine redundanten Daten erzeugt.

Abbildung 3.9: Übergeordneter DIT eines verteilten Directory

Beispiel:

Unsere Beispielfirma hat in München und New York eine Niederlassung und betreibt in beiden Standorten je einen Directory Server. Die Münchner Geschäftsstelle hält ihren eigenen Datenstamm im Verzeichnis vor und verlinkt bei Zugriffen auf den New Yorker Teil auf den dortigen Directory Server.

Der DIT ist wie in Abbildung 3.9 aufgebaut. Die Details der Verweise sind in Abbildung 3.10 und 3.11 dargestellt.

Referrals haben eine eigene Objektklasse und verweisen über *ldap*-URLs auf einen anderen Server. Eine Definition von *ldap*-URLs ist in 13.1.3 zu finden.

```
dn: l=europa, ou=People, o=meinefirma.com
objectclass: top
objectclass: referral
ref: ldap://ldap.europa.meinefirma.com/
  l=europa, ou=People, o=meinefirma.com
```

```
dn: l=amerika, ou=People, o=meinefirma.com
objectclass: top
objectclass: referral
ref: ldap://ldap.amerika.meinefirma.com/
  l=amerika, ou=People, o=meinefirma.com
```

Abbildung 3.10: Referral-Einträge im Directory Server New York (oben) und München (unten)

Erfolgen Suchabfragen über den europäischen Teilbaum, so sucht der Directory Server in München in seinem eigenen Datenstamm. Wird dort dagegen eine Suche über den amerikanischen Teil durchgeführt, verweist der Münchner Dienst auf den amerikanischen Server, indem er dem Client statt einer Ergebnismenge eine *ldap*-URL zurückgibt. Der Client wertet die URL aus und verbindet sich mit dem New Yorker Server, wo er die Suche fortsetzt.

Einer der Hauptnachteile von Referrals ist die Umsetzung der Umleitung durch den Client. Dieser muss die URL als solche erkennen, verstehen und auswerten. Obwohl bereits eine Reihe von Client-Applikationen Referrals unterstützen, kann dies noch nicht generell vorausgesetzt werden. Bei der Planung einer Directory-Topologie muss diese Einschränkung berücksichtigt werden.

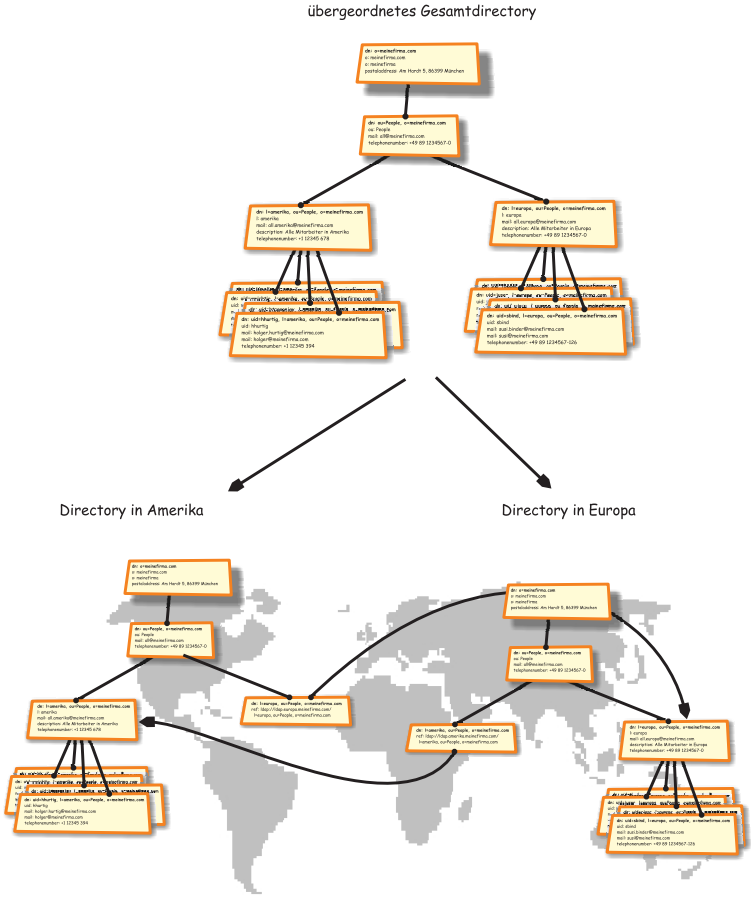


Abbildung 3.11: Datenbestände und Referrals auf zwei verteilten Directorys

3.7.6 Replikationstopologien

Replikationen und Skalierung

Ein Directory Server soll performant Daten liefern, unabhängig davon, ob in seiner Datenbank einige wenige User-Informationen oder viele Millionen Benutzereinträge enthalten sind. Diese Skalierung lässt sich über verschiedene Wege erreichen:

- horizontal über das Hinzufügen einer Maschine mit einem neuen Dienst, der in die Replikationstopologie eingebunden wird
- vertikal, indem eine Maschine über zusätzliche Hardware aufgerüstet wird

Während die vertikale Skalierung nicht oder nur sehr bedingt in die Architektur eines Directory Servers eingreift, basiert die horizontale Skalierung auf der kontrollierten Verteilung der Daten auf verschiedene Systeme. Kopien der Daten werden auf andere Directory Server verteilt, ohne dass der Vorteil der zentralen Benutzerverwaltung verloren geht. Entsprechende Mechanismen übernehmen dabei die Verteilung der Datenänderungen auf die Kopien, ohne dass es zu Konflikten in den Datensätzen kommt.

Einmal implementiert, stellen Replikationen eine einfache Möglichkeit dar, Kopien der Datensätze vorzuhalten. So können lokale Kopien in einer Geschäftsstelle des Unternehmens die geforderten Benutzerdaten sehr viel effizienter bereitstellen als bei einer Übertragung dieser Daten über eine Datenleitung von der Zentrale aus.

Hochverfügbarkeits-Szenarios verlassen sich ebenfalls auf die Mittel der Replikation: Es werden mehrere Server mit gleichem Datenstamm parallel aufgebaut und die Last der Anfragen wird über Loadbalancing-Mechanismen gleichmäßig auf alle Services verteilt. Fällt einer der Directory Server aus, führt dies nicht zu einem Totalausfall des Dienstes, da die verbleibenden Services die anfallende Last übernehmen.

Datenbesitz und Informationshoheit

Directory Server haben die Möglichkeit, Replikationen zu anderen Servern einzurichten. Dabei werden Teile der Datenbank oder der komplette Datensatz einschließlich der Zugriffsregularien an die Replikate kopiert. Dies ist insbesondere bei regional verteilten Zugriffsanforderungen von Vorteil.

Beispiel:

Ein Unternehmen hat in München seinen Hauptsitz und in Frankfurt und Hamburg Filialen. Die Mitarbeiter in den Filialen greifen entweder immer auf den einzigen Directory Server in München zu oder sie bekommen ein Replikat des Directory Servers an ihrem Standort eingerichtet. Dieser lokale Dienst ist ein vollwertiger Directory Service mit den gleichen User-Daten wie der Server am Hauptsitz.

Einen Problembereich der Replikation stellt die Verteilung der Datenänderungen dar. Hierfür gibt es unterschiedliche Strategien, die im Folgenden etwas näher analysiert werden.

Grundprinzip der Replikation ist es, Datenredundanzen aufzubauen, um einen performanteren Zugriff auf Daten zu bekommen. Man unterscheidet zwischen zentralisiertem und verteiltem Informationsbesitz. Ein Service »besitzt« Daten, wenn man hier und nur hier die Informationen ändern kann. Demgegenüber bezeichnet man Daten, die keine Zugehörigkeit zu einem Service aufweisen, als »besitzlos«.

Wie im täglichen Leben erfahren Dinge, die jemand besitzt, Pflege durch ihren Eigentümer, während Dinge ohne Besitzer eher vernachlässigt werden. Andererseits kann man auf Dinge, die sich in fremdem Besitz befinden, erst

nach Freigabe des Besitzers zugreifen, während sich besitzlose Sachen ohne große Nachfrage direkt nutzen lassen. In der Replikationspolitik verhält es sich ähnlich. Ein Service »besitzt« die Informationshoheit, das heißt, er hat die Hoheit über Änderungen an diesen Informationen. Daten, die unter keiner Hoheit stehen, können dagegen von jedem gelesen und verändert werden.

Im Folgenden wird ein Szenario beschrieben, in dem die Dienste ihre Informationen aus Performance-Gründen anderen Diensten zur Verfügung stellen, indem sie Kopien der Daten verteilen. Dies geht so lange gut, wie die Services sich untereinander abgleichen, das heißt die Änderungen direkt synchronisieren können. Geht der Synchronisationskanal verloren, kann es je nach Besitzverhältnis der Daten zu Kollisionen kommen.

Hier ein Beispiel einer Kollision:

In einem Directory-Netzwerk ist der Eintrag eines Anwenders mit dem Namen »Joe User« auf mehreren Services hinterlegt. Diese stehen nicht im direkten Abgleich miteinander, die Informationshoheit ist über alle Dienste verteilt.

In diesem Beispiel wird in Service A der Anwender gelöscht, während in Service B das User-Passwort dieses gleichen Anwenders geändert wird. Bei der Synchronisation entsteht ein Konflikt. Es muss nun entschieden werden, welche der Änderungen Gültigkeit hat bzw. in welcher Reihenfolge die Datenänderungen umgesetzt werden. Liegt der Zeitstempel von Service A vor dem von Service B, so liegt eine Dateninkonsistenz vor. Ein Administrator kann unter Umständen eine einzelne Inkonsistenz noch relativ leicht auflösen. Ein Service aber, der sich selbst organisieren muss und mit einer großen Anzahl von Datenänderungen konfrontiert ist, kann leicht Gefahr laufen, dass eine größere Summe von Dateninkonsistenzen zusammenkommt, die sich nur noch sehr schwer auflösen lassen.

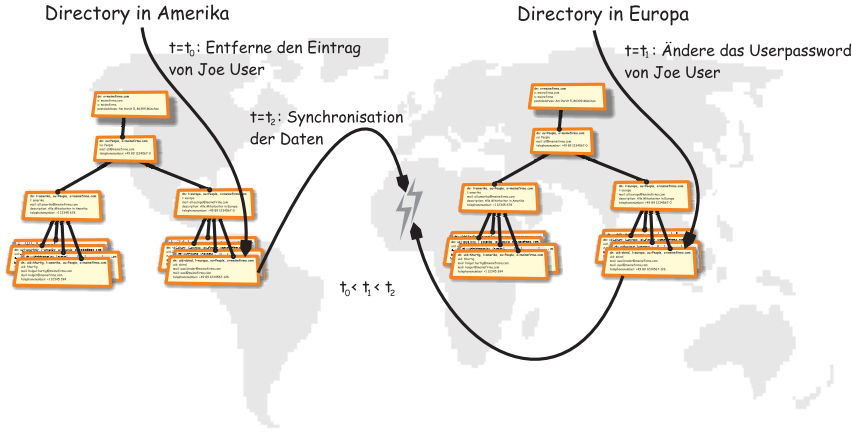


Abbildung 3.12: Beispiel einer Kollision bei einer Datensynchronisation

Die Probleme des vorgestellten Beispiels lassen sich auf die verteilte Informationshoheit zurückführen. Jeder Service kann beliebig Daten ändern, wobei entstehende Konflikte nicht immer aufgelöst werden können. Eine zentrale Informationshoheit eines einzigen Services würde diese Konflikte vermeiden. Dieser Server hat die Verantwortung für die Daten. Er stellt bereitwillig jedem Interessenten eine Kopie der Informationen zur Verfügung, gestattet es aber niemandem außer sich selbst, diese Daten zu ändern.

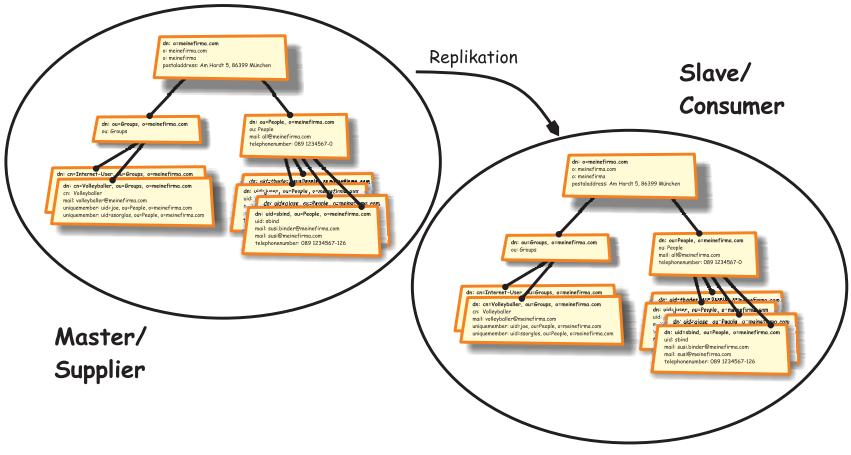


Abbildung 3.13: Single-Master Komplettreplikation

In einem solchen Gefüge besteht zu keiner Zeit die Möglichkeit, dass Änderungen ein und derselben Information kollidieren. Andererseits besteht die Gefahr, dass Änderungen bei einer Netzwerkstörung nicht mehr durchgeführt werden können. Der Datenbestand ist damit eingefroren und kann erst wieder modifiziert werden, wenn der »Eigentümer« der Daten wieder erreichbar ist.

Im Idealfall ist das zugrunde liegende Replikationsmodell selbst organisierend. Es verwaltet Änderungen ausfallsicher, ohne dass es zu Kollisionen in den Änderungen am Datensatz kommt, selbst wenn die Synchronisation über einen längeren Zeitraum nicht durchgeführt werden kann.

Im Folgenden orientieren wir uns an der zentralisierten Informationshoheit, da dies die vorherrschende Replikationsarchitektur in Directory-Implementierungen ist.

Befindet sich ein Datensatz im Besitz eines einzigen Services, so werden Änderungswünsche bezüglich dieser Daten, die an den Replikaten auflaufen, an diesen Service weitergeleitet. Dort werden die Änderungen umgesetzt und erst dann an die Replikate verteilt. Direkte Änderungen an den Replikaten sind nicht erlaubt.

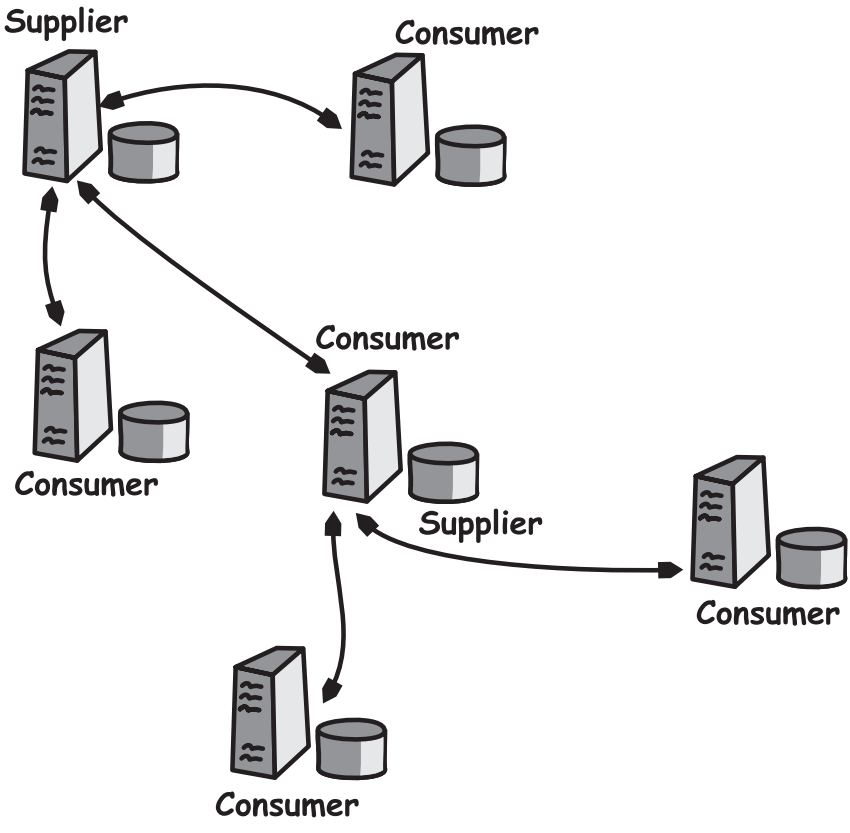


Abbildung 3.14: Einfache Replikationstopologie einer zentralisierten Datenhoheit

Beispiel:

Der User an Standort A greift auf einen Directory Server zu, der ein Replikat eines Servers an Standort B ist. Möchte dieser Anwender sein Passwort ändern, so wird dieser Änderungsrequest zwar am Replikat angenommen, dort aber nicht umgesetzt, sondern direkt an den Server am Standort B weitergeleitet. Erst dort wird das Passwort geändert und anschließend an die Replikate verteilt. Fällt der Weiterleitungs- oder Replikationsmechanismus aus, so sind keine Datenänderungen möglich.

Fehlende Informationshoheit

VORTEIL: Änderungen am Datenbestand können überall und zu jedem Zeitpunkt durchgeführt werden. Die Datensätze stehen den Anwendern jederzeit sowohl lesend als auch für die Modifikation zur Verfügung.

NACHTEIL: Es besteht die Gefahr der Dateninkonsistenz. Änderungen im Datenstamm können kollidieren, in der Summe kann es zu unauflösbaren Konflikten kommen, die zu korrupten Datensätzen führen.

Zentralisierte Informationshoheit

VORTEIL: Da es nur einen einzigen Dienst gibt, der Daten ändern kann, ist die Wahrscheinlichkeit von Kollisionen bei den Datenänderungen relativ gering und die Datenkonsistenz bleibt gewahrt.

NACHTEIL: Fällt der Replikationsmechanismus aus, sind Änderungen am Datenstamm nicht mehr oder nur noch eingeschränkt möglich.

In einer zentralisierten Topologie ist die einfachste Architektur diejenige, in der der gesamte Datenbaum von einem so genannten »Master«- oder »Supplier«-Directory zu den Replikaten (»Slaves« oder »Consumer«) verteilt wird. Zentralisierte Informationshoheit bedeutet aber nicht unbedingt, dass der gesamte Datenbestand unter der Kontrolle eines einzelnen Services steht. Vielmehr können Teile des DIT in die Verantwortung anderer Dienste gegeben werden.

Beispiel:

Das Datenmodell unserer Beispielfirma »meinefirma.com« hat die Zweige »ou=IT-Services« und »ou=marketing«. Der IT-Services-Zweig liegt in der Verantwortung des Directory Servers am Standort A, während der Marketing-Zweig vom Server an Standort B kontrolliert wird. Eine solche Replikationstopologie bezeichnet man als »Cross-Replikation« (siehe Abbildung 3.15).

Master (»Supplier«)

Als Master bezeichnet man den Service, der die Informationshoheit über einen Datensatz besitzt. Dieser ist in der Lage, Datenänderungen in seinem verantworteten Informationsbereich vorzunehmen.

Slave (»Consumer«)

Als Slave bezeichnet man den Service, der seine Daten von einem Master repliziert bekommt und selbst keine Datenänderungen vornehmen kann. Änderungsrequests auf Informationen werden direkt an den Master weitergeleitet. Erst über den Umweg über den Master werden dem Slave die Datenänderungen mitgeteilt.

Komplette Replikation

Der gesamte Datensatz eines Directory ist in der Verantwortung eines einzelnen Servers. Alle Datenänderungen werden hier zunächst eingepflegt, bevor sie an die Replikate verteilt werden.

Cross-Replikation

In diesem Fall wird die Verantwortung für Teilbäume auf verschiedene Services verteilt. Änderungen können in den eigenverantwortlich verwalteten Zweigen direkt vorgenommen werden, während sich die übrigen nur indirekt über den Umweg des jeweiligen Masters durchführen lassen.

Möchte man einem Zweig Individualität bieten, indem beispielsweise eine lokale Geschäftsstelle ihre eigenen Benutzerdaten selbst pflegt, wird der lokale Directory Server für diesen Datensatz Master gegenüber allen anderen Servern. Für den Rest der Daten übernimmt er die Rolle des Slave.

Beispiel (siehe Abbildung 3.15): Hier ist jeder Service gleichzeitig Supplier und Consumer, je nach Datenbereich. In diesem Szenario hat der linke Service (Geschäftsstelle A) die Informationshoheit über den linken Teilbaum. Nur hier dürfen Änderungen für diesen Teil der Daten durchgeführt werden. Der rechte Teilbaum repliziert für den Service in der Geschäftsstelle A eine Kopie seiner Daten von der Geschäftsstelle B aus.

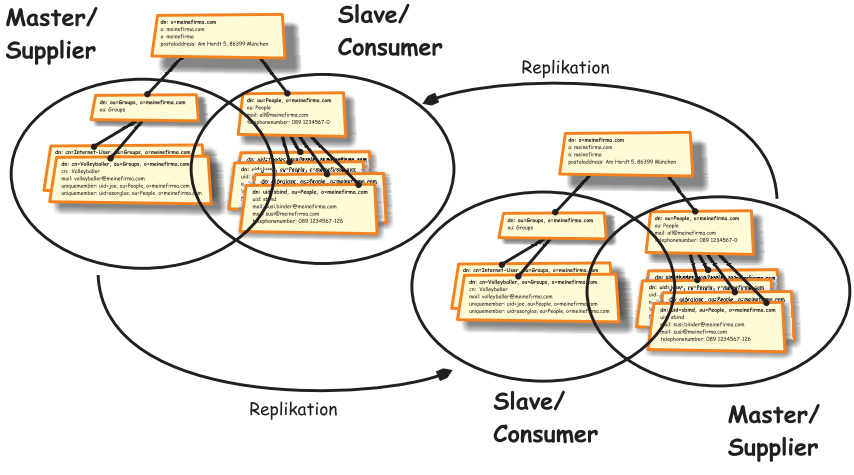


Abbildung 3.15: Cross-Replikation einzelner Teilbäume

Die verteilte Verantwortung für Daten eines DIT ist nicht zu verwechseln mit einer fehlenden Informationshoheit. Die einzelnen Daten unterliegen immer noch der Verantwortung eines einzelnen Dienstes, es besteht weiterhin eine zentralisierte Datenadministration und die Wahrscheinlichkeit für Kollisionen ist gering.

Supplier- und Consumer-Initiated Replication

Bei der Replikation werden unterschiedliche Trigger-Verfahren angewandt, um den Datenabgleich zu initiieren.

- SIR: Supplier Initiated Replication

  Die SIR basiert auf einer Replikationstopologie, in der ein Supplier (oder Master-) Directory Änderungen erhält und diese an die Replikate weitergibt. Dies geschieht entweder sofort nach Auftreten der Datenänderung oder es existiert ein Zeitfenster, in dem alle Updates vom Master aus an die Replikate übertragen werden. Dabei ist der Master der aktive Part: Er steuert den Zeitpunkt der Datenübertragung. Der Consumer (Slave) ist passiv und empfängt lediglich die Daten vom Master.

- CIR: Consumer Initiated Replication

  Bei der CIR fragt der Consumer (Slave) in regelmäßigen Zeitintervallen am Master an, ob sich irgendwelche Änderungen im Datensatz ergeben haben. Ist dies der Fall, fordert er diese an. Der aktive Part in dieser Replikationsarchitektur ist der Consumer. Der Master übermittelt nur dann die Datenänderungen, wenn er vom jeweiligen Consumer eine Anfrage erhält.

CIR ist nur dann sinnvoll, wenn sich der Consumer zu bestimmten Zeitpunkten über eine Dial-up-Verbindung mit dem Master verbindet. Nach der Datenübertragung aller Änderungen wird die Verbindung wieder abgebaut. CIR kommt allerdings recht selten zum Einsatz.

In der Praxis wird SIR am häufigsten als Replikationsverfahren verwendet. Insbesondere der direkte Upload von Directory-Daten stellt sicher, dass zu jedem Zeitpunkt der gesamte Datensatz auf allen Servern aktualisiert ist. Dies sichert die Datenzuverlässigkeit und fördert das Vertrauen der User in die Richtigkeit der Daten.

Multi-Master-Umgebungen und Replikationstopologien

Ein Directory-Netzwerk, das aus Gründen der hohen Verfügbarkeit mehrere Replikate für den lesenden Zugriff vorhält, weist immer noch einen Single Point of Failure auf: Der Master kann ausfallen und Datenänderungen sind nicht mehr möglich.

Für diesen Fall kann man entweder auf hardwarebasierte Lösungen zurückgreifen, indem man den Master in einer Cluster-Umgebung laufen lässt, oder man installiert den Master-Service in Form einer Multi-Master-Umgebung. Hierbei replizieren zwei oder mehr Master-Server ihre Daten wechselseitig im vollen Umfang (daneben werden auch die passiven Replikate mit Daten versorgt). Die Verteilung der Datenänderungen erfolgt über Loadbalancing-Mechanismen. Jeder Server nimmt einzelne Änderungen entgegen und repliziert diese unmittelbar auf den anderen Master. Fällt einer der Master aus, übernimmt der andere die volle Last der Datenänderungen. Wird der ausgefallene Server wieder in Betrieb genommen, synchronisiert der andere Master die Daten wieder.

Man unterscheidet hier zwischen verschiedenen Netzwerktypen:

- Das Replikationsnetzwerk
  In einem solchen Netzwerk werden die Daten ständig aktualisiert. Ein oder mehrere Master sind hier für die Aufnahme von Änderungen im Datenbestand verantwortlich. (Wir gehen hier von einer zentralisierten Informationshoheit aus. Die Master untereinander teilen sich jedoch die Verantwortlichkeit der Daten, beide »besitzen« die Daten.) Denkbar ist ein noch größerer »Masterpool«, um zusätzliche Ausfallsicherheit bereitzustellen. Da die Änderungen in einem Directory-Datensatz jedoch auf allen Servern übereinstimmen müssen, reduziert sich die Zahl der Einzeländerungen pro Server nicht. Damit ist auch keine Verbesserung der Schreib-Performance zu erwarten, wenn man mehrere Supplier bereitstellt. Damit sollte eine Topologie mit zwei Mastern für die Ausfallsicherheit des Schreibzugriffs genügen. Die Replikationen erfolgen nicht nur zwischen den Mastern untereinander, sondern auch auf die *n* verschiedenen Consumer in diesem Netz.

- Das *ldap*-Netzwerk
  Über dieses Netzwerk wird die direkte Abfrage an den Directory Server durchgeführt. Der Zugriff erfolgt über eine von mehreren Loadbalancing Bridges. Diese entscheiden aufgrund der Auslastung der einzelnen Consumer, auf welchen Server eine eingegangene *ldap*-Abfrage verteilt wird.

- Heartbeat-Netzwerk
  Dieses Teilnetz sorgt für die Überwachung mehrerer Loadbalancing Bridges. Diese sind mehrfach ausgelegt, um einen Single Point Of Failure zu vermeiden. Fällt der Haupt-Balancer aus, springt eine der Secondary Bridges ein und übernimmt dessen IP-Adresse und Aufgabe. Getriggert wird diese Aktion über den permanenten Heartbeat, der über dieses Netzwerk ausgetauscht wird. Im Normalfall setzt man zwei Loadbalancing Bridges ein.

- Client-Access-Netzwerk
  Der *ldap*-Client spricht auf einen Domain-Namen an, der ihm den Directory-Access liefert.

Vordringlichstes Ziel bei einer solchen Implementierung sind die Ausfallsicherheit und Performance. Derartige Anforderungen findet man in großen Unternehmen und bei ISPs, die darauf angewiesen sind, 24 Stunden 365 Tage lang eine Verfügbarkeit des Dienstes von 99,9995% – und besser – zu gewährleisten. Dies ist vor allem dann der Fall, wenn am *ldap*-Dienst Mail-, Einwahl-, Portal-, Profil- und/oder Abrechnungsdienste hängen. Unter solchen Umständen ist der Directory Server ein unternehmenskritischer Dienst. Die Daten müssen sowohl für den lesenden als auch für den schreibenden Zugriff in hoher Performance vorliegen. Die beschriebenen Techniken sind wesentlicher Bestandteil der Hochverfügbarkeitslösungen.

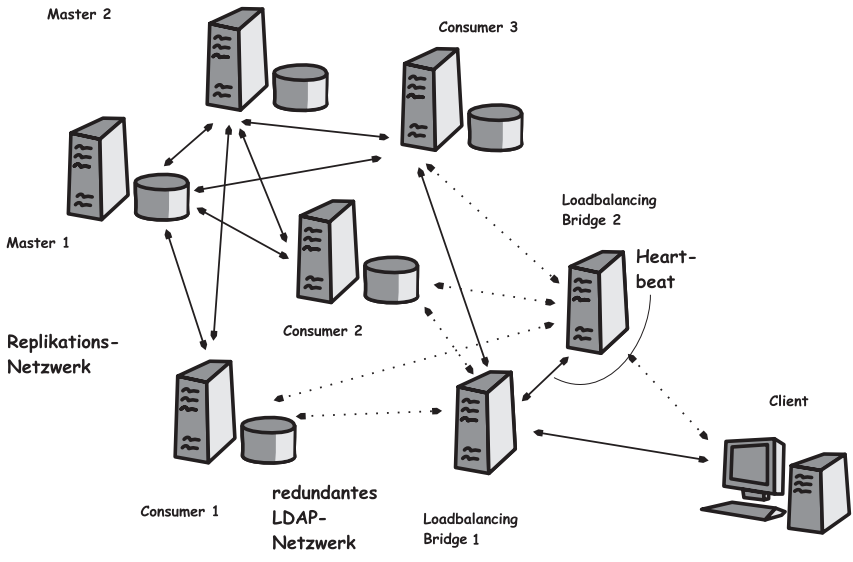


Abbildung 3.16: Mehrstufig-redundantes ldap-Netzwerk

Metadirectorys

Eine zentralisierte Directory-Topologie ist leider nur graue Theorie, in den seltensten Fällen können alle Daten in einem zentralen User-Repository hinterlegt werden. Dies liegt zum Teil an den verwendeten Applikationen: Viele Applikationen sind nicht in der Lage, über *ldap* auf das zentrale Directory zuzugreifen. In der Realität reduziert ein Directory Server den administrativen Aufwand der Pflege von Benutzerdaten, es gibt aber immer Applikationen, die zusätzlichen Aufwand dieser Art erzeugen.

Neben den reinen »Insellösungen«, die keinen Datenaustausch zulassen, gibt es so genannte Metadirectorys, die über Umwege die Datensynchronisation erlauben. So kann man über geeignete Synchronisationstools den Datenabgleich automatisieren, so dass Benutzerdaten, die im Directory geändert werden, über geeignete Tools mit den anderen Benutzerdatenbanken abgeglichen werden.

Beispiel:

Benutzerdaten liegen sowohl im Directory Server als auch in der Personaldatenbank vor. In beiden Systemen werden ähnliche Daten gepflegt. Eine direkte Synchronisation über *ldap* ist entweder nicht möglich oder aus Sicherheitsgründen nicht erwünscht. Wird dennoch eine Synchronisation angestrebt, können geeignete Tools die Daten aktualisieren, ohne dass eines der Systeme die Hoheit über die Informationen aufgibt. Jedes System hat damit sein individuelles Recht an den Daten. Werden diese automatisch synchronisiert, so spricht man von einem Metadirectory.

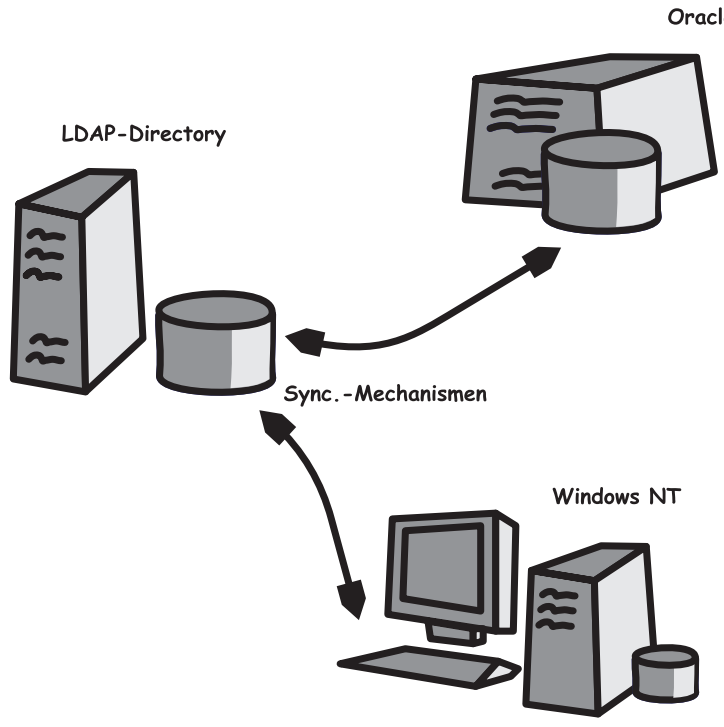


Abbildung 3.17: Metadirectory-Strukturen

Abbildung 3.17 zeigt ein Metadirectory mit drei Komponenten. Der Aufbau einer zentralen Instanz als Benutzerdatenbank, von der die anderen Systeme als Clients profitieren, ist nicht möglich. Daraus ergibt sich, dass alle diese Systeme ihre eigenen User-Daten haben, die synchronisiert werden müssen.

Wird im vorliegenden Beispiel das User-Passwort in Windows NT geändert, so registriert ein Synchronisationsmechanismus in Windows NT diese Änderung und übermittelt sie an das zentrale *ldap*-Verzeichnis. Genauso wird eine Änderung im *ldap*-Verzeichnis dem NT-Betriebssystem mitgeteilt.

Über ein entsprechendes Tool kann gleichfalls auf Seiten der Oracle-Datenbank eine Datenänderung ein Update auf dem Directory umsetzen und umgekehrt.

Metadirectory-Strukturen sind anfällig gegenüber Dateninkonsistenzen. Da alle beteiligten User-Repositorys die Datenhoheit haben, können Konflikte auftreten, die von dritter Seite aufgelöst werden müssen.

3.8 ldap-Clients

Ein Benutzerverzeichnis liefert Informationen mittels *ldap*. Dieses Kommunikationsprotokoll dient als Sprache zum Auslesen und Ändern von Daten an einem Directory Server. *ldap*-Clients greifen auf diese Daten zurück und

müssen dieses Protokoll »sprechen«, um einerseits Queries anzunehmen und in *ldap* zu übersetzen, mit dem Directory in Kontakt zu treten – und andererseits müssen sie die Ergebnismenge entgegennehmen, auswerten und je nach Anwendung aufbereitet darstellen.

Eine Suche in einem Verzeichnis basiert immer auf der gleichen Struktur: Man spezifiziert den Rechnernamen, auf dem das Verzeichnis läuft (z.B. ldap.meinefirma.com), formuliert den Filter, der die gesuchte Information benennt (z.B. cn=joe*), und gibt an, welche der Attribute gefordert werden, wenn der gesuchte Eintrag gefunden ist. Die »Base-DN« benennt bei diesem Request den Startpunkt der Suche im DIT, der »Scope« markiert den Suchbereich unterhalb dieses Startpunkts. Authentifizierungsdaten wie User-ID und Passwort können benötigt werden, wenn der Zugriff auf das Directory nicht für den anonymen Zugriff gestattet ist.

Wir betrachten die einzelnen Parameter der *ldap*-Request-Spezifikation im Folgenden etwas genauer.

BaseDN

Der BaseDN – oder Base Distinguished Name – benennt den Ausgangspunkt der Suche im Verzeichnis. In der Beispielfirma kann dies »o=meinefirma.com« sein. Mit diesem BaseDN sucht der Directory Server bei einer Abfrage alle Zweige unterhalb der Wurzel ab. Legt man bei einer *ldap*-Query den BaseDN unterhalb der Wurzel fest, beschränkt sich die Suche nach Daten nur auf diesen Eintrag und jeden weiteren, der »unterhalb« dieses Knotens liegt.

Beispiel:

Im Beispiel-DIT ist der Wurzeleintrag mit »o=meinefirma.com« definiert. Unterhalb dieses Suffix ist ein Zweig mit den Lokalisationen »Europa« (l=europa, o=meinefirma.com) und Asien (l=asien, o=meinefirma.com) hinterlegt. Eine Suche, die mit dem BaseDN »o=meinefirma.com« spezifiziert wird, beginnt in der Wurzel und erstreckt sich über den gesamten Datenbaum. Eine andere Suche, die über die BaseDN »l=europa, o=meinefirma.com« festgelegt wird, sucht ausschließlich im europäischen Ast des DIT. Der asiatische Zweig wird für die Suche nicht herangezogen.

Suchfilter

Neben der Basis für den Beginn der Suche spezifiziert ein *ldap*-Request weitere Suchkriterien. Dazu gehört der Suchfilter, der die gewünschte Information beschreibt. Hier können sehr genaue Angaben gemacht werden, die den Eintrag im Directory eindeutig benennen, oder aber es werden allgemeine Anfragen durchgeführt, die sich über die Wildcard-Nomenklatur definieren. Wir haben die Formulierung eines Suchfilters bereits in Abschnitt 3.7.1 angesprochen und führen ihn hier nur noch der Vollständigkeit halber erneut auf:

(uid=juser)	zielt auf den Eintrag mit der UID »juser«
(givenname=Joe*)	zielt auf alle Einträge, deren Vorname mit »Joe« beginnen
(&(mail=*sun*)(l=europe))	zielt auf alle Einträge, die die Lokation »Europe« beinhalten und bei denen der Substring »sun« Bestandteil der Mailadresse ist

Attribute

Wird ein Eintrag gefunden, der dem Suchfilter entspricht, so hat dieser Datensatz unter Umständen eine umfangreiche Liste an Attributen und zugehörigen Daten. So viele Informationen sind in den meisten Fällen gar nicht erforderlich bzw. sogar hinderlich. Mit der Angabe der Attribute, an denen man interessiert ist, wird aus der Ergebnismenge nur die Information herausgefiltert, die für den Interessenten relevant sind.

Beispiel:

Ein Request zielt auf den Eintrag von Joe User und fordert die Mailadresse über das Attribut »mail« an.

Ein anderer Request zielt auf alle Einträge, deren Adresse den Substring »Stuttgart« beinhalten (der Suchfilter lautet »(postaladdress=*stuttgart*)«). Aus dieser Ergebnismenge fordert man alle Telefonnummern an.

Scope

Die Angabe des Scope in einem Request hat drei verschiedene Werte:

- »base«: Wird in der Formulierung des Request dieser Wert angegeben, werden nur Informationen von genau diesem Eintrag des BaseDN geliefert. Eine weitere Suche nach unten im Teilbaum erfolgt nicht. Beispiel: Eine *ldap*-Suche zielt genau auf den Eintrag »ou=Marketing, l=europe, o=meinefirma.com«. Von diesem Eintrag wird die Telefonnummer der Zentrale gefordert. Bei dieser Anfrage wird keine weitere Suche in die unteren Zweige des DIT gefordert.
- »one«: Mit diesem Wert zielt der Request genau auf einen Level unterhalb des BaseDN. Weitere Level werden nicht berücksichtigt. Beispiel: Mit dem Scope »one« zielt eine Suche, die mit dem BaseDN »ou=People, o=meinefirma.com« definiert wurde, auf alle Einträge, die genau unterhalb dieses DIT installiert sind. Dazu gehört beispielsweise »uid=juser, ou=People, o=meinefirma.com«. Dazu gehört aber nicht mehr »userprofile=Portal, uid=juser, ou=People, o=meinefirma.com«.
- »sub« oder »subtree«: Ein Request, der mit dem Scope »sub« Informationen aus dem Directory anfordert, adressiert die Suche auf alle Einträge, die bei dem BaseDN beginnen und unterhalb dieses Knotens platziert sind.

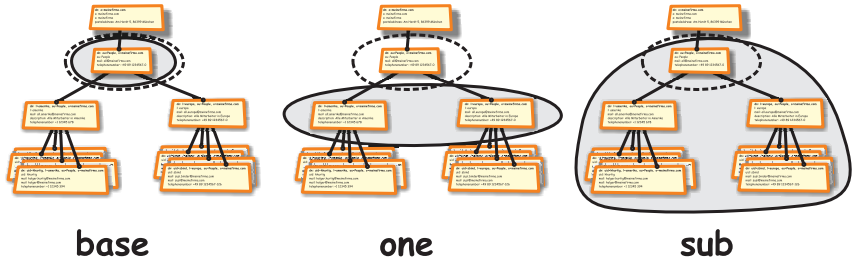


Abbildung 3.18: Verschiedener Scope eines ldap-Request. Der BaseDN bei dieser Suche ist jeweils gestrichelt markiert.

Alle *ldap*-Clients formulieren ihre Requests auf diese Art, unabhängig davon, ob es sich um ein grafisches Tool mit Buttons und Links handelt oder ob man ein Commandline-Tool verwendet.

Ein sehr verbreitetes *ldap*-Tool ist Netscapes Communicator-Adressbuch, das Bestandteil der Browser-Software ist. Das Adressbuch wird für einen Directory Server vorkonfiguriert, wobei der Rechnername des Verzeichnisses, der BaseDN und eventuell erforderliche Authentifizierungsdaten wie User-ID und Passwort eingetragen werden. Anschließend liefert das Adressbuch in einer sehr komfortablen Weise den Vor- oder Nachnamen des gesuchten Mitarbeiters, die Telefonnummer, die E-Mail-Adresse usw. Dazu reicht bereits das Eintragen eines Teils des Namens.

Neben dem Adressbuch hat Netscape die *ldap*-Fähigkeit direkt in der URL-Eingabemaske des Browsers integriert. Hierfür wird in die Adressleiste eine *ldap*-URL eingegeben, die auf eine Datenmenge in einem Directory Server zeigt. In Kapitel 13.1.3 ist deren Struktur in Einzelheiten beschrieben. Das Browser-Fenster liefert in *html*-Form die Ausgabemenge der Ergebnisse.

Viele Implementierungen in Unternehmen benutzen ein eigenes, für diesen Zweck erfasstes Adressbuch auf Web-Basis. Benutzer klicken auf die URL http://phonebook.meinefirma.com und finden eine Eingabemaske, in der sie ihre Suchanfrage formulieren und die Ergebnismenge über *html* auf dem Browser dargestellt bekommen. Diese Applikationen sind teilweise als Templates auf dem Markt erhältlich, man muss sie jedoch den eigenen Bedürfnissen anpassen, damit das Webadressbuch der jeweiligen Corporate Identity entspricht.

Neben Endanwender-Programmen der bisher vorgestellten Art sind natürlich alle möglichen Endgeräte ebenfalls denkbar – sei es ein PDA oder Handheld, ein Mobil- oder Festnetztelefon. Diese werden hier nicht weiter besprochen. Aus der Liste der Client-Applikationen, bei denen der Benutzer direkt mit dem Verzeichnis interagiert, sei aber eine besonders hervorgehoben: ldapsearch.

Mit diesem kommandozeilenbasierten Tool kann man jede Information aus einem *ldap*-basierten Verzeichnis abfragen. Die Syntax ist umfangreich, Manpages mit detaillierten Beschreibungen der einzelnen Befehle sind im Internet zu finden. Es existieren verschiedene Implementierungen auf dem Markt, die allerdings weitgehend identische Parametrisierungen beinhalten.

Deshalb folgt an dieser Stelle keine ausführliche Vorstellung dieses Tools, sondern nur eine beispielhafte Darstellung der Verwendung.

Die Syntax von ldapsearch lautet vereinfacht:

```
ldapsearch -D <Username> -w <Passwort> -h <Directory-Host> -p <Port>
-b <BaseDN> <Suchfilter> <Ergebnis>
```

Folgender Befehl steuert eine Suche über unser Beispielverzeichnis:

```
ldapsearch -D “cn=Joe User" -w soisses -h ldap.meinefirman.com -p 389
-b "o=meinefirma.com" “(&(cn=joe*)(ou=marketing)) mail
```

Das Ergebnis einer solchen Suche kann etwa folgendermaßen aussehen:

```
dn: uid=juser, ou=people, o=meinefirma.com
mail: joe.user@meinefirma.com
mail: juser@meinefirma.com
mail: freak@meinefirma.com
```

Das Attribut »mail« im vorliegenden Beispiel ist für den Eintrag von Joe User mehrfach verwendet worden, entsprechend werden alle diese Daten aufgelistet.

Die Bedeutung der Flags im vorliegenden Beispiel im Einzelnen:

-D »cn=Joe User«	Ich möchte mich als »Joe User« am Verzeichnis anmelden.
-w »soisses«	Als »Joe User« habe ich das Passwort »soisses«.
-h ldap.meinefirma.com	Ich möchte meine Abfrage auf diesem Directory Server starten.
-p 389	Der Directory Server läuft auf Port 389.
-b »o=meinefirma.com«	Ich möchte die Suche auf diesem BaseDN starten.
(&(cn=joe*)(ou=marketing))	Ich suche alle Mitarbeiter, deren Name mit »Joe« beginnt und die im Marketing arbeiten.
mail	Findet die Suche einen Treffer, dann hätte ich gerne aus diesem Eintrag die Mailadresse bekommen.

Weitere Clients, die vom Verzeichnis Benutzerdaten einfordern, sind Dienste wie der Mails-, Remote Access-, News- oder Proxyservice. Wie wir noch sehen, verlassen sich mittlerweile eine Reihe von Programmen dieser Art auf einen bestehenden Verzeichnisdienst.

3.9 Directorys und Passwörter

Einer der wichtigsten Vorteile eines Directory Servers ist die Möglichkeit, dass der Benutzer idealerweise ein einziges Passwort hat, mit dem er sich an allen Applikationen anmeldet, die ihre Authentifizierung dem Directory überlassen. So vergleicht beispielsweise eine Webapplikation die User-Daten nicht mehr gegen ihren eigenen isolierten Benutzerdatenstamm, sondern greift über *ldap* auf den Datenstamm des Directory zurück. Dadurch kann der User sich mit diesem einen Passwort an allen Applikationen anmelden und ist nicht mehr gezwungen, sich viele verschiedene Passwörter merken zu müssen.

Auf den ersten Blick birgt dieser Ansatz aber eine Sicherheitslücke: Der Zugriff auf das Attribut »userpassword« kann zwar über ACIs geschützt werden. Dies hindert allerdings den Administrator nicht, die Datenbank des Verzeichnisses in ein LDIF-File zu exportieren, das die Daten in ein Textfile sichtbar macht.

Aus diesem Grund werden die Passwörter der User nicht im Klartext im Directory hinterlegt, sondern sie werden zunächst gehashed, bevor dieses Ergebnis im Directory aufgenommen wird. Wir werden in Kapitel 6.5.1 detailliert auf den Hash-Algorithmus eingehen. An dieser Stelle sei nur gesagt, dass es sich um eine Einwegfunktion handelt, deren Eingabe der Klartext ist und deren Ausgabe einen Wert ergibt, der für die Eingabe charakteristisch ist, aber keine Rückschlüsse auf diesen zulässt. (Die Quersumme einer Zahl ist eine Einwegfunktion: Die »Klartext«-Zahl 1234 liefert den Wert 10, aus dem Wert 10 kann man niemals die Zahl 1234 errechnen. Damit ist aber noch keine Hash-Funktion gegeben, diese muss wesentlich umfangreicheren Kriterien genügen, als dies bei einer simplen Quersumme gegeben ist. Dazu folgt später mehr.)

Wie funktioniert nun eine Authentifizierung über ein User-Passwort, das gehashed im Directory hinterlegt wurde? Eine Applikation, die über einen Authentisierungsmechanismus geschützt ist, nimmt die User-Daten (User-ID und Passwort) entgegen. Zunächst überprüft sie über *ldap* am Directory Server, ob es einen User mit der bezeichneten ID überhaupt im Verzeichnis gibt. Ist dies der Fall, nimmt sie den Hash des Passworts und vergleicht diesen Wert mit dem (gehashten) Passwort im Verzeichnis. Da diese Funktion bei gleicher Eingabe immer eindeutig das gleiche Ergebnis liefert, kann ein Vergleich der Hashes die Gültigkeit des Passworts überprüfen, ohne dass es im Klartext bekannt sein muss.

Durch die Verwendung eines Directory Servers kann man die verwendeten Authentifizierungsdaten reduzieren. Dadurch wird die Administration der Applikationen erleichtert. Außerdem müssen die User nicht mehr so viele verschiedene Passwörter und User-IDs verwalten, so dass sich eine deutliche Verbesserung des Sicherheitsgefüges in einem Unternehmen einstellen sollte.

3.10 ldap – Integration und Migration

Die Integration ins bestehende Netzwerk basiert auf einer sehr komplexen Vorgehensweise. Da das Directory nicht zum Selbstzweck eingerichtet werden soll, bedarf es einer Analyse der bestehenden Systeme, die in diesem Konzept berücksichtigt werden sollen. Hier wird man bald feststellen, dass eine hundertprozentige Integration aller Applikationen in das *ldap*-Konzept nicht möglich ist und man Kompromisse in Bezug auf die bestehende Infrastruktur machen muss. Applikationen sind nicht immer aus dem Stand *ldap*-fähig und müssen gegebenenfalls überarbeitet werden, unter Umständen führt dies zu Neuanschaffungen oder zur Integration in andere Anwendungen. Eine elektronische Telefonliste beispielsweise, die von einer Telefonzentrale gepflegt wird, muss unter Umständen neu programmiert werden. Andere Systeme, wie z.B. Personalverwaltung oder ERP-Systeme, sind so komplex, dass sie sich nicht integrieren lassen. Daher ist das Ideal einer hundertprozentig zentralisierten Benutzeradministration nur in den seltensten Fällen möglich und man begnügt sich damit, so viele Applikationen wie möglich an das Verzeichnis zu binden.

3.10.1 System- und Businessanalyse

Bei der Untersuchung, welcher Dienst nun in das Directory integriert wird, spielt natürlich die *ldap*-Fähigkeit eine übergeordnete Rolle. Er muss in der Lage sein, einen *ldap*-Request über TCP/IP an den Directory Server zu übermitteln und anschließend die Ergebnismenge auszuwerten und zu interpretieren. Das allein reicht aber noch nicht aus. In einer Applikation wie einem elektronischen Telefonbuch werden möglicherweise Attributnamen verwendet, die nicht im Standardschema enthalten sind. Hier muss man entweder die Applikation oder das Directory an die Gegebenheiten anpassen. Gleiches gilt für die anderen Anwendungen. So bedarf es einer System- und Businessanalyse, die eine Gegenüberstellung der erforderlichen Investitionen in Form von Zeit, Material und Ressourcen mit dem Nutzen einer zentralisierten User-Datenhaltung vornimmt. Darin müssen alle Faktoren enthalten sein, auf die ein Directory Server Einfluss hat. Alle in Frage kommenden Applikationen und Clients müssen in Bezug auf Anpassung oder Neuinvestition analysiert werden, um den wirtschaftlichen Nutzen einschätzen zu können.

Des Weiteren müssen die Auswirkungen einer geänderten Informationshoheit untersucht werden. Unter Umständen ist nicht jede Abteilung damit einverstanden, wenn sie die über Jahre hinweg gepflegten Daten aus der Hand geben und einem anderen Verantwortungsbereich überlassen muss.

Ein weiterer Kostenfaktor ist die Datenmigration. Je nach Ausgangssituation liegen die Daten in einer mehr oder weniger brauchbaren Form vor. Datensätze können unvollständig sein, andere Datenquellen liefern unter Umständen veraltete Einträge oder es existieren Inkonsistenzen in den Ausgangsdaten.

Das Ergebnis einer System- und Businessanalyse liefert letztendlich die Argumente für (oder gegen) die Investitionen. Erst dann sollte die Projektierung mit Systemkonzeption, Test- und Pilotphase bis hin zur Produktion erfolgen.

3.10.2 Konzeptionierung der Directory-Daten

Applikationen, die an den Directory-Dienst angebunden werden sollen, müssen nicht nur auf technische Realisierung hin untersucht werden, sondern vielmehr auch auf die Wirtschaftlichkeit des Aufwands. Eine Applikation, die zwar auf einen zentralen Verzeichnisdienst zugreifen kann, aber nur von einer Handvoll Benutzer sporadisch genutzt wird, muss nicht unter allen Umständen in dieses Konzept eingegliedert werden. Dies gilt insbesondere dann, wenn die Integration einen nicht unerheblichen Aufwand an Systemänderungen erfordert.

Nachdem die Applikationen identifiziert sind, die für eine zentralisierte Benutzerhaltung in Frage kommen, müssen die Personen, die mit diesen Applikationen arbeiten, an der Konzeption beteiligt werden. Eine Telefonzentrale wird wahrscheinlich darauf drängen, die Informationshoheit über die Modifikation von Telefonnummern zu erhalten. Die Personalabteilung legt Wert darauf, als einzige Instanz Namen und Dienstgrade ändern zu dürfen. Kompetenzen müssen diskutiert und neu verteilt werden, Zugriffs- und Änderungsrechte bezüglich der Benutzerdaten gilt es, neu zu überdenken. Applikationen, die die Administration der Daten ermöglichen, müssen entwickelt oder eingekauft werden. Sicherheitskonzepte müssen auf Modifikationen untersucht, Änderungen in den Applikationen realisiert werden.

Neben einer Aufstellung von Diensten und Anwendungen, die auf das neue System zugreifen werden, muss man die Quellen identifizieren, aus denen die Informationen der Benutzer stammen. Beispiele hierfür reichen von der Personalabteilung und Telefonlisten bis hin zu Mail- und Betriebssystemen.

Schließlich müssen die verwendeten Attribute noch einmal auf den Sinngehalt hin überprüft werden. Daten, die sich häufig ändern, sind nicht für einen Directory Server geeignet. Genauso wenig sind umfangreiche Daten *ldap*-tauglich. Das Verzeichnis ist kein Dokumenten-Managementsystem! Demzufolge sollten weder Bilder noch Videos oder Powerpoint-Präsentationen Einzug in das Directory halten. Im Idealfall hat ein Eintrag im Verzeichnis eine Größe von bis zu 2 Kbyte. Größere Einträge verschlechtern die Performance des Systems maßgeblich.

In Bezug auf die Datendefinition gilt außerdem, dass man sich die Gesamtmenge der durch das Standardschema vorgegebenen Attribute und Objektklassen ansehen sollte, bevor man eine Erweiterung des Schemas implementiert. Jede Schemaerweiterung ist proprietär, die damit verbundenen Attribute und Objektklassen werden nur von den eigenen Applikationen unterstützt und dienen nicht der unternehmensübergreifenden Kommunikation.

3.10.3 Architektur des Directory Information Tree (DIT)

Für die Architektur des DIT existieren die verschiedensten Ansätze. Das Spektrum reicht von sehr »tiefen« Bäumen bis hin zu besonders »flachen« Datenstrukturen. Weiter kann die Wahl des Suffix oder die Syntax des Distinguished Name für bestimmte Applikationen entscheidend sein. Ziel ist es nun, herauszufinden, welche Architektur optimal für die eigenen Bedürfnisse ist und welche einen guten Kompromiss bezüglich der bestehenden Topologie darstellt.

Eine der ersten Aufgaben ist die Benennung der Wurzel des Baums. Da der *ldap*-Directory-Dienst seine Entstehungsgeschichte im X.500-Verzeichnis hat, folgen viele Implementierungen noch dieser Notation, die sich aus Organization und Country zusammensetzt:

```
dn: o=meinefirma, c=de
```

Diese Nomenklatur ist dann interessant, wenn das *ldap*-Verzeichnis mit einem X.500-Directory kooperieren muss. Sie ist festgelegt auf die Syntax:

```
dn: o=<Organisation>, c=<Land>
```

Andere Ansätze verwenden den Unternehmensnamen oder die Domain als Suffix, wie beispielsweise:

```
dn: o=meinefirma
```

oder über die »Domain Component« (dc)

```
dn: dc=meinefirma, dc=com
```

Üblicherweise lassen es die verschiedenen Directory-Server-Produkte auf dem Markt zu, dass man das Suffix frei wählen kann. Demgegenüber existieren viele Applikationen, die die eine oder andere Form der Syntax nicht unterstützen, was die Wahl des Suffix einschränkt.

Der nächste Schritt ist ungleich schwieriger und betrifft die Verteilung der Daten in eine effiziente Directory-Topologie. Diese wird im Wesentlichen durch die drei folgenden Aspekte beeinflusst:

1. X.500-Koexistenz

   Wenn das Verzeichnis mit einem X.500-Verzeichnis abgeglichen werden oder in irgendeiner anderen Form koexistieren soll, wird man sich an der Nomenklatur dieser Directory-Form orientieren. Die originäre Implementierung sieht vor, dass der Suffix durch »organization« und »country« definiert wird, gefolgt von den Einträgen »organization units (ou)« und »common name (cn)«.
   Diese Vorgaben erschweren es manchmal, eine Topologie zu finden, die auf die tatsächliche Organisation in Form von Abteilungen und Gruppierungen passt.

2. Replikation oder Referrals

   In einer Topologie, in der Replikationen oder Referrals (Verweise über *ldap*, siehe Abschnitt 3.7.5) vorkommen, ist es sehr wichtig, dass man diese Aspekte frühzeitig in die Konzeptionierung des DIT übernimmt. Beispielsweise kann man einen Baum, der die Struktur

   ```
   dn: ou=People, o=meinefirma.com
   ```

   besitzt, nicht nach Lokalisationen verteilt replizieren, da diese Information nicht in Form eines Knotens im Distinguished Name enthalten ist. Dies wird erst durch Einbeziehung eines neuen Knotens im DIT möglich:

   ```
   dn: ou=People, l=Hamburg, o=meinefirma.com
   dn: ou=People, l=Muenchen, o=meinefirma.com
   ```

   Diese Topologie ist für die Replikation sehr einfach, da sie es gestattet, die jeweiligen Teilbäume der einzelnen Lokalisationen Hamburg und München getrennt voneinander zu replizieren.

3. Dynamische und administrative Aspekte

   Orientiert man sich bei der Bestimmung des DIT zu sehr an der tatsächlichen Organisation des Unternehmens, kann es sehr schwierig werden, Änderungen der Unternehmensstruktur im Verzeichnis nachzuvollziehen. Das liegt daran, dass die Schlüsselwerte des Distinguished Name in alle Teilbäume und Blätter des DIT vererbt werden. Wird der Name einer Organisationseinheit, beispielsweise einer Abteilung, geändert oder aufgelöst, so müssen alle Teilbäume und Blätter des DIT, die im Schlüsselwert den Namen dieser Organisationseinheit verwenden, geändert werden. Dies kann unter Umständen nur über Löschen und Neuanlegen der Daten geschehen, was auf Kosten der System-Performance geht.

   Beispiel:

   Bei der Erstellung des DIT für den Directory Server der Firma meinefirma.com orientiert man sich an den Vorgaben der Abteilungen:

   ```
   dn: ou=People, ou=IT-Services, o=meinefirma.com
   dn: ou=People, ou=Marketing, o=meinefirma.com
   dn: ou=People, ou=Sales, o=meinefirma.com
   ```

   Gerade in dieser Topologie ist es schwierig, ganze Teilbäume wie zum Beispiel die Organization Unit

   ```
   »ou=People, ou=Marketing, o=meinefirma.com«
   ```

   in den Teilbaum

   ```
   »ou=People, ou=Technisches Marketing, o=meinefirma.com
   ```

   umzubenennen. Diese Maßnahme betrifft nicht nur den Distinguished Name des benannten Knotens, sondern auch den aller Blätter unterhalb dieses Eintrags. Damit beeinflusst diese Änderung alle Personeneinträge unterhalb des Teilbaums »ou=Marketing, o=meinefirma.com«. (Um diese Änderungen durchzuführen, muss zunächst der neue Teilbaum

»ou=Marketing, o=meinefirma.com« als neuer Knoten im DIT angelegt werden. Anschließend muss der Knoten »ou=People« angefügt werden, gefolgt von der Neuanlage aller Benutzereinträge aus dem alten Teilbaum »ou=Marketing, o=meinefirma.com«. Schließlich werden alle alten Einträge aus dem Directory Server unterhalb gelöscht.)

Je weiter oben im DIT diese Änderungen durchzuführen sind, desto umfangreicher fallen die Maßnahmen aus. Je flacher ein DIT ist, umso weniger betreffen Änderungen der Daten die Schlüsselinformationen im Distinguished Name und umso dynamischer können Änderungen in den Daten durchgeführt werden.

Die organisatorische Zugehörigkeit lässt sich immer noch in einem »ou«-Eintrag im Datensatz der Person zuordnen, ohne dass diese im dn erscheint:

```
dn: uid=juser, ou=People, o=meinefirma.com
…
uid: juser
ou: IT-Services
…
```

Wechselt ein Mitarbeiter oder eine ganze Gruppe die Organisationseinheit (von »ou=IT-Services« zu »ou=Support«), so genügt es, dieses Attribut in jedem Eintrag zu ändern. Es ist dann nicht mehr notwendig, die Gesamtinformation zu löschen und woanders wieder neu anzulegen.

Die beiden letzten Aspekte wirken einander entgegen. Erfahrungsgemäß findet sich ein Optimum der Baumtiefe irgendwo zwischen drei und sechs Levels. Diese Werte orientieren sich vornehmlich an Replikationstopologien, aber auch an der Menge der zu verwaltenden Einträge. Ein Verzeichnis mit 20 Millionen Objekten stellt andere Anforderungen an den Baum als eines mit 2000. Die besten Ansätze liefert üblicherweise eine Strukturierung mit geringer Granularität, geordnet nach den notwendigsten Organisationsstrukturen mit Augenmerk auf die zu realisierende Replikationsstruktur. Wenn sich dann eine Baumtiefe im genannten Bereich ergibt, sollte für die grundlegende Architektur ein solides Gerüst gefunden sein.

3.10.4 Charakterisierung der Daten

Konsolidierung der vorhandenen Datenbezeichnungen

Mit der Festlegung des DIT steht die grundsätzliche Organisation der Daten fest. Im nächsten Schritt der Datenmodellierung müssen die Attribute benannt werden, die die Daten aufnehmen werden. Hierfür trägt man alle Quellen zusammen, die zu den Directory-Daten beitragen, und ordnet diese zu einem gemeinsamen Datenverständnis. Ziel ist es, diese Daten zu konsolidieren und mit der Inbetriebnahme des Directory abzulösen.

Für diese Aufgabe ist es sehr hilfreich, wenn man eine Matrix erstellt, in der alle beteiligten Datenquellen zusammengefasst und aufgelistet werden, die im späteren Verlauf vom Directory profitieren. Jede Datenquelle hat ihre eigene Art, die User-Daten zu verwalten. Die dabei verwendeten Bezeichner werden in der Matrix erfasst und mit den Attributnamen im Directory abgeglichen. An dieser Stelle sieht man bereits, in welchem Rahmen sich der Aufwand für das Abgleichen der einzelnen Applikationen auf den neuen Standard bewegt.

Beispiel:

IT-Admin	Personal-abteilung	Telefonliste	Mailsystem	**Directory**
		»Vor- und Zuname«	»Name«	**»cn«**
»firstname«	»Vorname«			**»givenname«**
»name«	»Nachname«			**»sn«**

In diesem Beispiel sind vier verschiedene Datenquellen für User-Daten angegeben, die alle den Namen einer Person in irgendeiner Form bereitstellen. Der Directory Server als zentrales User-Repository liefert später ab Produktionsbeginn diese Daten an diese Applikationen aus und löst die Pflege dieser Quellen ab. In der Datenmodellierungsphase aber werden alle vorhandenen Daten dieser Quellen konsolidiert und in ein einheitliches Schema gebracht. Im vorliegenden Beispiel sehen wir, dass beispielsweise der Nachname von Personen an zwei Stellen (»IT-Admin« und »Personalabteilung«) gepflegt wird, wobei die Bezeichnungen dafür unterschiedlich sind (»Name« und »Nachname«).

Datenbereinigung

Unter Umständen sind auch die Werte dieser Quellen unterschiedlich: Die IT-Abteilung führt den Namen einer Mitarbeiterin beispielsweise unter »mueller«, die Personalabteilung unter »Dr. Müller-Schmidt«. Bei der Inbetriebnahme des Directory soll es aber nur noch eine einzige Version des Namens geben: den, den das User-Repository liefert.

Beispiel:

	IT-Admin	Personal-abteilung	Telefon-liste	Mailsystem	**Directory**
Name			»Sabine Müller-Schmidt«	»Sabine Mueller«	**»Sabine Müller-Schmidt«**
Vorname	»sabine«	»Sabine-Anna«			**»Sabine«**
Nach-name	»muel-ler«	»Dr. Müller-Schmidt«			**»Müller-Schmidt«**

Man muss den beschriebenen Konflikt der unterschiedlichen Datenversionen auflösen und ein gemeinsames Verständnis für die Datensicht herstellen.

Nach Beendigung dieser Phase ist die Konzeption der vorläufigen Architektur abgeschlossen. Im nächsten Schritt werden die Replikationstopologien festgelegt. Für den Fall, dass keine Replikate vorgesehen sind, kann man diesen Punkt überspringen.

3.10.5 Festlegung der Replikationstopologie

Sind im Netzwerk die Arbeitsplätze über mehrere Standorte verteilt, so macht die Überlegung einer verteilten Datenhaltung Sinn. Je nach Anforderung müssen an den einzelnen Lokalisationen schnelle Antwortzeiten des Directory gewährleistet werden.

Beispiel:

Proxyserver, die den Internetzugriff steuern, authentifizieren jedes neu angeforderte Dokument über den Directory Server. (Jedes Webdokument, das ein Anwender aus dem Internet anfordert, sei es ein *html*-Dokument oder die Grafiken innerhalb dieser Seite, wird authentifiziert. Damit der Anwender nicht jedes Mal seine User-ID und das Passwort angeben muss, werden diese Daten nach der ersten Eingabe am Browser zwischengespeichert.) Versorgt ein solcher Proxy eine große Zahl von Anwendern mit dem Internetaccess, so muss der Directory Server unter Umständen eine hohe Zahl von Anfragen über *ldap* beantworten. Steht dieser Server an einem entfernten Ort und ist er nur über eine geringe Bandbreite an die Filiale angebunden, kann es zu Engpässen in der Authentifizierung kommen und der Internetaccess ist nicht mehr performant.

Besteht die Anforderung einer lokalen Replikation eines Datensatzes aus einem Masterverzeichnis, können verschiedene Architekturen zum Einsatz kommen.

- Das Replikat bekommt den kompletten Datensatz aus dem Masterverzeichnis übertragen oder nur einen Teil.
- Die Filiale erhält die Verantwortung über ihre eigenen Daten und kann Datenänderungen direkt vornehmen. Die Daten der anderen Zweige des DIT werden in die Filiale hineinrepliziert. (Wir haben eine solche Architektur bereits in Abschnitt 3.7.6 beschrieben.) Oder aber das Unternehmen entscheidet sich dagegen und organisiert eine komplett zentralisierte Replikationstopologie.
- Ein Master überträgt seine Daten auf ein Replikat, das seinerseits gegenüber einem oder mehreren anderen Servern die Stellung eines Masters wahrnimmt und die Daten weiterrepliziert (Abb. 3.14).

- Replikate werden entweder über Supplier oder Consumer Initiated Replication (SIR oder CIR, siehe Abschnitt 3.7.6) aktualisiert. Die Entscheidung darüber hängt von der Netzwerktopologie des Gesamtsystems ab.

Sind all diese Fragestellungen geklärt und hat sich dabei eine Replikationstopologie herausgebildet, kann man die nächsten Schritte in der Evaluierung einer Directory-Architektur angehen.

3.10.6 Sicherheitsfragen: der Zugriff auf die Daten

Überblick

Nachdem Sie nun einen Einblick in die Datenstrukturen eines Directory Servers gewonnen haben, beschäftigen sich die folgenden Abschnitte mit dem Zugriff auf die Daten.

Da User-Daten teilweise sensible Inhalte umfassen (User-Passwort, Personalstammnummer etc.), muss der Zugriff auf diese Daten kontrolliert werden. Directory Server gestatten die Definition von allgemeinen bis hin zu sehr speziellen Regeln, die entweder die Gesamtheit der Daten oder nur Teile oder einzelne Informationen betreffen.

Das Thema befasst sich mit folgenden Fragestellungen:

- Wie sieht der anonyme Zugriff auf die Daten aus? Welche Informationen sind für diese Art von Requests unkritisch?
- Wie werden die regulären Benutzer Daten abfragen und welche Informationen bekommen diese zu Gesicht? Können diese Benutzer fremde oder eigene Daten modifizieren?
- Welche besonderen Gruppierungen gibt es im Unternehmen, die eigene Zugriffsrechte einfordern? Welche Rechte haben insbesondere Administratoren?
- Werden die Informationen über verschlüsselte Datenverbindungen zwischen dem Server und dem Client ausgetauscht?
- Wie werden die Zugriffskontrollen im Directory Server umgesetzt? Welche Hierarchie besteht zwischen einzelnen Zugriffsrechten? Und welche Regel dominiert?

Zugriffsdefinitionen

Beim anonymen Zugriff ist das Verzeichnis vom Ansatz her ein offenes System. Das heißt, für jeden sind die freigegebenen Daten ohne besondere Authentifizierungsregeln frei zugänglich. Je nach Unternehmenspolitik sind die Standpunkte verschieden, was die Freigabe von Benutzerdaten betrifft.

Beispiel:

Betrachten wir die Richtlinien einer Beispielfirma und deren Umsetzung: Zunächst wird festgelegt, dass wir in meinefirma.com einen anonymen Zugriff erlauben. Der Zugriff soll nur für die internen Mitarbeiter freigegeben sein, das Verzeichnis wird keine Daten nach außen ins Internet weitergeben. Das ist in diesem Fall nur möglich, wenn der Zugriff aus dem Internet durch eine Firewall abgeschirmt ist. Die internen Benutzer können nun mit den verschiedensten Clients auf das Verzeichnis zugreifen. Ohne sich am Directory authentifizieren zu müssen, werden ihnen die Attribute »cn«, »sn«, »givenname«, »mail« und »telephonenumber« auf Anfrage bereitgestellt. Änderungen sind dagegen für alle Zugriffe dieser Art nicht möglich.

Als Nächstes wird beschlossen, dass Änderungen der eigenen Daten erlaubt sein sollen, um den administrativen Aufwand zu reduzieren. Insbesondere Daten wie das User-Passwort müssen dem Benutzer selbst in die Hand gegeben werden, damit er bei einer Änderung nicht jedes Mal den Administrator kontaktieren muss.

Authentifizierte Benutzer sollen mehr Informationen bekommen, als bei anonymen Zugriffen freigegeben werden. Für diesen Zugriff soll gelten, dass alles außer »userpassword« übermittelt wird.

Änderungen über die eigenen Modifikationen hinaus werden nur bestimmten Personengruppen erlaubt. Beispielsweise darf die Gruppe der Telefonzentrale die »telephonenumber« ändern, die Personalabteilung legt Wert auf die nahezu alleinige Änderungsberechtigung der Attribute »cn«, »sn«, »givenname« und »postaladdress« aller Benutzer. Administratoren sollen den kompletten Zugriff auf die Daten haben. So weit das Beispiel.

Der Zugriff auf einen Datensatz im Directory wird durch so genannte Access Control Items (ACI) geregelt, die in Access Control Lists (ACL) zusammengefasst werden. Die Implementierungen sind unterschiedlich und nicht standardisiert. Allen gemeinsam ist jedoch die Strukturierung der Zugriffsregeln in folgende Bereiche:

- Informationseinheiten

  Innerhalb des Datenstamms kann man über den Directory Information Tree verschiedene Informationseinheiten isolieren:

  - den vollständigen Directory Information Tree. Dieser beginnt mit der Wurzel und umfasst alle Daten darunter.
  - ein kompletter Zweig in einem Directory, beginnend mit einem Knoten, alle untergeordneten Knoten und Blätter.
  - ein dedizierter Knoten oder Blatt innerhalb des Baums; ein oder mehrere Attribute
  - sonstige Teilmengen des DIT (mit Hilfe eines spezialisierten Filters)

Jedes ACI hat damit einen Gültigkeitsbereich für eine bestimmte Informationseinheit. Diese wird im Rahmen der ACI-Syntax mittels *ldap*-Filter umgesetzt (siehe Abschnitt 13.1.3).

Beispiel:
Mit der folgenden Spezifikation wird der gesamte Baum des Directory adressiert:

```
((target="ldap:///o=meinefirma.com "))
```

Beispiel:
Mit der folgenden Spezifikation werden im gesamten Baum des Directory alle Daten mit Ausnahme der User-ID spezifiziert:

```
((target="ldap:///o=meinefirma.com ")(targetattr !="uid")
```

Hier ist noch nicht definiert, welche Zugriffsberechtigung für diesen Datensatz gelten soll. Dies wird durch die nächsten beiden Strukturen der ACI bereitgestellt.

- Service

  Es werden im *ldap* die folgenden Zugriffe spezifiziert:

  Read (Lesen)
  Write (Schreiben)
  Add (Hinzufügen)
  Delete (Löschen)
  Search (Suchen)
  Compare (Vergleichen)
  Selfwrite (Eigenen Eintrag beschreiben)

  Diese Zugriffe werden in diesem Teil entweder erlaubt (allow) oder verweigert (deny).

- Clients

  Im dritten und letzten Teil werden die Zugriffe auf die Clients angewendet, von denen der Request kommen kann. Dies kann verschiedenste Ausprägungen haben. Am häufigsten werden Spezifikationen nach UserDN oder GroupDN vorgenommen. Damit wird ein Benutzer anhand seiner Anmeldung identifiziert und für den Fall einer Gruppenregel auf Zugehörigkeit zu diesem Personenkreis identifiziert.

  Ein ACI kann beispielsweise festlegen, dass alle Benutzer der Gruppe »HR« (groupdn="ldap:///cn=HR,ou=groups,o=meinefirma.com") die in den Punkten »Service« und »Informations-Einheiten« festgelegten Zugriffsrechte bekommen.

  Diese so genannten »Bind-Rules« können aber noch andere Spezifikationen beinhalten:

  - Userdnattr
    Benutzer mit einem bestimmten Attribut (z.B. ou=Marketing)
  - Groupdnattr
    Gruppen, die z.B. zum Gebiet Europa (l=Europa) gehören

- IP
  Benutzer, die den Request von einer bestimmten IP-Adresse oder einem IP-Adressbereich aus durchführen
- DNS
  Anfragen aus einer bestimmten Domain
- Dayofweek
  Anfragen zu einem bestimmten Tag in der Woche
- Timeofday
  Anfragen zu einer bestimmten Tageszeit
- Authmethod
  Anfragen basierend auf einer bestimmten Art der Authentifizierung

Die Kombination der Komponenten »Informationseinheiten«, »Service« und »Clients« formen eine ACI, die ein Directory Server erkennt und umsetzt.

Hier ein Beispiel, wie eine ACI in einem Directory Server umgesetzt werden kann (das Beispiel entspricht der Nomenklatur des iPlanet Directory-Servers, die Implementierungen sind unter den Herstellern unterschiedlich):

```
aci: (target) (version 3.0; acl "name"; <permission> <bind_rules>;)
```

Eine Zugriffssteuerung, die es einem User erlaubt, alle seine Einträge zu ändern, würde damit folgendermaßen aussehen:

```
aci: (target="ldap:///uid=juser, ou=People, o=meinefirma.com")
(targetattr=*) (version 3.0; "Joe darf alle seine Einträge ändern";
allow (write) userdn="ldap///self";)
```

Diese Zeile wird bei diesem Hersteller als »Attribut/Wert«-Information in einem Directory-Eintrag hinterlegt.

Um das Ziel, den Zugriff auf die Daten im Verzeichnis zu reglementieren, zu erreichen, müssen die Regularien der ACIs genau verstanden und umgesetzt werden.

Umsetzung der Access-Control-Regeln

Ein Directory Server gehorcht im Allgemeinen der folgenden Vorgehensweise: Es wird eine Liste der verfügbaren ACIs erstellt, die den Zugriff regeln sollen. Diese Liste wird vom Directory Server in einer bestimmten Hierarchie geordnet, beginnend mit dem speziellsten ACI (in Hinblick auf die Daten, beispielsweise auf ein bestimmtes Attribut), dann über den Eintrag und die Knoten hinweg bis zu den allgemeinsten ACIs, die den gesamten Baum betreffen.

Diese Liste wird nun zweigeteilt, ohne die Hierarchie zu verändern: die Deny-Regeln auf der einen und die Allow-Regeln auf der anderen Seite. Anschließend durchläuft der Server bei einem Zugriff zunächst den Stapel der Deny-Liste, beginnend mit dem speziellsten ACI. Findet er hier eine Regel, die auf den Client-Request zutrifft, wird der Durchlauf abgebrochen und die Anfrage (als nicht bewilligt) ohne Ergebnismenge beantwortet.

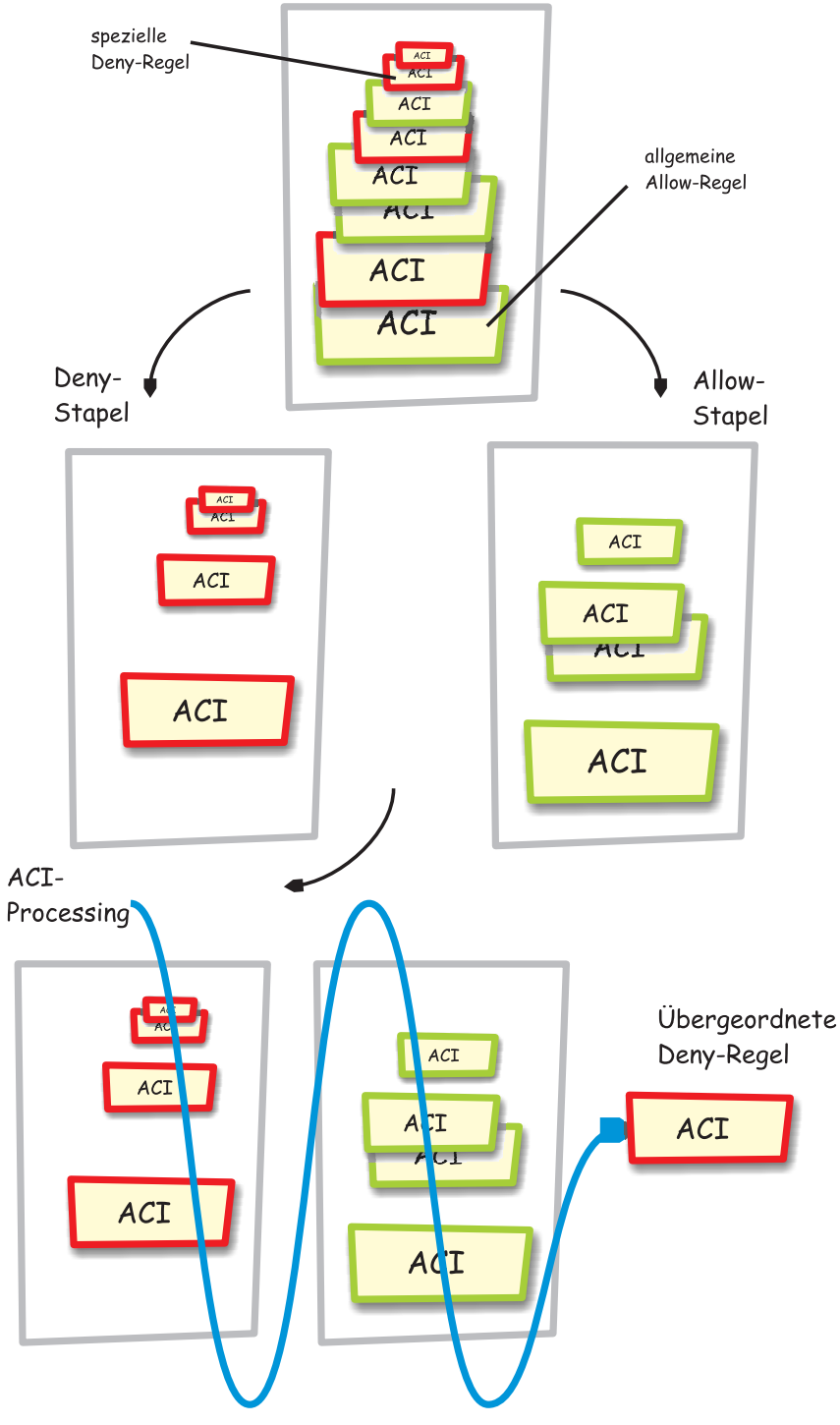


Abbildung 3.19: ACI-Processing eines Directory Servers

Trifft jedoch keine Deny-Regel zu, so wird der Durchlauf im Allow-Stapel fortgesetzt, wieder beginnend mit der speziellsten Regel. Findet sich hier ein ACI, das auf den Request zutrifft, wird der geforderte Datensatz gesucht und die Ergebnismenge zurückgeliefert. Durchläuft der ACI-Prozess allerdings erfolglos den Allow-Stapel, fällt er in eine Standard-Deny-Regel, die generell alles verbietet. Damit wird der Zugriff auf die Daten verweigert.

Diese Form der Zugriffssteuerung bedingt, dass im Falle eines vollkommenen Fehlens eines einzigen ACI im Verzeichnis generell keine Daten herausgegeben werden, weder an authentifizierte Benutzer noch an Administratoren. Erst mit dem gezielten Freischalten von Daten werden die Zugriffe beantwortet. Damit sind die meisten der eingesetzten ACIs in einem Verzeichnis von der Kategorie »Allow«. Anders gesagt, eine falsch gesetzte Deny-Regel kann jede folgende Allow-Regel aushebeln.

Eine Skizzierung des ACI-Processing ist in Abbildung 3.19 dargestellt.

Eine sorgfältige Definition des Zugriffs auf die Benutzerdaten muss zunächst auf Papier festgelegt und diskutiert werden. Erst anschließend wird man sich der Umsetzung mit dem genannten Regelwerk der ACIs widmen.

3.10.7 Indexierung der Daten

Abschließend sollte man einen Blick auf die Applikationen und insbesondere auf die Daten werfen, die von diesen angefordert werden. Gerade wenn man die Art der Anfragen kennt, kann man über eine geeignete Indexierung der *ldap*-Datenbank eine Menge an Performance gewinnen. Angenommen, eine Applikation wie zum Beispiel das webbasierte Telefonbuch kommt zum Einsatz. Hier ist vorherzusehen, dass Attribute wie »cn«, »sn« und »givenname« sowie »telephonenumber« und »mail« bevorzugte Ziele für eine Suche sind. Die meisten Hersteller von Verzeichnisdiensten bieten hierfür die Möglichkeit, die Datenbanken zu indexieren. In diesem Fall werden spezielle Unterdatenbanken vom Verzeichnisdienst angelegt, die nicht den gesamten Inhalt, sondern nur die im Index erfassten Attribute enthalten. Damit wird eine Suche über diese reduzierten Datenbanken wesentlich schneller und effizienter durchgeführt, als wenn der gesamte Datenstamm durchsucht werden muss.

Die Art, wie diese Unterdatenbanken angelegt werden, bestimmt man bei der Installation des Services. Kennt man die voraussichtlich häufigsten Anfragen, selektiert man die relevanten Attribute und erzeugt deren Indexe.

3.10.8 Testsystem, Pilotierung und Roll-out

Für die Implementierung eines zentralen User-Repository sollte man alle Phasen der Planung, der Datenkonsolidierung und der Pilotierung des Directory bis hin zum Roll-out durchlaufen, also:

- Aufbau eines Testsystems, in dem man mit dem Produkt vertraut wird und einzelne Integrations- und Performance- und Last-Tests durchläuft, bis man schließlich die dringendsten Probleme verstanden hat, die bei einer Neueinführung dieses Systems entstehen, und eine Lösung hierfür gefunden hat.
- Ausbau zu einer Pilotierung, die nicht den bisherigen produktiven Betrieb stört, aber bereits mit einigen Applikationen den Produktionsbetrieb des Systems unter realen Bedingungen testet.
- Roll-out, in dem alle vorgesehenen Applikationen auf das neue System übergehen. Gerade in dieser Phase ist eine Interaktion mit den Benutzern sehr wichtig, damit man bereits in einer frühen Phase mögliche Fehler identifizieren und beheben kann.

3.10.9 Einschränkungen

Der Vorteil eines zentralen User-Repository ist, dass nur eine einzige Benutzerdatenbank administriert werden muss. Ändert man die Daten hier, so arbeiten sofort alle angeschlossenen Dienste mit den aktualisierten Daten. Der administrative Overhead reduziert sich damit drastisch und man erhält eine verlässlichere Datenkonsistenz.

Die Einführung eines solchen Systems birgt aber auch andere Aspekte: Man muss sicherstellen, dass die Applikationen *ldap*-fähig sind, was nicht unbedingt vorausgesetzt werden kann. Im Gegenteil: Viele Applikationen unterstützen erst in den neuesten Releases *ldap*, bei anderen ist es nicht abzusehen, wann sie je dieses Protokoll unterstützen werden. Dies ist der Grund, warum sich in vielen Topologien eine reine Implementierung eines zentralisierten Directory-Dienstes nicht realisieren lässt. Man begnügt sich damit, die Zahl der zu administrierenden Datenquellen zu reduzieren.

Bei der Implementierung eines User-Repository verzichtet man bewusst auf den Funktionsumfang von relationalen Datenbanken. Man konzentriert sich vielmehr auf die Performance für die Datenlieferungen über ein standardisiertes Protokoll. Für die Datenstruktur sollten folgende Aspekte berücksichtigt werden:

- Die Einführung eines Verzeichnisdienstes setzt eine standardisierte Schematisierung der Attributsbegrifflichkeit voraus. Dies kann bedeuten, dass die verwendeten Attribute nicht auf die gegenwärtige Bezeichnung der Applikationen abgestimmt sind und es einer Anpassung bedarf. Die im Standardschema vorgesehene Begrifflichkeit umfasst eine Vielzahl von Beschreibungsmöglichkeiten, die schwerpunktmäßig auf die Beschreibung von Personen ausgelegt sind. Beschreibungen von Sachen oder Lokalisationen spielen eher eine untergeordnete Rolle. Nicht selten aber sind im eigenen Umfeld Attribute notwendig, die nicht Bestandteil des Standardschemas sind. Diesen Anforderungen begegnet man in einem Directory Server über Schemaerweiterungen, in denen man frei

nach den eigenen Anforderungen Attribute und Objektklassen (entweder von vorhandenen Objektklassen oder als neu definierte) ableiten kann.

- Directory Server enthalten nach Abschluss der rein technischen Installation keine Daten. Es handelt sich also hier nicht um einen »out-of-the-box« Dienst, den man unmittelbar nach Installation in Betrieb nimmt. Vielmehr sollte einer Implementierung eine sorgfältige Planungs- und Datenkonsolidierungsphase vorausgehen, in der man alle Informationsquellen berücksichtigt, die zum Datenstamm des Directory Servers beitragen.
- Gerade die Definition der Informationshoheiten kann zu politischen Diskussionen in einem Unternehmen führen. Daten, die von der einen Abteilung »schon immer« durchgeführt wurden, können von einer anderen Abteilung mit der gleichen Argumentation für sich beansprucht werden. Hier ergeben sich plötzlich Überschneidungen und Kompetenzstreitigkeiten, die vor der Einführung des Verzeichnisses beigelegt werden müssen.
- Ein Directory Server ist keine Datenbank im herkömmlichen Sinn. Vielmehr hält er Benutzerdaten vor, deren Charakteristik eher von kleinen Datensätzen bestimmt ist. Aufgrund der Natur der Daten sind die Einträge eines Verzeichnisses nicht volativ, das heißt, Änderungen erfolgen bei weitem weniger häufig als lesende Zugriffe. Der Dienst ist auf diese Anforderung hin abgestimmt und bietet nur geringe Performance, wenn die Hauptaufgabe im schreibenden Zugriff erfolgt.
- Directory-Einträge sind klein. Dieser Punkt macht neben nicht volativen Daten die Hauptperformance des Directory Servers aus. Werden dagegen Bilder oder Dokumente der Benutzer in das Verzeichnis gepackt, ist ein Zugriff auf diese Daten mit einem ftp-Download vergleichbar, der den Dienst lahm legt. Daher muss man entscheiden, welche benutzerrelevanten Informationen in die Liste der Daten für das Verzeichnis aufgenommen werden.

Ein Directory Service ist ein einfacher und leicht zu administrierender Dienst. Seine Vorteile zieht er aus der Performance, mit der er Daten über ein Netzwerkprotokoll liefern kann. Diesen Vorteil sollte man nicht dadurch verspielen, dass man Daten implementiert, die den positiven Effekt neutralisieren.

Beispiel:

Wenn ein Bild von einem Benutzer unbedingt im webbasierten Telefonbuch hinterlegt werden soll, dann muss dies nicht in der Datenbank des Directory erfolgen. Man kann dies auch über einen Link realisieren, der auf das Bild auf einer Webressource zeigt.

3.11 Zusammenfassung

Ein zentraler Benutzer-Verzeichnisdienst bringt eine Vielzahl von Vorteilen mit sich. Einer der augenfälligsten ist die zentralisierte Administration dieser Art von Daten. Gerade in großen Unternehmen erweist sich der Aufwand der Datenpflege bei einer verteilten Datenhaltung als sehr umfangreich. Mit jeder Anwendung, die diese Daten benötigt und nicht in der Lage ist, an einer zentralisierten Datenhaltung teilzunehmen, wird mehr und mehr Arbeitszeit in die Pflege von bereits vorhandenen Daten investiert.

In vielen Unternehmen ist der Directory Service nicht mehr wegzudenken. Viele Applikationen verlassen sich auf die Bereitstellung von User-Daten über das Netzwerk. Die zentrale User-Datenhaltung erleichtert die Administrierbarkeit und unterstützt die Datenkonsistenz. Ein standardisiertes Kommunikationsprotokoll vereinfacht die Datenverfügbarkeit. Eine Skalierung ist sowohl horizontal als auch vertikal unproblematisch. Gerade diese Vorteile machen einen solchen Dienst zu einer wichtigen Komponente im internen Netzwerk.

4 Mail

4.1 Einleitung

Die elektronische Nachrichtenübermittlung wurde bereits Ende des neunzehnten Jahrhunderts in Form der Telegrafie eingeführt. Mit Hilfe des Morsecodes wurden die Nachrichten verschlüsselt und schließlich über große Entfernungen hinweg übertragen. So gesehen stellt die elektronische Post einen der ältesten Dienste in der modernen Kommunikation dar.

Weiterentwicklungen ermöglichten immer schnellere und zuverlässigere Übertragungswege. Das reichte von Telex (das noch bis weit in die 80er Jahre eine nicht unbedeutende Rolle spielte) und mainframebasierten Informationsarchitekturen bis hin zu den ersten echten Netzwerksystemen, wie dem ARPANET, das als Vorläufer des Internets etwa 1971 seine Wurzeln hat. Hierbei bildeten sich eine Vielzahl verschiedener proprietärer Systeme heraus (Western Unions Easylink, U.S. Sprints Telemail, MCI Mail, AT&T Mail usw.). Benutzer dieser Dienste konnten innerhalb ihrer »Mailwelt« weitgehend problemlos miteinander kommunizieren, ein Austausch über verschiedene Systeme war, wenn überhaupt, nur über so genannte Gateways möglich, die die Nachrichten aus der einen »Welt« in die andere übersetzten.

Um den proprietären Ansätzen der verschiedenen Hersteller zu begegnen, entstand unter Mitarbeit von Jonathan B. Postel im Jahr 1981 das *smtp*-Protokoll, das 1982 in den RFCs 821 [6] und 822 [7] den Grundstein für eine standardisierte Mailimplementierung legte. Das primäre Ziel war es, über eine standardisierte Technologie eine stabile und verlässliche Mailarchitektur aufbauen zu können. Die Implementierung sollte offen sein, das heißt, ohne proprietäre und herstellerabhängige Eigenschaften auskommen.

1983 wurde mit RFC 882 [8] das Domain-Prinzip für Netzwerke gefordert. Zeitgleich kam mit »sendmail« der erste *smtp*-basierte Dienst zum Einsatz.

Nahezu parallel dazu entwickelte das »Consultative Committee for International Telegraph and Telephone« (CCITT), die spätere ITU (International Telecommunications Union), die erste Version des X.400-Standards. Dieser sollte die Übertragung von Nachrichten von einem beliebigen Sender an einen beliebigen Empfänger ermöglichen. Obwohl dieser Standard über das Jahr 1993 hinaus ständig weiterentwickelt und um eine Reihe von Features (starke Authentisierung, Replikation und vieles mehr) erweitert wurde, fand eine der ersten Versionen von 1984 immer noch die weiteste Verbreitung.

Das Bestreben nach einer Vereinheitlichung des Mailsystems stellte die Weichen für eine Technologie, wie wir sie heute kennen. Die Standardisierung ist

aber bis dato noch nicht abgeschlossen. Systeme wie Banyan und Exchange, die noch nicht diesen Standardisierungsbestrebungen gefolgt sind, arbeiten in vielen Bereichen noch immer proprietär und stellen Anforderungen an die Integration der übrigen Mailsysteme.

Dennoch haben sich die Bemühungen in Richtung einer einheitlichen Technologie mehr und mehr durchgesetzt und heute ist nahezu jedes Mailsystem in der Lage, eine E-Mail mit jedem beliebigen Mailaccount im Internet auszutauschen.

4.2 Überblick

Maildienste stellen eine einfache Art der asynchronen Kommunikation dar. Nachrichten werden über ein Textinterface verfasst, der Adressat wird angegeben und die Nachricht wird schnell und unproblematisch verschickt. Im Gegensatz zum traditionellen Briefwechsel ist man nicht auf Ressourcen wie Briefpapier, Marke und Kuvert angewiesen und man spart sich den Gang zum Postamt. Ein Mailserver nimmt eintreffende Mails entgegen und hinterlegt sie in einer so genannten Mailbox bzw. übernimmt den Transport von Mails an den Adressaten. Dieser erhält eine Mitteilung, dass eine Nachricht in seiner Mailbox auf die Abholung wartet. Alle Schritte dieses Nachrichtenaustauschs entsprechen denen des klassischen Postverkehrs. Gleiches gilt für die Erwartungshaltung beim Versenden von Nachrichten. Man verlässt sich einerseits darauf, dass die E-Mails garantiert übermittelt werden. Andererseits akzeptiert man, dass sie vom Empfänger nicht sofort entgegengenommen werden, wenn dieser nur in unregelmäßigen Zeitabständen seine Mailbox auf neue Nachrichten überprüft. Parallelen zum klassischen Postverkehr finden sich sogar bis zur Quittierung von Nachrichten: Genauso wie man bei einem Einschreiben den Empfang quittieren muss, kann man für eine E-Mail eine Bestätigung über ihr Eintreffen beim Empfänger einfordern. Leider ist diese Form der Empfangsbestätigung noch nicht beweiskräftig. Der Empfänger kann diese Quittierung verweigern und dennoch die Nachricht erhalten.

Eine weitere Parallele zwischen der klassischen und der elektronischen Form des Nachrichtenaustauschs betrifft die Dokumentation: Jede Nachricht kann gespeichert und verwaltet werden, um sie im Bedarfsfall wieder hervorzuholen. Mails, die im Eingangsordner des Mailaccounts eintreffen, kann man zwar sofort nach der Bearbeitung löschen, in der Regel werden sie aber eine Weile vorgehalten. In bestimmten Situationen ist es sogar notwendig, den gesamten Mailverkehr zu sichern, um Konflikte nachzuverfolgen. Nachrichten, die vor Wochen, Monaten oder sogar Jahren ausgetauscht wurden, können Aussagen belegen und Abmachungen nachweisen. Die einfache Verwaltung durch die Funktionen der verwendeten Mailclients unterstützt diese Dokumentation: Eingetroffene Nachrichten kann man nach Datum, Sender oder sonstigen Kriterien ordnen. Je nach Mailclient werden Suchfunktionen angeboten, die nach Schlüsselwörtern den gesamten Mailbe-

stand durchforschen. Nachrichten können in Unterverzeichnissen abgelegt werden, mit den entsprechenden Einstellungen lässt sich dies sogar über Mailfilter automatisieren.

Die E-Mail erfüllt damit in weiten Bereichen die gleiche Funktionalität, die der klassische Postversand bietet, sie ist aber ungleich schneller und erfordert weniger Aufwand. Daraus entwickelte sich eine sehr viel höhere Bereitschaft, Nachrichten per E-Mail zu versenden, als dies je beim Postverkehr der Fall war. Untersuchungen ergaben, dass mittlerweile mehr Nachrichten auf elektronischem Weg versendet werden als mit Papier und Briefmarke.

Die Einfachheit einer E-Mail hat noch weitere Konsequenzen: Kurze informelle Nachrichten lösen den ausführlichen Brief ab. Während man beim klassischen Brief so viel wie möglich an Informationen in einer Sendung unterbringen möchte, enthalten E-Mails häufig nur kurze Statements. Das führt aber nicht zwingend zu kurzen Mails: Über die »Beantworten«-Funktion wird in der Antwort auf eine Mail oft auch die ursprüngliche Nachricht des Absenders zitiert. Gehen nun einige Mails hin und her, können sich Dokumente entwickeln, die in ihrem Umfang stetig wachsen, weil sie sämtliche Nachrichtentexte sammeln.

Mit der Einführung der digitalen und multimedialen Form der Unterhaltungsindustrie halten auch Bilder, Musikclips und Videos Einzug in die Mailsysteme. Solche Dateien werden in Form von »Attachments« per E-Mail verschickt. Mittlerweile können Mailclients diese Daten im Gesamtkontext der Mail identifizieren und entsprechende Viewer starten, um die Datei darzustellen. So werden Bilder automatisch als solche in einer Mail interpretiert und dargestellt, *html*-Dokumente in der Mail starten ohne weiteres Zutun des Anwenders automatisch den Browser, Word-Dokumente werden direkt der entsprechenden Applikation überstellt, um sie zu öffnen und zu bearbeiten.

Solche Attachments bergen aber auch Risiken, denn diese Dateianhänge können zur Verbreitung von Computerviren dienen. E-Mails haben sich zum typischen Übertragungsweg für digitale Schädlinge entwickelt. Hier entstehen vielfach Lücken in den Sicherheitskonzepten der Unternehmen, aber auch der Privatanwender ist betroffen, wenn er ohne Vorsichtsmaßnahmen Attachments öffnet.

Dieses Kapitel befasst sich mit der Architektur eines Mailsystems und beschreibt die Grundfunktionalität der einzelnen Komponenten.

E-Mail: Ein Kommunikationsmittel zwischen zwei oder mehreren Personen, das neben reinem textbasierten Informationsaustausch die Übertragung multimedialer Dateien gestattet.

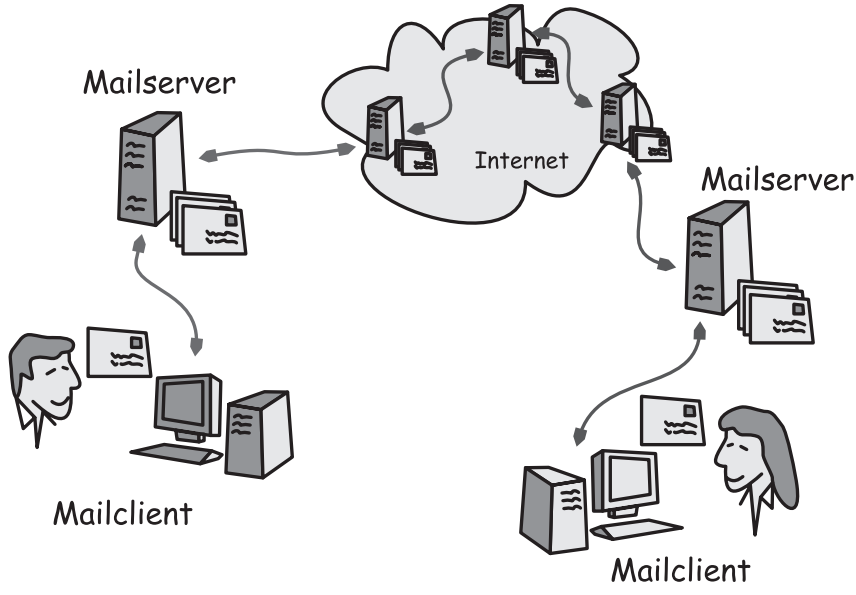


Abbildung 4.1: Internetbasiertes Mailsystem

4.2.1 Komponenten eines Mailsystems

Ein Mailsystem ist ein Kommunikationsmedium zur Übermittlung von Nachrichten in asynchroner Weise von einem Sender zum Empfänger. Auf dem Weg dorthin durchläuft die Mail verschiedene Phasen der Weiterleitung und des Zwischenspeicherns. Die folgende Beschreibung der einzelnen Komponenten eines Mailsystems konzentriert sich auf die gebräuchlichste Architektur, die *smtp*-basierte Übertragung und deren Standards.

E-Mail

Eine Nachricht, die in elektronischer Form geschrieben wurde und mit den Mitteln der Datenübertragung über das Netzwerk bis zum Adressaten weitergeleitet wird, bezeichnet man als E-Mail.

Es gab und gibt heute immer noch eine Reihe von unterschiedlichen Mailformaten. Der erste Ansatz für eine Standardisierung erfolgte bereits 1982 in RFC 822 [7]. Hierin wurde festgelegt, wie das Datenformat für eine ARPA-Textnachricht auszusehen hat. Um der multimedialen Entwicklung zu folgen, wurden Erweiterungen hierzu in RFC 2045 [9] und 2046 [10] veröffentlicht.

Im Rahmen des Themas »E-Mail« gibt es eine Reihe von Begriffen, die immer wieder auftauchen und im Folgenden beschrieben werden.

- **smtp Envelope**

  Jede Mail wird in eine Art elektronischen Umschlag verpackt, in dem sie von einem Mailserver zum nächsten übermittelt wird. Dieser Umschlag wird vom sendenden Server erzeugt und vom empfangenden Server

wieder entfernt. Er ist nicht zu verwechseln mit den Header-Informationen in der Mail selbst, deren Inhalte keinen Einfluss auf das Routing der Mail haben und nur die Informationen der einzelnen Schritte beim Übermitteln der Mail durch das Netzwerk protokollieren.

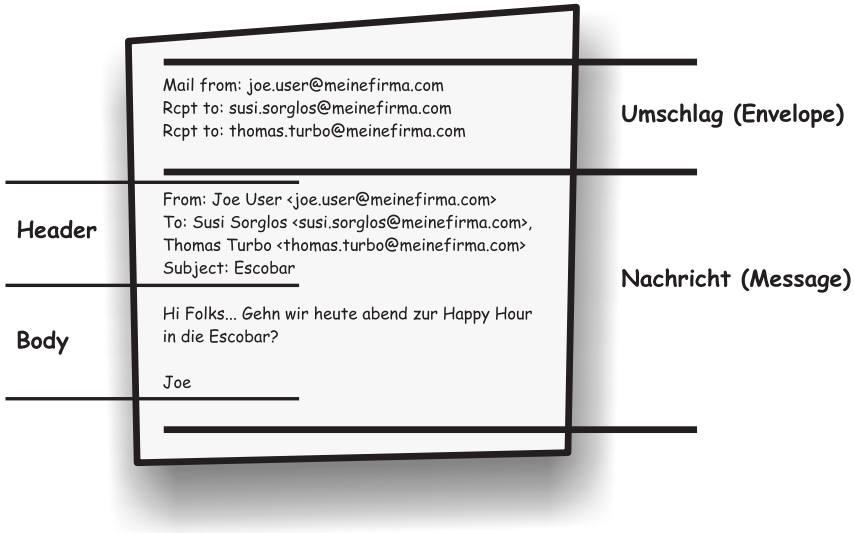
- **Header/Body-Informationen**

  Eine Mail besteht aus zwei verschiedenen Bereichen, dem Header (Kopfbereich) und dem Body (Informationsbereich). Header-Informationen haben nur eine protokollierende Funktionalität und informieren den Adressaten über Herkunft und Route der Nachricht, die ihn erreicht hat.

  Im Folgenden sehen Sie eine Liste verschiedener Informationen, die im Mailheader zu finden sind:

  - received: <Server-Name>
    Diese Information wird vom *smtp*-Server jeweils zu Beginn der Übertragung eingefügt.
  - from: <Absender-Adresse>
    Adresse des Absenders.
  - to: <Ziel-Adresse(n)>
    Adresse(n) der Empfänger.
  - cc: <Empfänger-Adresse(n)>
    Kopie der Nachricht an einen oder mehrere andere Empfänger (carbon copy). Diese Header-Zeile ist für den Originalempfänger der Nachricht zu sehen.
  - bcc: <Empfänger-Adresse(n)>
    Blindkopie an einen anderen Empfänger (blind carbon copy). Nach dem Versenden der Mail werden diese Einträge aus dem Header der Nachricht entfernt. Dadurch bleiben diese Empfängeradressen den anderen Empfängern der Nachricht verborgen.
  - subject: <Betreff>
    Betreffangaben. Hier kann man den Inhalt der Mail in kurzen Stichworten beschreiben.
  - date: <Zeitstempel> Datum und Uhrzeit, meist mit Zeitzone (0000=GMT, +0001=MEZ, +0002=MESZ...).
  - reply-to: <Adresse>
    Gewünschte Antwortadresse, falls diese Adresse vom »From«-Feld abweicht.
  - message-id: <MailID>
    Eindeutige Mail-ID zum Kennzeichnen der Nachricht.
  - in-reply-to:
    Kennzeichnung der Mail-ID bei Beantwortungen.
  - x-...:
    Herstellerspezifische Zusatzangaben.

Abbildung 4.2: Aufbau einer E-Mail



- **base64**

  Die Implementierungen der Mailservices sind standardisiert auf ASCII-Basis, das heißt, es werden textbasierte Informationen übermittelt. Mittlerweile wächst allerdings der Bedarf, neben den reinen textorientierten Nachrichten Binärdaten wie zum Beispiel Bilder zu übermitteln. Diese Daten in Binärform sind für den Transport per E-Mail nicht geeignet. Um dennoch eine Übermittlung durchführen zu können, werden die Binärdaten zunächst in ASCII-Daten »übersetzt«. Erste Implementierungen dieser Art funktionierten auf Basis der Unix-Befehle »uuencode« und »uudecode« (Unix to Unix encode/decode), neuere Implementierungen nutzen das verbreitetere »base64-encoding«. Dabei werden jeweils 6 Bits einer beliebigen Datei (binär oder ASCII) wie folgt verschlüsselt (6 Bits = 2^6 = 64 verschiedene Zeichen):

```
 0-25:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
26-51:   abcdefghijklmnopqrstuvwxyz
52-63:   0123456789+/
```

  Mit dieser Technik kann jede Datei in das ASCII-Format umgewandelt werden. Umgekehrt kann jede base64-verschlüsselte Datei eindeutig in die ursprüngliche Binärform zurückverwandelt werden. Die base64-kodierte Datei wird in Zeilen von maximal 76 Zeichen hinterlegt. Bei der Kodierung einer Binärdatei erhöht sich das Volumen der ursprünglichen Datei um ca. 33%.

- **MIME**

  Das Akronym »MIME« steht für »Multipurpose Internet Mail Extensions« und definiert, wie sich in einer Mail Binärdaten hinzufügen lassen,

so dass sie vom Mailsystem übermittelt und vom MIME-kompatiblen Mailclient richtig interpretiert werden. Die Implementierung ist standardisiert, die Definitionen sind in RFC 2045 [9] und 2046 [10] dokumentiert.

Um die Übermittlung per Mail für non-ASCII-Dateien zu ermöglichen, werden die Binärdaten über base64 kodiert und damit in ein Mail-konformes Format gebracht.

Die Client-Software, die diese Mail erhält, sammelt die Daten und versucht sie darzustellen. Ohne weitere Informationen würden die Texte zwar richtig, die Binärdaten allerdings nur base64-kodiert dargestellt. Dieses Manko wird über Zusatzinformationen in der Mail behoben, die den Inhalt kennzeichnen und der Client-Software mitteilen, wie diese Daten zu interpretieren sind.

Dieses Verfahren wird nicht nur in Mails eingesetzt. Später wird noch beschrieben, dass sich auch für News eine solche Problematik ergibt, die ebenfalls mit MIME gelöst wird.

Um dem Client mitzuteilen, was eigentlich in der Mail steht, werden die Bezeichner »Content-Transfer-Encoding« und »Content-Type« verwendet:

```
Received: from inbound.meinefirma.com (inbound [123.45.67.8]) by
maildrop.meinefirma.com …

…

From: Other Boss <other.boss@someothercompany.com>
Return Path: Other Boss <other.boss@someothercompany.com>
Message-ID: <abe7caaaca.aaacaabe7c@someothercompany.com>
X-Mailer: Netscape Webmail
MIME-Version: 1.0
Content-Language: de
Subject: Ergebnisse 2001
X-Accept-Language: de
Content-Type: multipart/mixed; boundary="--52f842b96502f81"
Content-Length: 1779

This is a multi-part message in MIME format.
----52f842b96502f81
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
Content-Transfer-Encoding: 7bit

Hallo Andreas, hier die Webseite mit den Umsatz-Ergebnissen.
Viel Spass, Mark
----52f842b96502f81
Content-Type: image/gif
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Image.GIF"
```

```
R0lGODlhJwAoAPcAAGNjnHNzpYyMtaWlzsbG3s7O/+fn7+/v9/////////////////
/////+e8KaYVWf+
5dNd9AsS1kXGGdnYUQcgwJIBlwBxKk3kISFlbRhTgJlOFHFeLO4UQcdmihYweE
aOKKLLboYosBAQA7
```

----52f842b96502f81--

Die Informationen im Feld »Content-Type« übermitteln, um welche Daten es sich in der Nachricht handelt. Im vorliegenden Beispiel ist einer der Inhaltstypen »text/plain«, was dem einfachen Klartext entspricht. Der Bezeichner »text« definiert den Typ, »plain« ist eine Unterklassifizierung, beide Attribute werden durch ein »/« getrennt.

Die Abgrenzungen der einzelnen MIME-Attachments werden über einen eindeutigen »boundary«-Wert gekennzeichnet, der im ersten »Content-Type«-Feld der Mail definiert wurde. Anhand dieses Markierungsstrings kann der Mailclient die verschiedenen Inhalte der Mail voneinander unterscheiden.

Im Folgenden sehen Sie eine Übersicht von verbreiteten Werten für das Content-Type-Feld:

text/plain	Rein textbasierter Inhalt
text/html	Daten im *html*-Format
image/jpeg	Bilderdaten im »jpg«-Format
image/gif	Bilderdaten im Graphics-Interchange-Format (»gif«-Format)
application/ps	Ein PostScript-Dokument
audio/midi	Audio-Dateien im »midi«-Format
application/octet-stream	Unbekanntes Format

Daneben gibt es noch eine Reihe weiterer Formate, die im Internet verwendet und hier nicht aufgelistet werden.

Mit der Charakterisierung »Content Transfer Encoding« wird dem Mailclient vermittelt, wie die Daten gepackt sind. Über diese Schlüsselwörter kann der Client die Nachrichten durchforsten (»parsen«). Entsprechend markierte Bereiche werden als Dokumente besonderen Formats erkannt und der Mailclient kann diese Daten individuell darstellen. Ein eingebettetes Bild wird beispielsweise über den »Content-Type: image/gif« erkannt und kann über einen gif-Interpreter als grafischer Bestandteil der Nachricht dargestellt werden.

Der MIME-Standard umfasst noch eine Reihe weiterer Content-Beschreibungen, die in RFC 2045 [9] und 2046 [10] dokumentiert sind und auf die hier nicht näher eingegangen werden soll.

4.2.2 Mailclient/User-Agent

Ein Mailclient, auch User-Agent genannt, ist ein Programm, das dem Benutzer die Möglichkeit gibt, Mails zu lesen, zu verwalten und/oder zu schreiben. Der Markt bietet eine breite Palette von verschiedenen Clients. Die Bandbreite reicht von reinen textbasierten Programmen wie »mail« und »elm« in der Unixwelt bin hin zu den multimedialen Alleskönnern wie Microsoft Outlook, Netscape Messenger, Qualcomm Eudora, Mulberry, Pegasus Mail und vielen anderen. Welcher Client verwendet wird, ist in der Regel eine Frage des persönlichen Geschmacks oder der internen Firmenvorgabe.

Je nach Konfiguration verbindet sich der Mailclient manuell oder automatisch nach einer festgelegten Zeit mit dem Server, zeigt die eingetroffene Mail an und lädt sie gegebenenfalls auf den Rechner des Anwenders.

Die Verbindungen zwischen Mailserver und -client sind in der Regel nicht von permanenter Dauer, sondern werden nach einer bestimmten Zeit wieder abgebaut. Dies geschieht deshalb, weil ein Mailserver nur eine bestimmte Zahl von Sessions parallel aufrechterhalten kann. Sobald diese Grenze erreicht wird, ist der Dienst nicht mehr erreichbar. Um dies zu vermeiden, werden die Mailverbindungen nach einer bestimmten Zeit der Inaktivität (»idle«) wieder getrennt.

4.2.3 Message Transfer Agent

Ein Message Transfer Agent (MTA) ist der »Zustelldienst«. So wie wir im wirklichen Leben einen Brief bei der Post aufgeben, so werden die Mails, die der Benutzer mit Hilfe des Mailclients schreibt, am MTA »aufgegeben«. Das Weiterleiten der elektronischen Nachricht erfolgt dann durch diesen virtuellen Postdienst, wobei der Absender der Mail nicht mehr in diesen Prozess involviert ist.

Der interne Bearbeitungsweg der E-Mail führt über weitere MTAs, bis der Bestimmungsort erreicht ist. Im E-Mail-Umfeld spricht man bei der Weiterleitung von »Routen«: Eine Mail wird durch das Netzwerk geroutet.

4.2.4 Mailbox

Eine Mailbox ist der Briefkasten eines jeden Benutzers. Die Mails aus dem Internet oder auch aus dem internen Netzwerk werden hier hinterlegt. Der Anwender kann dann die eingegangenen Mails über einen Mailclient aus der Mailbox abrufen. Damit der Zugriff auf die Mails geschützt ist, muss sich der Anwender gegenüber seinem elektronischen Postfach über User-ID und Passwort ausweisen.

Die Größe einer Mailbox lässt sich durch eine so genannte Quota begrenzen, um zu verhindern, dass im Laufe der Zeit die Speicherkapazitäten des Mailservers überbeansprucht werden. Dies ist insbesondere dann wichtig, wenn einzelne Anwender über einen längeren Zeitraum ihre Nachrichten nicht

abholen. Nach Erreichen der vorgegebenen Quota werden weitere Nachrichten mit einer Fehlermeldung abgewiesen.

Der Mailserver hält eine Mail meist so lange vor, bis der Benutzer sie abholt. Bei bestimmten Services kann es vorkommen, dass die Nachrichten »altern«. Wenn der Benutzer seine Nachrichten nach einer bestimmten Zeit noch nicht abgeholt hat, werden sie automatisch gelöscht. Diese Form des Alterungsprozesses ist optional, viele Dienste gestatten eine zeitlich unlimitierte Speicherung von Nachrichten. Welche Form der Mailverwaltung man implementiert, hängt vom eingesetzten Mailsystem und den Umgebungsvorgaben ab. Ein freier Maildienst im Internet, bei dem jeder nach Lust und Laune einen Mailaccount anlegen kann, wird wahrscheinlich den Speicherplatz beschränken. Bei Mailsystemen in Unternehmen dagegen verzichtet man meist auf eine solche Art der Reglementierung, da die Gefahr zu groß ist, unternehmenswichtige Daten zu verlieren.

4.2.5 Mailqueue

Die Liste der Mails, die die Anwender schreiben und an den Mailserver zur Weiterleitung übergeben, bezeichnet man als Mailqueue. Im normalen Betrieb ist die Queue leer, weil der Server üblicherweise sofort die Mail an die Zieladresse weitergibt. Bei Netzwerkproblemen kann es allerdings dazu kommen, dass der Server nicht mehr in der Lage ist, Mails zuzustellen. Die Queue füllt sich mit Nachrichten, die auf ihre Auslieferung warten. Der Anwender merkt zunächst nichts von diesem Problem, da der Dienst ja nach wie vor zur Verfügung steht. Die Mails, die auf die Auslieferung warten, füllen mit der Zeit die Queue, die als physikalischer Speicher nur eine begrenzte Größe hat. Somit muss ein Administrator, nachdem er feststellt, dass sich die Queue mit nicht ausgelieferten Mails füllt, relativ zügig den Fehler beheben. Geschieht dies nicht, stellt der Mailserver seinen Dienst zunächst ein, bis neuer Platz zur weiteren Speicherung von nachfolgenden Mails in der Queue vorhanden ist. Dies geschieht im Allgemeinen dann, wenn der Fehler behoben ist.

4.2.6 Mailmultiplexer

Der Anbieter eines Internetservice übergibt seinen Anwendern die Zugangsdaten für ihren Mailzugang. Mit Ausnahme der Account-Daten (User-ID und Passwort) sollten alle anderen Konfigurationsdaten hierfür einheitlich sein. So gibt es einen Mailserver-Hostnamen, an den ausgehende Mails versendet werden, und einen Servernamen, in dem der User seine Mailbox erwartet.

Sobald für immer mehr Anwender jeweils eine Mailbox eingerichtet werden muss, reicht eine einzelne Hardware-Plattform nicht mehr aus. Die Mailboxen müssen über mehr als einen Rechner verteilt werden. Dies würde für die Anwender bedeuten, dass sie den jeweiligen Hostnamen wissen müssten,

von wo sie ihre Mails geliefert bekommen. Dies aber möchte man vermeiden, da erstens eine weitere Konfigurationsmöglichkeit für den Anwender eine Problemquelle bedeutet und zweitens ein Sicherheitsrisiko darstellt.

Dieses Problem löst man, indem man einen Mailmultiplexer installiert, der lediglich die Aufgabe hat, die Benutzer auf den »richtigen« Mailserver zu lenken.

Beispiel:

Ein Internet Provider richtet für den Mailbox-Zugriff seiner Kunden einen Mailserver mit dem Namen »pop3.meinprovider.de« ein. Diese Konfigurationsinformation wird auf allen Mailclients der Kunden eingetragen. Im Zuge der Neustrukturierung des Rechnerpools aufgrund des rasanten Wachstums des Kundenstamms bedarf es einer Verteilung der Benutzer auf pop3-1.meinprovider.de, pop3-2.meinprovider.de und pop3-3.meinprovider.de. Für die Kunden würde dies eine Umkonfigurierung ihrer Clients-Programme bedeuten, je nachdem, auf welchem Server die jeweilige Mailbox des einzelnen Users eingerichtet ist. Das Resultat wäre, dass neben dem enormen administrativen Aufwand eine Flut von Helpdesk-Anfragen auf das Unternehmen zukäme.

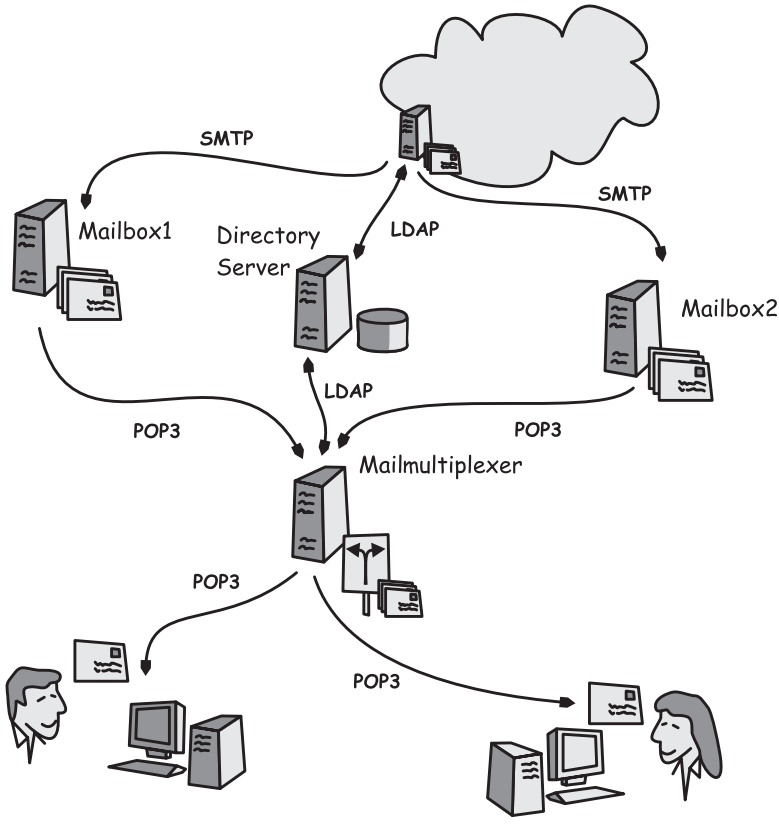


Abbildung 4.3: Architektur einer verteilten Mailbox-Topologie

Um dieses Szenario zu umgehen, kann das Unternehmen einen Mailmultiplexer mit dem Namen »pop3.meinprovider.de« einrichten, der den Mailzugriff des Benutzers entgegennimmt, über einen Directory Server den wahren Namen der Mailbox (pop3-1, pop3-2 oder pop3-3) des Anwenders erfährt, die Nachrichten abholt und sie dem Benutzer zur Verfügung stellt. Diese Stellvertreter-Funktionalität wird mit dem Begriff »Proxy« beschrieben. Der Multiplexer ist ein Mailproxy für pop3. Er stellt für die User den Adressaten für den Mailzugriff dar, gegenüber den Mailboxen nimmt er die Stellung eines Mailclients ein.

4.3 Kommunikationsprotokolle

Für die Kommunikation der einzelnen Komponenten werden je nach Aufgabe besondere Protokolle eingesetzt:

1. Zwischen den Servern

   In Standardumgebungen wird zum Übertragen der Mails von einem MTA zum anderen das »Simple Mail Transfer Protocol« (*smtp*) und dessen Erweiterung *esmtp* verwendet.

2. Zwischen Client und Server wird mehr als ein Protokoll eingesetzt:

   pop3: Das »Post Office Protocol Version 3« ist das am häufigsten verwendete Protokoll, um Mails von der Mailbox auf den eigenen Rechner zu laden. Der Client stellt eine Verbindung zum Mailserver her und lädt alle Nachrichten auf den Rechner des Anwenders herunter.

   imap4: Das Internet Message Access Protocol Version 4 ist eine weitere Möglichkeit, die Mails zu lesen. Die wichtigsten Unterschiede zum *pop3* sind, dass die Mails auf dem Mailserver belassen werden können und man sich bei der Verbindung zu seiner Mailbox nur die Header-Informationen angeben lassen kann. Dadurch erspart man sich unter Umständen den aufwändigen Download von Nachrichten, die man eigentlich gar nicht empfangen möchte. Auch kann die Mailbox (Unter-)Verzeichnisse enthalten, was die Verwaltung der eingetroffenen Nachricht erleichtert.

   smtp: Nachdem eine Mail geschrieben wurde, baut ein Mailclient-Programm eine Verbindung mit dem Server auf, der für die ausgehenden Nachrichten verantwortlich ist, und versendet über *smtp* die Mail. Im Unix-Umfeld kommt bisweilen auch das Unix-Delivery zum Einsatz, bei dem über *smtp* Mails an den Client ausgeliefert werden.

   http: Das »Hypertext Transfer Protocol« ist kein typisches Mailprotokoll, kommt aber im Mailumfeld zum Einsatz, wenn man über einen so genannten Webmail-Account seine Nachrichten empfängt und versendet. Die Nachrichten werden auf Seiten des Clients in einem *html*-Umfeld verfasst und dargestellt, die Daten werden vom Client zum Webmail-Service über *http* versandt und gelangen erst dort in die »wirkliche« Mail-

welt. Über Webmail wird es möglich, ohne explizit konfigurierten Mailclient über einen einfachen Browser einen Zugang zur persönlichen Mailbox zu bekommen. Dies kann dann unter Einbeziehung aller Sicherheitsvorgaben im Prinzip von jedem öffentlichen Webterminal aus geschehen.

Abbildung 4.4 zeigt noch einmal die verwendeten Protokolle.

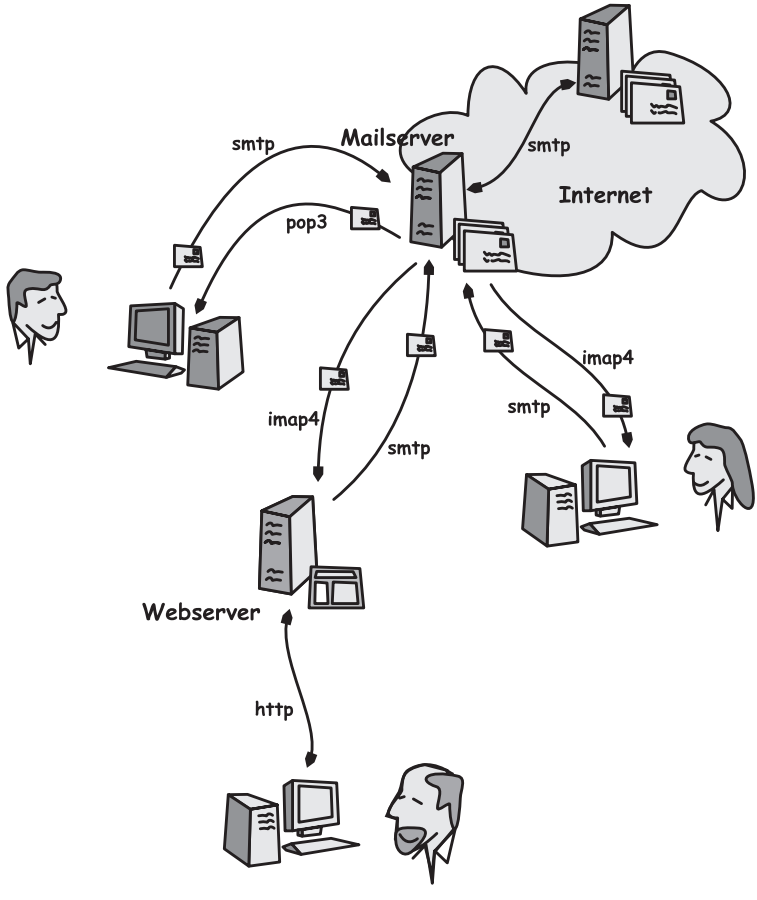


Abbildung 4.4: Protokolle im Mailsystem

Im Folgenden werden die einzelnen Protokolle detaillierter beschrieben.

4.3.1 pop3

Überblick

Der *pop3*-Dienst läuft als eigener Prozess auf einem Mailserver, internetweit hat man sich auf die Verwendung der Portnummer 110 für diesen Service geeinigt. Alle Details der *pop3*-Implementierung sind in RFC 1939 [12] beschrieben.

Wenn der Anwender über *pop3* mit seinem Mailclient eine Verbindung zu seiner Mailbox aufbaut, werden alle eingegangenen Nachrichten über dieses Protokoll auf den Rechner des Users übertragen. Die Mails werden anschließend auf Seiten des Mailservers gelöscht (die meisten Mailclients löschen die übertragenen Mails automatisch, *pop3* erzwingt dies jedoch nicht). Dabei geschieht die Übertragung ohne Interaktion mit dem Anwender: Er kann sich nicht aussuchen, welche Nachricht er empfangen möchte und welche nicht. Nachdem die Mails übertragen wurden, wird die Verbindung zum Mailserver abgebaut und der Anwender hat alle seine Nachrichten auf seinem lokalen Rechner zur Verfügung. Damit sind diese nach dem Abbau der Verbindung ins Internet auf Seiten des Clients vorhanden und können offline gelesen und bearbeitet werden.

pop3 ist ein sehr einfaches Protokoll. Es wird über TCP/IP übertragen und ist nicht für umfangreiche Operationen auf dem Mailserver gedacht. Die meisten Aktionen beschränken sich darauf, die Nachrichten über den Mailclient an den Benutzer zu übertragen, eine weitere Vorhaltung der bereits empfangenen Nachrichten auf Seiten des Servers ist nicht vorgesehen.

pop3-Befehle

Eine *pop3*-Session wird über ein TCP/IP-Netzwerk initiiert, es werden Befehle und Antworten über diese Session ausgetauscht, bis die Verbindung abgebaut wird. Man kann diese Befehlssequenzen über telnet auf Port 110 nachstellen. Wir werden dies exemplarisch in den folgenden Abschnitten tun, um einen Eindruck davon zu vermitteln, wie eine *pop3*-Sitzung eines Mailclients mit dem Server umgesetzt wird.

Das *pop3*-Protokoll hat eine bestimmte Syntax, die über ASCII-Befehle umgesetzt und über »+OK« oder »-ERR« quittiert werden. Im Folgenden sehen Sie ein Beispiel, wie ein Mailserver auf einen *pop3*-Account zugreift.

Man unterscheidet die Initiierungs-, die Authentifizierungs-, die Transaktions- und die Update-Phase.

1. **Initiierungsphase**

   Die Verbindung zum *pop3*-Server wird aufgebaut:

   ```
   >telnet pop.meinefirma.com 110
   +OK pop.meinefirma.com pop3 service (1231.12423@popserv1)
   ```

   Hiermit ist der erste Kontakt über das Netzwerk zum *pop3*-Server auf Port 110 hergestellt und der Mailserver hat eine kurze Begrüßung übermittelt.

2. **Authentifizierungsphase**

   In der darauf folgenden Authentifizierungsphase muss sich der Anwender gegenüber seiner Mailbox als rechtmäßiger Eigentümer ausweisen. Dies geschieht durch Übertragung von User-ID und Passwort:

```
user iamtheboss
+OK Name is a valid mailbox
pass nobodyknows
+OK mailbox has 4 messages (5737 octets)
```

Damit ist die Authentifizierungsphase erfolgreich abgeschlossen, es folgt die Transaktionsphase.

3. **Transaktionsphase**

Im vorliegenden Beispiel kann man als Nächstes über den »stat«-Befehl den Status der Mailbox anzeigen lassen:

```
stat
+OK 4 337160
```

Der Server liefert als Ergebnis die Anzahl der vorliegenden Mails und den damit belegten Speicherplatz (in Byte) auf Seiten des Mailservers.

Mit »list« erfolgt eine Auflistung der vorhandenen Nachrichten auf dem Mailserver:

```
list
+OK scan list follows
1 3984
2 1342
3 411
4 331423

.
```

Die Liste weist pro Zeile die Nummer und die Größe in Byte der jeweiligen Mail auf.

Mit »top <n><m>« lässt sich insbesondere bei einer sehr langen Mail feststellen, was in den ersten <m> Zeilen der <n>-ten Nachricht geschrieben steht:

```
top 3 9

+OK message follows

Return path: <other.boss@someothercompany.com>

Delivered-To: meinefirma delivery to iamtheboss@meinefirma.com

Received: from someother.company.com (194.64.51.223) by
mx0.meinefirma.com (mx0) with smtp; 30 Mar 2002 17:45:19 -0000

Date:30 Mar 2002 13:29:28 -0000

Message-ID: <20020330132928.23878@ smtp.someothercompany.com >

From:  other.boss@someothercompany.com

To:    Andreas Arbeitgeber <iamtheboss@meinefirma.com>
```

```
Subject: Golfpartie

Hallo Andreas...
```

Um die gesamte <n>-te Mail angezeigt zu bekommen, wird der Befehl »retr <n>« verwendet:

```
retr 2

+OK 1342 octets

Return path: <other.boss@someothercompany.com>

Delivered-To: meinefirma delivery to iamtheboss@meinefirma.com

Received: from someother.company.com (194.64.51.223) by
mx0.meinefirma.com (mx0) with smtp; 30 Mar 2002 17:45:19 -0000

Date: 30 Mar 2002 13:29:28 -0000

Message-ID: <20020330132928.23878@smtp.someothercompany.com >

From:other.boss@someothercompany.com

To: Andreas Arbeitgeber <iamtheboss@meinefirma.com>

Subject: Golfpartie

Hallo Andreas...

Ich hoffe, es bleibt bei unserer Golfpartie am Dienstag um 14h....

Gruss

Mark

.
```

Um eine Nachricht zu löschen, steht der Befehl »dele« zur Verfügung:

```
dele 2
+OK message deleted
```

Die zu löschenden Mails werden erst nach dem Beenden der Session aktiv entfernt. Wird innerhalb der Verbindung der Befehl »rset« verwendet, so sind die vorher als zu löschen markierten Mails wieder vorhanden.

4. **Update-Phase**

   Mit »quit« geht man in die so genannte Update-Phase, das heißt, die Verbindung zum Mailserver wird beendet und die als zu löschen markierten Mails werden aktiv vom Mailserver entfernt.

Obige Darstellung ist nur ein Beispiel, jeder Hersteller implementiert die Antworten des Servers etwas unterschiedlich. Die grundlegende Liste der Befehle sind aber standardisiert und werden von allen *pop3*-Servern gleichermaßen interpretiert.

Hier eine Liste der in RFC 1939 festgelegten Befehle für die Verwendung des *pop3*-Protokolls zum Verwalten von Mails:

QUIT	Beendet eine *pop3*-Session mit einem Mailserver.
STAT	Informiert den Mailclient über Anzahl und Gesamtgröße der bereitliegenden Mails.
LIST	Listet dem Mailclient die einzelnen Mails nach Nummer und Größe auf.
RETR <n>	Lädt dem Mailclient die Mail mit der Nummer <n> aus der Liste der vorliegenden Mails auf.
DELE <n>	Löscht die Mail mit der Nummer <n>.
NOOP	Der Mailserver antwortet mit einer Meldung, ohne eine weitere Aktion durchzuführen.
RSET	Die Löschmarkierungen für einzelne Mails werden komplett aufgehoben.
TOP <n><m>	Die ersten <m> Zeilen der <n>-ten Mail werden angezeigt.
UIDL	Es wird eine Liste der Unique IDs der Mails angezeigt, die über den gesamten Mailbestand eines Mailservers eindeutig ist und die aus dem Hash einer Nachricht gebildet wird.
USER	Die User-ID wird übermittelt.
PASS	Das Passwort wird übermittelt.
APOP	Eine alternative Methode zur Authentifizierung neben dem Austauschen von User-ID und Passwort in Plaintext: Die Accountdaten werden bei der Übertragung verschlüsselt. Die Syntax zur Authentifizierung lautet: Apop<userID>{MD5}(<process-ID.clock@hostname> <shared secret>)

Die Spezifikation des *pop3*-Standards wird in RFC 1939 [12] beschrieben.

pop3 ist ein ASCII-basiertes Übertragungsprotokoll für Clients. Es dient zum direkten Download der Nachrichten vom Mailserver zum Mailclient. Der Anwender kann dabei nicht entscheiden, welche der Nachrichten er empfangen möchte, sondern er muss den kompletten Download abwarten.

pop3 lässt sich über Telnet bedienen und verwendet üblicherweise den Standardport 110.

4.3.2 imap4

Überblick

Wie *pop3* ist *imap4* (Internet Message Access Protocol Version 4) ein Protokoll, mit dem ein Benutzer seine Mail von einem Mailserver abholen kann. Der Mailclient verbindet sich mit Hilfe dieses Protokolls mit dem *imap4*-Server, der in der Standardkonfiguration auf Port 143 läuft. Nach dem Verbindungsaufbau durch den Client durchläuft die Kommunikation die verschiedenen Phasen der Initiierung, Authentifizierung, Transaktion und Update.

Gegenüber *pop3* bietet *imap4* dem Client die Möglichkeit, zu entscheiden, welche der Mails aus seiner Mailbox auf den lokalen Rechner heruntergeladen werden sollen. Dazu werden zunächst die Header-Informationen der Mails an den Mailclient übertragen. Der Anwender erhält Informationen über die Absenderadresse, den Betreff der jeweiligen Mail, Größe, Priorität und ob sie bereits gelesen wurde oder noch nicht.

Wenn der Anwender eine Mail lesen möchte, wird eine Kopie davon auf den Client-Rechner übertragen. Während das *pop3*-Protokoll vorsieht, dass Mails nach dem Übertragen zum Mailclient auf Seiten des Mailservers gelöscht werden, hält ein *imap4*-Server die Nachrichten weiter vor, auch nachdem sie gelesen wurden.

Gerade wenn man mit einer schmalbandigen Verbindung wie zum Beispiel über eine analoge Modemverbindung seine Mail vom Server abrufen möchte, weiß man den Vorteil von *imap4* gegenüber *pop3* zu schätzen. Bei einer großen Zahl von Nachrichten für den Download blockiert das Laden der gesamten Nachrichten unter *pop3* jeden Zugriff auf bereits eingetroffene Mails. Man kann nicht entscheiden, welche der Nachrichten man sofort lesen und welche man lieber direkt und ungelesen entsorgen möchte. Gerade unter diesen Bedingungen spart *imap4* Zeit, Nerven und Verbindungskosten.

Das Vorhalten der Nachrichten auf dem Mailserver hat Vor-, aber auch Nachteile. Der Vorteil ist, dass man mit verschiedenen Clients auf die Mails zugreifen kann. So stehen die gleichen Nachrichten sowohl auf dem Arbeits-PC im Büro als auch auf dem Notebook zur Verfügung, wenn man sich beim Kunden über eine Dial-up-Verbindung ins interne Netzwerk des Unternehmens einwählt. Auf der anderen Seite benötigt man immer eine Netzwerkverbindung, um die Nachrichten zu lesen. Besteht diese nicht, können keine Mails bearbeitet werden, es sei denn, man lädt diese explizit auf den lokalen Rechner herunter.

imap4 bietet dem Benutzer eine gute Möglichkeit, seine eingegangenen Nachrichten zu verwalten. Man kann seine privaten Mails in einen eigenen Ordner befördern, es lassen sich Unterordner anlegen, es gibt einen Papierkorb, der gelöschte Mails zur Wiederherstellung bereitstellen kann, und die Nachrichten können als »neu«, »gelesen«, »beantwortet« oder »gelöscht« markiert werden. Moderne grafische Client-Programme unterstützen die Verwaltung der Mails über Point&Drop-Möglichkeiten.

Aber auch für den Administrator hat diese Unterscheidung Auswirkungen. Die Bequemlichkeit der Anwender verführt diese häufig dazu, die eingegangenen Nachrichten dort zu lassen, wo sie eingetroffen sind: in der Inbox des Servers. Dadurch wird aber im Laufe der Zeit das Speichersystem des Mailservers sehr belastet. Die Mailboxen der Anwender wachsen immer mehr an und schließlich ist die Speicherkapazität des Servers nahezu erschöpft, was die Gefährdung des Betriebs bedeutet. Vor diesem Hintergrund gelten üblicherweise für die Mailboxen der User Grenzwerte (Quotas), die nicht überschritten werden dürfen. Das zwingt den User, in regelmäßigen Abständen seine Mailbox »aufzuräumen« und alte bzw. unwichtige Mails zu löschen.

Das *imap4*-Protokoll wurde im Dezember 1996 mit RFC 2060 [13] erstmals vorgestellt und stellt mittlerweile eine verbreitete Form der Mailübertragung zwischen Server und Client dar.

Das Protokoll *imap4* wurde erstmals im Dezember 1996 im RFC 2060 vorgestellt. Es ist ein sehr flexibles und mächtiges Protokoll.

Neben den Download-Optionen, die dem Anwender die Entscheidung ermöglichen, welche Mails er lesen möchte und welche nicht, bietet es auch die Möglichkeit, die Mails permanent auf dem Mailserver vorzuhalten.

Ob man als Anwender die Mails auf dem Mailserver vorhält oder sie lieber auf den lokalen Rechner herunterlädt: *imap4* bietet mehr Flexibilität im Umgang mit Nachrichten als *pop3*.

imap4-Befehle

Die Verbindung zu einem *imap4*-Server bleibt ähnlich wie bei einer *pop3*-Session so lange erhalten, bis eine der beiden Seiten die Session abbaut. Sie durchläuft die Phasen der Initiierung, Authentifizierung, Transaktion und des Update. In den folgenden Abschnitten wird eine *imap4*-Session eines Mailclients mit dem Mailserver über telnet auf Port 143 nachgestellt.

1. Initiierungsphase

   Für die Simulation einer Mailverbindung eines *imap4*-Clients mit dem Server wird wie beim *pop3*-Protokoll eine Telnet-Session verwendet:

   ```
   > telnet imap.meinefirma.com 143

   OK imap.meinefirma.com imap4 service (Netscape Messaging Server
   4.15 Patch 4 (built Dec 7 2000))
   ```

   Damit ist zunächst die Netzwerkverbindung hergestellt und der Mailserver quittiert die erfolgreiche Verbindungsaufnahme mit einem »OK«, seinem Servernamen und weiteren Detailinformationen.

2. Authentifizierungsphase

   Nach dem initialen Verbindungsaufbau mit dem Server muss sich der Benutzer mit User-ID und Passwort authentifizieren. Für jeden Befehl muss eine fortlaufende Befehlskennung vorangestellt werden, die dann mit der Antwort des Servers quittiert wird (alle Befehle sind Case-Insensitive):

   ```
   a001 login iamtheboss nobodyknows
   a001 OK User logged in
   ```

   Der Benutzer ist nun für diese Session am Server angemeldet und kann Transaktionen durchführen. Wir werden zunächst einen ping-ähnlichen »No-Operation« (noop)-Befehl durchführen, den der Server quittiert:

   ```
   a002  noop
   a002 OK Completed
   ```

3. Transaktionsphase

   Mit *Capability* überprüfen wir die Möglichkeiten, die uns der Server bietet. In der Auflistung muss der Server mindestens *imap4rev1* bieten:

   ```
   a003 capability

   CAPABILITY imap4 imap4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS LANGUAGE
   XSENDER X-NETSCAPE XSERVERINFO AUTH=PLAIN AUTH=LOGIN

   a003 OK Completed
   ```

   In der Transaktionsphase können wir nun die Mailbox auflisten lassen:

   ```
   a004  list / *
   LIST (\NoInferiors) "/" INBOX
   LIST (\HasNoChildren) "/" All-Deutschland
   LIST (\HasNoChildren) "/" Boerseninfo
   LIST (\HasNoChildren) "/" Drafts
   LIST (\HasChildren) "/" Beruflich
   LIST (\HasChildren) "/" Beruflich/Mitarb
   LIST (\HasNoChildren) "/" Beruflich/Mitarb/Joe
   LIST (\HasNoChildren) "/" Beruflich/Mitarb/Jim
   LIST (\HasNoChildren) "/" Beruflich/Mitarb/Jack
   LIST (\HasNoChildren) "/" Beruflich/Mitarb/Maria
   LIST (\HasChildren) "/" Privat
   LIST (\HasNoChildren) "/" Privat/Susi
   LIST (\HasNoChildren) "/" Privat/Karl
   LIST (\HasNoChildren) "/" Privat/Omi
   LIST (\HasNoChildren) "/" Schedule
   LIST (\HasNoChildren) "/" Schneier
   LIST (\HasNoChildren) "/" Sent
   LIST (\HasNoChildren) "/" Trash
   LIST (\HasNoChildren) "/" Werbung
   a004 OK Completed
   ```

Es erfolgt eine Auflistung verschiedener Ordner, in denen die eigene Mail verwaltet werden kann. Der Benutzer kann jederzeit weitere Ordner anlegen oder einschließlich des Inhalts entfernen. Wir beschränken uns im Folgenden auf einen Ordner und selektieren die Inbox:

```
a005 select inbox

FLAGS (\Answered \Flagged \Draft \Deleted \Seen $Forwarded
$MDNSent)

OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted \Seen
$Forwarded $MDNSent \*)]

3 EXISTS
1 RECENT
OK [UIDVALIDITY 924303801]
a005 OK [READ-WRITE] Completed
```

Der Status der Inbox lässt sich gesondert abfragen:

```
a006 status inbox (messages)
3 EXISTS
1 RECENT
STATUS inbox (MESSAGES 3)
a006 OK Completed
```

Neben »messages« kann man den Status auch noch auf »recent«, »uidnext«, »uidvalidity« und »unseen« abfragen. Dem Leser bleibt es überlassen, selbst mit diesen Optionen seine Erfahrungen zu sammeln.

Im nächsten Schritt werfen wir einen Blick auf eine Mail, wobei zunächst einmal die Header-Informationen genügen:

```
a007 fetch 2 rfc822.header
2 FETCH (RFC822.HEADER {1240}
Return-Path: <joe.user@meinefirma.com>
Received: from meinefirma.com ([205.213.231.41]) by
mail.meinefirma.com
(Netscape Messaging Server 4.03) with esmtp id FXVRZB00.MUL;
Tue, 18 Jan 2002 06:42:47 +0000
…
…
X-Mailer: Mozilla 4.7 [en]C-CCK-MCD NSCP  (WinNT; I)
X-Accept-Language: en
MIME-Version: 1.0
To:all@meinefirma.com
Subject: BBQ am 5. August 2002
Content-Type: multipart/mixed;
boundary="------------967232A0F1388C47B38C04C5"
From: <joe.user@meinefirma.com>

)
a007 OK Completed
```

Ein *imap4*-Client-Programm durchläuft genau diese Schritte. Nach dem Verbindungsaufbau stellt es dem Anwender eine Liste der vorhandenen Mails zusammen, indem die Nachrichten nach ihren Header-Informationen abgefragt werden. Anhand dieser Liste entscheidet nun der User, welche der Mails er lesen möchte.

Zum Lesen einer Nachricht wird das »fetch«-Kommando übertragen:

```
a008 fetch 2 rfc822.text
222 FETCH (RFC822.TEXT {1916}
Hallo Folks…
Ich werde ein BBQ spendieren…
...
...
)
a008 OK Completed
```

Es gibt eine Vielzahl von weiteren Befehlen, die im *imap4*-Protokoll eine umfangreiche Verwaltung und Mailmanipulation zulassen. Die komplette Beschreibung jeder einzelnen Option würde den Rahmen dieses Buchs sprengen. Daher sei hier auf die Dokumentation in RFC 2060 [13] verwiesen.

4. Nun bleibt nur noch das Ausloggen aus dem Server, was in der Close/Logout-Phase erfolgt:

```
a009 logout
BYE LOGOUT received
a009 OK Completed
```

Mit *imap4* erhält der Anwender eine sehr flexible Mailverwaltung. Er kann entscheiden, ob er seine Mails auf dem Mailserver belässt und sie damit von verschiedenen Rechnern aus erreichbar macht oder ob er sie auf den eigenen Rechner herunterlädt. Auf diese Weise kann der User die für seine Zwecke günstigste Möglichkeit aussuchen.

Die Verwendung von *imap4* erfordert aber auch einen Mehraufwand an Administration. Viele Anwender sehen keine Notwendigkeit, die Nachrichten auf dem Server zu löschen. Der wachsende Umfang der Mailboxen führt schließlich zu einem hohen Speicherbedarf für den Server. Diesem Problem begegnet man mit Beschränkungen der Größe der Mailboxen. Durch entsprechende Quotas werden die Anwender gezwungen, alte Mails zu löschen oder zumindest auf den lokalen Rechner zu laden.

4.3.3 smtp

Das *smtp*-Protokoll ([6], [7]) wird verwendet, um vom Client verfasste Nachrichten zum ersten Mailserver für ausgehende Mails zu versenden, sie von einem Message Transfer Agent (MTA) zum nächsten zu übertragen und sie in einer Mailbox zu hinterlegen. Es existieren zwei Rollen, die ein *smtp*-Server einnehmen kann:

- *smtp* accept: Ein Server nimmt Mails entgegen und hinterlegt sie in der Queue zum sofortigen oder späteren Ausliefern.
- *smtp* deliver: Ein Server oder ein Client versendet eine Mail zu einem anderen Server.

Bei einem Mailtransfer startet der Sender eine Verbindung über *smtp* und wartet auf die Nachricht »220 Service Ready« oder »421 Service not available«.

Das *smtp*-Protokoll verwendet zum Austausch von Daten mit Clients per Standard den Port 25. Wie in den vorangegangenen Protokollen kann man auch hier einen Verbindungsaufbau über eine Telnet-Session nachbilden:

```
>telnet smtp.meinefirma.de 25

220 smtp.meinefirma.com esmtp service (Netscape Messaging Server 4.15
(build Aug 4 1999))
```

Der Sender übermittelt ein Hallo (HELO), worauf der Server seinerseits mit dem Full Qualified Hostname antwortet.

```
helo

250 smtp.meinefirma.com
```

Um zu überprüfen, ob der Server mit *esmtp* umgehen kann, ist es möglich, den ELHO-Befehl zu verwenden. Dieses Protokoll ist eine Erweiterung des *smtp*-Protokolls mit neuen Funktionalitäten. Für den Fall, dass der Server diese Extension des *smtp*-Protokolls nicht implementiert hat, antwortet er mit einer »500 Syntax Error«-Nachricht, andernfalls mit einer »250 OK«-Message, zusammen mit einer Liste der Erweiterungen, die er unterstützt.

```
elho
250-smtp.meinefirma.com
250-PIPELINING
250-HELP
250-EXPN
250-ETRN
250-DSN
250-SIZE
250-AUTH PLAIN LOGIN
250 AUTH=LOGIN
```

Die Spezifikation der *esmtp* Services Extentions-Implementierung wird in RFC 1869 [18] beschrieben. Hier nur eine kurze Zusammenfassung der verwendeten Spezifikationen:

- RFC 2487 – Secure smtp over SSL [14]
- RFC 2554 – Authenticated smtp [15]
- RFC 1870 – Size Keyword [16]
- RFC 1985 – Remote Message Queue Starting [17]

Zurück zu *smtp*: Der Sender beginnt nun mit der Übertragung der Mail. Hierfür verwendet er das Kommando »mail from: <absenderadresse>«, gefolgt von seiner Mailadresse. Im *smtp*-Protokoll ist an dieser Stelle keine Verifikation dieser Adresse erforderlich. Genau genommen wird in vielen Mailserver-Implementationen nicht einmal eine Überprüfung der Syntax auf Konformität mit der Mailadresse durchgeführt. Diesen Umstand nutzen so genannte Spammer zum Versenden von Werbemails, ohne ihren wahren Namen oder ihre Mailadresse dem Empfänger der Mail preiszugeben. Der Empfänger hat dann nicht mehr die Möglichkeit, sich beim Absender über die Mails zu beschweren.

Der Server bestätigt die Mailinitiierung mit einem »OK«:

```
mail from: joe.user@meinefirma.com
250 Sender <joe.user@meinefirma.com> OK
```

Im nächsten Schritt wird der Server über die Adresse des Empfängers der Mail informiert. Man kann auch mehrere Adressen angeben, die mit dem Ausdruck »rcpt to:<Zieladresse>« übermittelt werden:

```
rcpt to: susi.sorglos@meinefirma.com
250 Recipient <susi.sorglos@meinefirma.com> OK
```

Im Anschluss erfolgt mit dem Befehl »data« der Beginn der Übertragung des Body-Teils der Nachricht an den Server. Der Server antwortet mit einer Bestätigung und einer Information, in der er dem Sender mitteilt, wie dieser die Nachricht zu kennzeichnen hat, wenn sie komplett übertragen ist:

```
data
354 OK Send data ending with <CRLF>.<CRLF>
```

Der Sender übermittelt nun die Daten. Hier wird der Inhalt der Mail übertragen. Über die Control-Sequenz wird dem Server das Ende der Nachricht mitgeteilt, was er mit einem »250 OK« bestätigt:

```
Subject: Kino heute abend?
Date: Thu, 14 Mar 2002 13:22:19 +0100
From: Joe User <joe.user@meinefirma.com>
To: susi.sorglos@meinefirma.com

Hallo Susi. Wie waere es mit Kino heute abend?

Gruss, Joe User

.

250 Message received: FKSJIS00.000

quit

221 smtp.meinefirma.com esmtp server closing connection
```

Man kann nun die Verbindung mit einem simplen »quit« terminieren oder aber weitere Nachrichten versenden. Umgekehrt kann sich aber der Sender

nun in einen Empfänger für Mails umwandeln. Beide Parts wechseln ihre Rollen, indem der Sender den »Turn«-Befehl absetzt.

Der gesamte Datenverkehr erfolgt ohne Verschlüsselung und Authentifizierung.

4.3.4 http

Über *http* wird der Zugriff auf eine Webseite gesteuert, deren Applikation ihrerseits Mailclient für einen Standard-Mailserver ist. Genauere Informationen zu Webmail folgen in Abschnitt 4.5, das *http*-Protokoll wird in Kapitel 7.3 beschrieben.

4.4 X.400

Dieser kurze Abschnitt befasst sich mit der X.400-Architekur. Darüber hinaus steht jedoch der *smtp*-Standard im Mittelpunkt der Darstellungen. Beide Technologien sind sich sehr ähnlich, eventuelle Unterschiede werden in den folgenden Kapiteln, falls notwendig, herausgearbeitet.

4.4.1 X.400-Mailkomponenten

Ein Message Handling System (MHS) umfasst in der X.400-Welt folgende Komponenten:

- Benutzer-Clients (User Agents – UA), mit deren Hilfe der Anwender seine Nachrichten verfasst und empfängt,
- Message Transfer Agents (MTA) zur Übermittlung der Nachrichten vom Sender zum Empfänger,
- ein Gateway (Access Unit – AU), das die Mails aus der X.400-Welt in eine andere übersetzt,
- Message Store, (MS), der Dienst der eingehende Mails für einen User sammelt und sie zur Abholung bereitstellt. Die Nachrichten werden mit Statusinformationen versehen, die eine Mail als »gelesen« oder »ungelesen« markieren. Mit dem Wechsel dieses Status zu »gelesen« kann eine Quittierungsnachricht an den Absender der Mail übermittelt werden.

Das gesamte X.400-Mailsystem, das alle MTAs, User Agents und Gateways zu den anderen Mailsystemen beinhaltet, wird als Message Transfer System (MTS) bezeichnet.

4.4.2 Mailadress-Format

Eine Nachricht besteht in der X.400-Definition aus einem Nachrichtenkopf und einem Body, der wiederum mehrere Teilnachrichten unterschiedlichen Typs enthalten kann (Bodyparts).

In der Adressierung unterscheidet sich die X.400-Implementierung von der *smtp*-Architektur. Eine Adresse wird in einer Top-down-Hierarchie angegeben, die sich an die Implementierung des X.500-Distinguished-Name anlehnt und aus einer Aneinanderreihung von Attribute/Value-Paaren besteht.

Beispiel:

Joe User arbeitet in der Abteilung »Professional-Services« von »meinefirma«, die ihre Hauptniederlassung in Deutschland hat. Seine X.400-Adresse könnte folgendermaßen aussehen:

```
C=DE,O=MEINEFIRMA,OU=PROFESSIONAL-SERVICES,S=USER,G=JOE
```

Die wichtigsten Attribute sind:

- **C:** Country Name
- **A:** Administration Management Domain (ADMD)
- **P:** Private Management Domain (PRMD)
- **O:** Organization
- **OU:** Organizational Unit
- **S:** Surname
- **G:** Givenname

Bei der Verwendung der Zeichen spielt die Groß-/Kleinschreibung keine Rolle, die X.400-Spezifikation enthält aber eine Empfehlung bezüglich der zu verwendenden Zeichen.

4.4.3 Kommunikationsprotokolle

Die Kommunikation findet über folgende Protokolle (so genannte P-Protokolle) statt:

- **P1:** Protokoll zur Kommunikation der Message Transfer Agents innerhalb eines Message Transfer Systems
- **P3:** Protokoll zum Informationsaustausch zwischen User Agent bzw. Message Store und Message Transfer System
- **P5:** Protokoll, das den Zugang zu einer Telex-Schnittstelle in die X.400-Architektur ermöglicht
- **P7:** das Zugriffsprotokoll eines User-Agents auf den Message Store

4.4.4 Mailformat

Eine X.400-Mail besteht aus einem Umschlag (Envelope) und einem Inhalt (Content).

Envelope

Der Umschlag dient zum Übertragen der Nachricht von einem Sender zum anderen bzw. zum Routen der Mails zwischen den MTAs. Er enthält Informationen wie

- Absenderadresse (Originator)
- Empfängeradressen (Recipients)
- Dringlichkeit (Priority)
- Log-Einträge der MTAs auf dem Übertragungsweg (Trace-Information)
- Eindeutige Kennzeichnung der Mail (MPDU-ID)
- Angaben zur Art des Inhalts (Contenttyp und Encoded Information Type (EIT))

Content

Der Inhalt einer X.400-Nachricht umfasst:

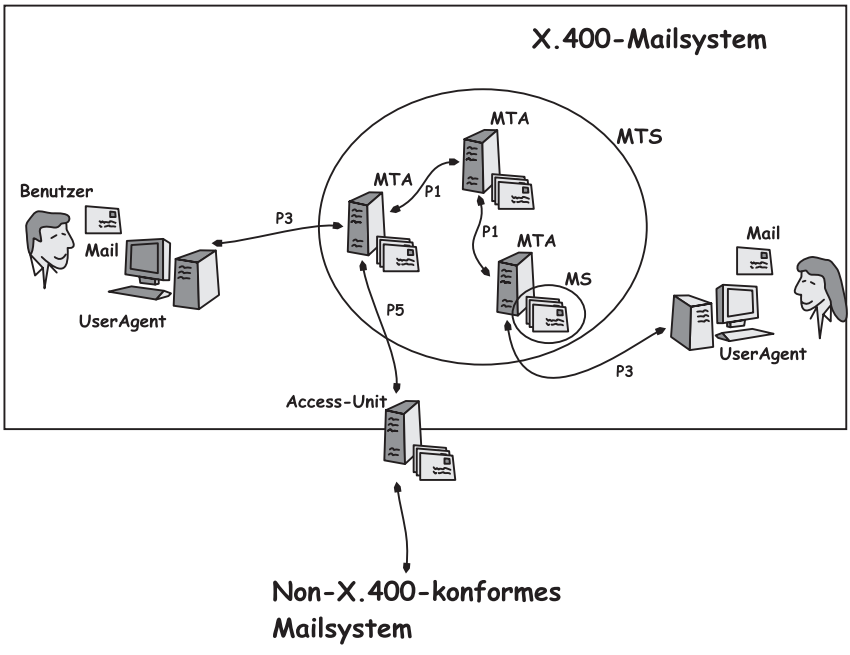
- Kopf (Header), bestehend aus
  - Absenderadresse (Originator)
  - Adressen von vertrauenswürdigen Stellvertretern (Authorizing Users)
  - Hauptadressaten (Primary Recipients)
  - Adressaten für Kopien (CopyRecipients)
  - Betreff (Subject)
  - Priorität (Importance)
  - Vertraulichkeit (Sensitivity)
  - Antwort an (ReplyToUsers)
  - Antwort erwartet bis (Replyby)
  - In Bezug auf (CrossReferences)
  - Gültigkeitsdatum (ExpiryDate)
  - Eindeutige Kennung (IPM-ID)
- Rumpf (Body)
  - Nachrichtenteile (Bodyparts)

Letztere können von unterschiedlichem Typ sein: andere Mails, ASCII-Text, Binär-Files usw.

Eine solche Vielfalt von Möglichkeiten bietet das vereinfachte *smtp*-System nicht. Beispielsweise ist es nicht einfach, den Vorteil einer Priority, wie man

sie aus dem X.400-Umfeld kennt, in einer *smtp*-basierten Architektur mit Standardmitteln nachzubilden. Auch die Verwendung von Nachrichtenteilen unterschiedlichen Typs wurde erst mit dem MIME-Standard eingeführt, bei X.400 waren sie von Anfang an Bestandteil der Spezifikation.

Abbildung 4.5: X.400-Mail-architektur



Die Ausführungen der folgenden Abschnitte basieren auf dem gebräuchlicheren *smtp*-Standard. Für Details zur X.400-Implementierung sollte man sich in die Publikationen der ITU [11] einlesen.

4.5 Webmail

Der Mailzugriff über das *http*-Protokoll wird spätestens seit der Einführung der Portale im Internet ständig weiterentwickelt. Man hat die Werbewirksamkeit von Webseiten erkannt, die häufig von Benutzern frequentiert werden. Um eine solche Seite für den einzelnen Internetanwender attraktiv zu machen, wurden Dienste wie zum Beispiel Webmail entwickelt. Dabei wird einem Interessenten, der auf dieser Seite vorbeikommt, ein kostenloser Mailaccount angeboten. Um seine Mail abzurufen, besucht der Benutzer immer wieder diese Webseite und kann dadurch gezielt mit Produkten beworben werden.

Für den einzelnen Interessenten hat diese Art des Zugriffs auf die Mail einen besonderen Vorteil: Er kann von jedem Internetcafé der Welt aus auf seine Mail zugreifen, die einzig erforderliche Zugangssoftware ist ein *html*-Brow-

ser. Wir werden später auf die Syntax des *http*-Zugriffs zurückkommen und beschränken uns hier auf die Architektur dieses Dienstes.

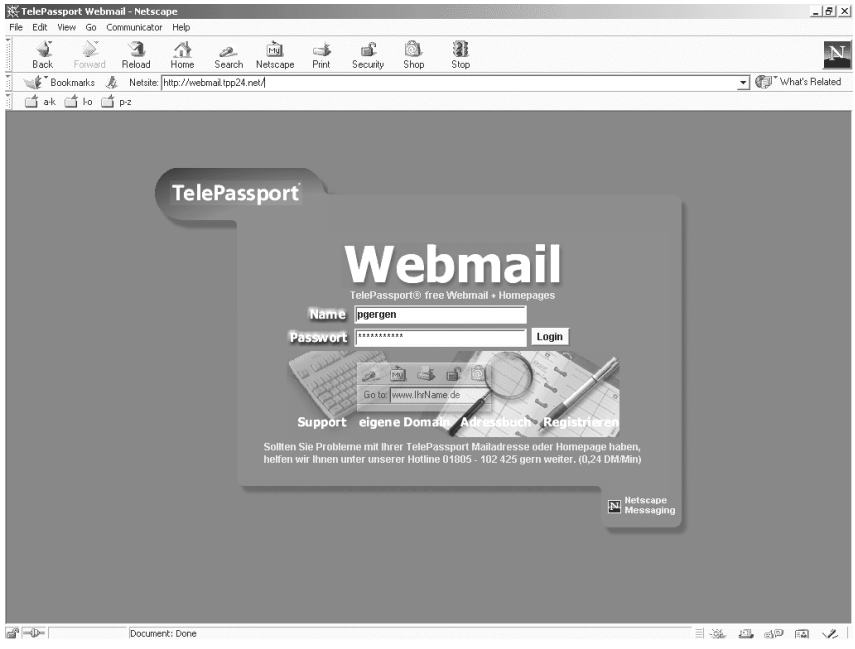
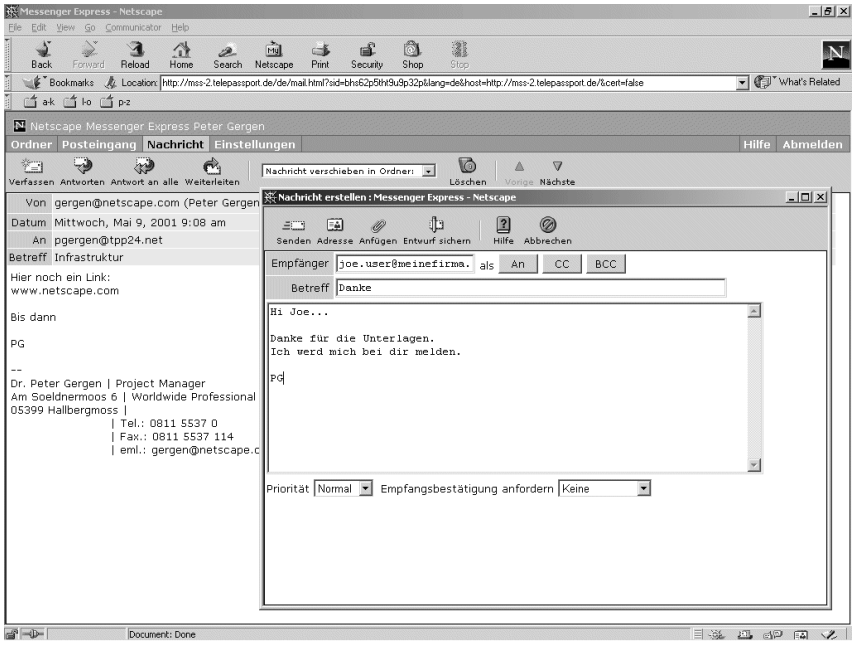


Abbildung 4.6: Login-Seite eines Webmail-Anbieters

Voraussetzung für einen Webmail-Dienst ist ein Mailserver, der über *smtp* die für den Benutzer vorgesehene Mail entgegennimmt und über *imap4* bereitstellt. Ein vorangestellter Webserver bietet eine *html*-Seite an, auf der sich der Benutzer anmelden kann. Anschließend generiert eine Applikation auf dem Webserver einen *imap4*-Request, der die Betreffzeilen der eingegangenen und bereits gelesenen Mails vom Mailserver anfordert, in *html*-Seiten umwandelt und sie dem Benutzer darstellt. Dieser kann nun über weitere Links auf die einzelnen Mails zugreifen, die jedes Mal vom Webserver in *html*-Seiten übersetzt und vom Browser auf Benutzerseite dargestellt werden.

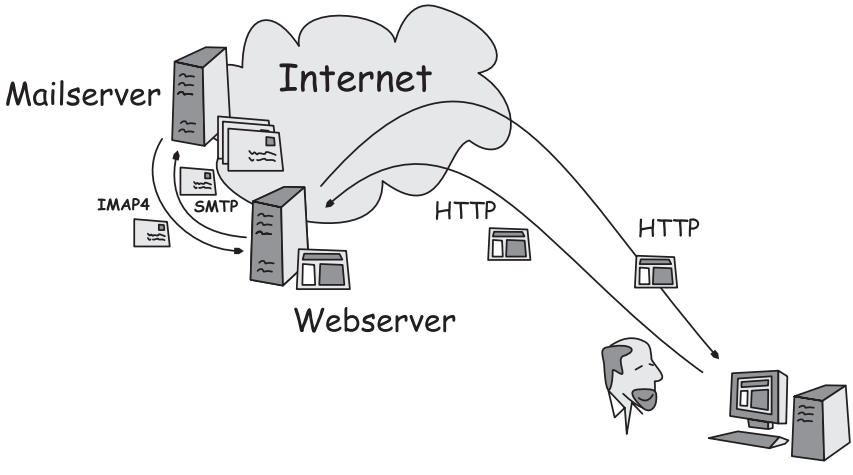
Genauso werden Mails geschrieben: Eine Webseite mit einem *html*-Formular stellt ein Textfeld bereit, in dem der Benutzer seine Mail formulieren kann. Mit dem Absenden durch Klicken auf den entsprechenden Button übernimmt die Applikation auf dem Webserver die Daten des oder der Textfelder und sendet sie über *smtp* an den Mailserver, der die Nachricht weiterleitet. Damit ist der Webserver im klassischen Sinn gegenüber dem Mailserver ein *smtp*- und *imap4*-Client, der die Aufgabe hat, die Mails von der einen in die andere Form zu übersetzen und zu übertragen.

Abbildung 4.7: Persönlicher html-basierter Mail-desktop



Mit der beschriebenen Architektur ist auch klar, warum hier nur *imap4* als Zugriffsprotokoll in Frage kommt: Bei *pop3* wären die Mails bei einem Download auf dem Webserver hinterlegt. Mit *imap4* bleiben die Nachrichten auf dem Mailserver und können jederzeit und überall gelesen werden.

Abbildung 4.8: Webmail-Topologie



4.6 Mailarchitekturen

Für den Aufbau eines Mailsystems existieren eine Reihe von verschiedenen Ansätzen. Die aufzubauende Topologie orientiert sich eng an den vorhandenen Gegebenheiten, insbesondere an der Menge der zu übermittelnden Mails, daneben aber auch an der vorhandenen bzw. noch einzurichtenden Netzwerkkomponenten.

In diesem Zusammenhang werden nun verschiedene Szenarios diskutiert, in denen die Notwendigkeit einer frühen Planung des Mailsystems verdeutlicht werden soll. Wie bei den vorangegangenen Systemen bedarf es auch hier einer Businessanalyse zusammen mit einem Implementierungsplan, einer Pilotphase bis hin zum Roll-out des Gesamtsystems, um bereits in einer frühen Phase die Weichen für eine skalierbare Lösung zu stellen.

4.6.1 Analyse der gegebenen Netztopologie

Zunächst gilt es, die bereits vorhandene Unternehmenslandschaft zu analysieren. Die Beispielfirma »meinefirma« sei in München, London und Paris vertreten, in Deutschland existieren außerdem zwei weitere Geschäftsstellen, jeweils in Frankfurt und Hamburg. Hauptsitz ist München mit 120 Mitarbeitern, in Frankfurt sind 10, in Hamburg 12 Mitarbeiter beschäftigt. In London arbeiten 60, in Paris 70 Mitarbeiter. Für dieses Unternehmen soll nun eine sinnvolle und skalierbare Verteilung der Mail organisiert werden.

In einem Mailsystem unterscheidet man zwischen zwei Topologien: dem verteilten und dem zentralisierten Ansatz.

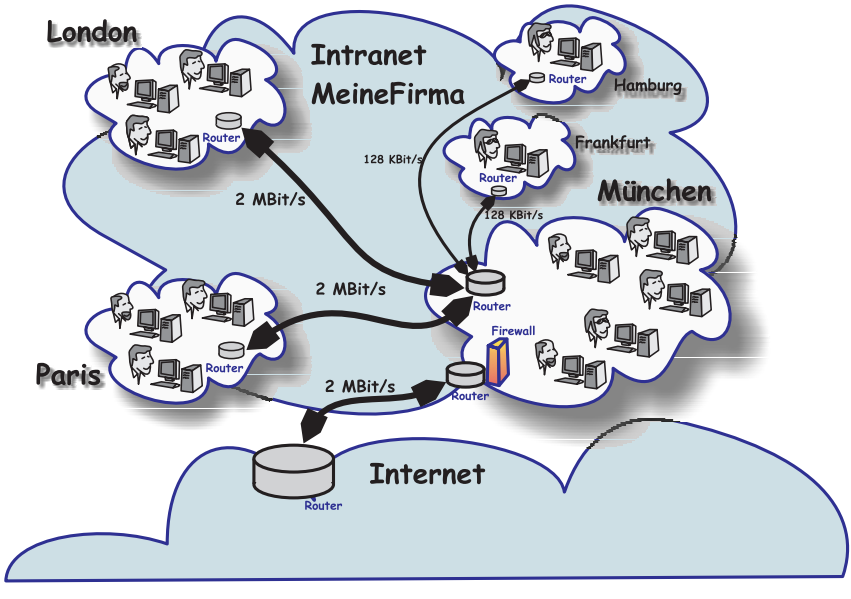


Abbildung 4.9: Beispiel einer Netzwerktopologie

Der verteilte Ansatz sieht vor, dass in jeder Geschäftsstelle ein Mailserver aufgebaut ist, der den vollen Umfang der Dienste übernimmt, also Mails versendet und empfängt. In einem zentralisierten Ansatz dagegen erfolgt der gesamte Mailverkehr von einem einzigen Standort aus, ohne lokale Server. Beide Ansätze sind als Extreme einer realistischen Implementierung anzusehen, die irgendwo zwischen verteilter und zentralisierter Topologie anzusiedeln ist.

Welche Topologie die optimale Lösung für ein Unternehmen darstellt und in welchem Maß sie eher zentralisiert oder verteilt zu implementieren ist, hängt von einer Reihe von Randbedingungen ab:

- Technische Gesichtspunkte

  Diese Aspekte umfassen zunächst die Berücksichtigung der gegebenen Netzwerkstruktur. Insbesondere die Netzwerkanbindung der einzelnen Unternehmenszweige untereinander beeinflusst bereits im Vorfeld sehr stark die weitere Ausrichtung der Mailtopologie. Gegebenenfalls muss eine bestehende Verbindung mit dem Einrichten eines Mailsystems erweitert werden.

- Sicherheits- und Wartungsfragen

  Welchen Einfluss hat ein neues System auf die unternehmensweite Datensicherheit? Wer soll den Dienst administrieren? Sind genügend Ressourcen vorhanden, um einen permanenten Betrieb des Mailsystems zu gewährleisten? Welche Failover-Szenarios sind vorgesehen, falls einmal ein Dienst ausfällt?

- Unternehmerische Gesichtspunkte

  Diese betreffen insbesondere Fragen der Wirtschaftlichkeit und der Akzeptanz durch den Benutzer. Welche Benutzer verwenden das Mailsystem? Mit welcher Intensität? Welche Datenmengen sind zu erwarten? Wie kritisch wäre ein Ausfall des Systems für die unternehmensweite Kommunikation? Andere Fragestellungen beziehen sich auf die politische Ebene: Möchte eine lokale Geschäftsstelle den Maildienst selbst verwalten und damit einen Teil politischer Unabhängigkeit bewahren? Ist die Zahl der Benutzer vor Ort zu gering, als dass sich ein lokaler Dienst lohnen würde? Und welcher administrative Aufwand ist mit der einen oder anderen Topologie verbunden?

- Finanzielle Gesichtspunkte

  Welche Kosten kommen auf das Unternehmen zu, wenn ein Mailsystem eingesetzt wird? Welche Kosten entstehen, wenn das System von einer Architektur auf die andere migriert werden soll? Wächst die gewählte Topologie mit der Mailarchitektur mit, ohne neue Kosten zu verursachen? Oder reicht eine schnelle billige Lösung für den ersten Ansatz?

Es lassen sich folgende Aspekte der verschiedenen Topologien festhalten:

1. Benutzerverhalten

   Wesentlichen Einfluss auf die Topologie hat die Beschaffenheit der Mails. Werden viele lange, umfangreiche Mails mit großen Attachments versendet, eignet sich eher eine verteilte Topologie als die zentralisierte. Der Anwender wählt sich in seine Mailbox ein und findet im lokalen Netzwerk bereits seine Mails vor. Es wird keine überregionale Verbindung aufgebaut, die die Mail erst mit dem Anmelden des Benutzers überträgt. Die Nachricht befindet sich vielmehr bereits im lokalen Netzwerk und kann mit Ethernet-Geschwindigkeit auf den Client geladen werden. Dies erhöht die Akzeptanz des Benutzers, seine Arbeitszeit wird nicht mit langen Downloads vertan. Natürlich verringert diese Topologie nicht die Auslastung des Netzwerks, aber die Verteilung geschieht bereits zu Zeiten, in denen der Benutzer noch nicht mit dem Lesen der Mails beschäftigt ist.

   Umgekehrt ist bei kurzen Mails der Aufwand geringer, um die Daten von einer überregionalen Stelle auf den Client zu laden, und es entstehen beim Laden der Mails keine großen Wartezeiten für den Benutzer. In diesem Zusammenhang ist eine zentralisierte Topologie durchaus akzeptabel, wobei eine verteilte Architektur in Bezug auf diesen Aspekt keinen Nachteil mit sich bringt.

2. Benutzerverteilung

   Ein weiterer wesentlicher Faktor, der die Architektur des Mailsystems beeinflusst, ist die lokale Verteilung der Benutzer. Sind in einer Geschäftsstelle viele Mitarbeiter beschäftigt, die alle auf den Maildienst angewiesen sind, ist sicherlich das Einrichten eines lokalen Mailservers notwendig. Umgekehrt wird sich ein einzelner Mitarbeiter in seinem Homeoffice keinen Mailserver einrichten.

3. Netzwerkbandbreite

   Gute Netzwerkanbindungen der Geschäftsstellen untereinander und insbesondere zur Hauptstelle unterstützen eine zentralisierte Mailtopologie, bei wenigen bzw. teuren Anbindungen hat eine verteilte Architektur ihre Vorteile.

Beispiel:

Abbildung 4.9 zeigt die Netzwerktopologie der bereits angesprochenen Beispielfirma. Die Standorte London und Paris sind jeweils mit einer 2 Mbit/s-Verbindung an die Zentrale in München, die Niederlassungen Hamburg und Frankfurt jeweils über eine 128 Kbit/s-Leitung angeschlossen. Der gesamte Internetverkehr erfolgt über eine 2 Mbit/s-Leitung in der Zentrale, das Netzwerk wird gegen unerlaubte Zugriffe aus dem Internet über eine Firewall gesichert.

Wie schon gesagt, sind in München 120 Mitarbeiter beschäftigt, von denen 80% über einen eigenen Mailaccount verfügen. In London und Paris mit 60 bzw. 70 Mitarbeitern sowie in Hamburg (12 Mitarbeiter) und Frankfurt (10 Mitarbeiter) besitzen alle Beschäftigten einen Mailaccount.

Das Mailverhalten der Benutzer ist recht unterschiedlich. Es gibt Mitarbeiter, die am Tag etwa zehn Mails versenden, andere verschicken eine Mail in zwei Tagen. Durchschnittlich aber werden pro Person etwa drei Nachrichten pro Tag versendet.

Die Größe einer Mail unterlag von Beginn an keinen Einschränkungen. Es zeigte sich bald eine Tendenz zu immer längeren Nachrichten, hauptsächlich verursacht durch Attachments. Im Schnitt beträgt die Größe einer Mail etwa 30 Kbyte.

Das Mailsystem entwickelte sich zu einem sehr wichtigen Kommunikationsmittel, das auf keinen Fall für längere Zeit ausfallen darf. Für die Geschäftsführung ist es wichtig, dass die Architektur skalierbar ist, also langfristig mit dem Unternehmen mitwächst.

Welches ist nun eine geeignete Mailarchitektur? Aus Sicht des Unternehmens existieren zwei Datenflüsse: einer zwischen dem Unternehmen und dem Internet, ein weiterer ausschließlich innerhalb des Intranets. Der zentrale Punkt innerhalb dieser Topologie ist München, dieser Standort hat auch die meisten Mitarbeiter. Hier wird der zentrale Mailserver stehen, der die Mail weiter im Unternehmen verteilt.

Die Standorte Paris und London sind zwar mit einer guten Bandbreite an die Zentrale angeschlossen, man muss aber berücksichtigen, dass neben dem Mailverkehr auch Webdienste und Downloads aus dem Internet über diese Leitungen erfolgen. Darüber hinaus stellt eine Verteilung des Maildienstes auf verschiedene Mailboxen an den einzelnen Standorten eine Entlastung des zentralen Servers dar. Hier erscheinen also weitere Mailserver an beiden Standorten auf jeden Fall sinnvoll.

In den Niederlassungen Frankfurt und Hamburg sind relativ wenig Mitarbeiter beschäftigt. Obwohl das übrige Internetaufkommen über die gleichen Leitungen geht, werden im Tagesdurchschnitt nur drei Mails pro Person mit etwa 30 Kbyte Größe versendet und empfangen, was bei Hamburg und Frankfurt etwa 2000 Kbyte an geschätztem Mailaufkommen pro Tag ausmacht. Dementsprechend beträgt das mittlere Netto-Mailaufkommen in beiden Fällen etwa 70 Byte/s pro Arbeitstag (der acht Stunden umfasst). Das Einrichten je eines Mailservers an jedem Standort, verbunden mit dem administrativen Aufwand und der Neuanschaffung von zusätzlicher Hardware, erscheint hier nicht unbedingt notwendig. Dementsprechend kann man sich darauf beschränken, dass die Mitarbeiter in beiden Niederlassungen die Mails vom Server in der Hauptstelle laden. Wachsen die Anforderungen in diesen Niederlassungen, kann man jederzeit einen lokalen Server installieren.

Im Allgemeinen ist eine dezentrale Struktur von Vorteil. Man sollte aber berücksichtigen, dass lokale Mailserver nicht nur Hard- und Software benötigen, sondern auch Administratoren, die diese Dienste pflegen. Wenn sich einzelne Niederlassungen so weit entwickelt haben, dass der administrative Aufwand für eine zentrale Verwaltung der Mails größer wird als der für einen lokalen Mailserver, sollte man die jeweilige Niederlassung aus dem zentralen Gefüge entkoppeln.

Ein weiteres Argument für eine lokale Instanz ergibt sich aus der teilweisen Entkopplung vom WAN. Geht einmal die Verbindung zum zentralen Server aufgrund einer Störung des Netzwerks verloren, können lokal immer noch Mails ausgetauscht werden. Selbst Mails, die nach außen adressiert sind, werden vom Mailserver entgegengenommen und bei späterer Verfügbarkeit des Netzwerks ausgeliefert, so dass der Benutzer wenigstens einen teilweise funktionierenden Dienst vorfindet.

Erfahrungsgemäß kommt vor dem Dezentralisieren des Maildienstes der Ausbau der Netzwerkanbindungen. Man kann bei kleinen Standorten die zentralisierten Ansätze weiterhin betreiben und Engpässen bei der Mailzustellung durch den sukzessiven Ausbau der Bandbreite des Netzwerks begegnen. Dies unterstützt insbesondere die Kommunikation der Clients über andere Protokolle, beispielsweise den *http*-Access für die webbasierte Internetanbindung der Benutzer.

4.6.2 Mail-Delivery-Prozess

Wie wird denn nun eigentlich eine Mail an den Bestimmungsort ausgeliefert? Woher »weiß« das System, was zu tun ist, wenn eine Nachricht von einem Absender zu einem Adressaten gesendet werden soll?

Im Folgenden beschreiben wir die einzelnen Schritte anhand eines Beispiels:

1. Eine Mail wird von einem Anwender über ein Mailclient-Programm geschrieben. Beispielsweise möchte Joe User (»joe.user@meinefirma.com«) eine Mail an seine Kollegin Susi Sorglos schreiben. Die Mail wird verfasst und mit dem Adressaten »susi.sorglos@anderefirma.com« abgeschickt. Die Nachricht soll demnach aus dem Intranet »meinefirma.com« über das Internet in ein anderes Intranet mit dem Namen »anderefirma.com« verschickt werden.

2. Im Mailclient-Programm ist der Mailserver eingetragen, der die Nachrichten zum Versenden entgegennimmt, hier ist es »*smtp* OUT«. Sobald der Benutzer das Abschicken der Nachricht initiiert, wird dieser Mailserver über *smtp* kontaktiert und die Nachricht übermittelt. Dieses Versenden an den Server kann mit oder ohne Authentifizierung geschehen, die meisten Nachrichten werden ohne Überprüfung des Absenders versendet.

3. Der Mailserver »*smtp* OUT« wertet das Flag »rcpt to:« aus und führt einen Syntaxcheck durch. Eine Mailadresse hat dabei der Schablone [name]@[Domain] zu genügen.

4. Beim nächsten Schritt wird überprüft, ob die Mailadresse lokal ist oder ob die Mail in eine externe Domain weitergeleitet werden soll. Der Server »kennt« seine Domänen, die er verwaltet. Es kann also durchaus sein, dass der Server mehrere Domänen mit Mail bedient, bei denen er »weiß«, wohin er die Nachricht senden soll. Diese Informationen werden in »Routing Tables« beschrieben. Dabei handelt es sich um Tabellen, in denen beschrieben wird, wie die Mails für diese Domänen weitergeleitet werden.

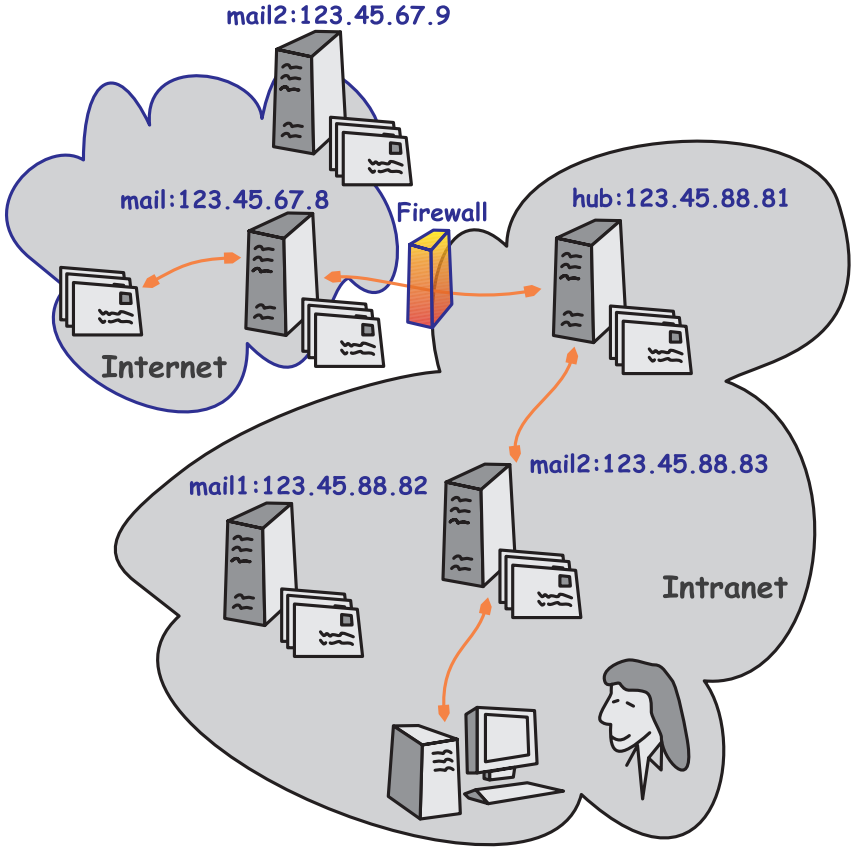


Abbildung 4.10: Mailtopologie des Beispielsystems »anderefirma.com«

Handelt es sich aber nicht um eine lokale Domain, wird eine Anfrage auf den »MX«-Record-Eintrag der Ziel-Domain gestellt. Das Akronym steht für »Mail eXchange« und beschreibt den verantwortlichen Server für eine Domain.

Beispiel:

Joe User sendet seine Mail an Susi Sorglos in der Domain »anderefirma.com«. Der Server *smtp* OUT in der Domain »meinefirma.com« bekommt die Nachricht, die zur Auslieferung an »susi.sorglos@anderefirma.com« adressiert ist. Der Domain-Teil ist nach dem »@«-Zeichen

definiert, also »anderefirma.com«. Für *smtp* OUT ist damit noch nicht klar, wohin die Mail übermittelt werden soll. Er fragt also beim DNS-System nach, an welchen Rechner er eine Mail senden soll, wenn diese an die Domain »anderefirma.com« adressiert ist. Dies wird über die MX-Record-Abfrage realisiert. In der Domain »anderefirma.com« findet sich im DNS-System der Eintrag, dass der Rechner »mail.anderefirma.com« mit der IP-Adresse »123.45.67.8« für eingehende Mails verantwortlich ist. Sollte dieser Dienst nicht reagieren, steht ein Backup-Server mit der IP-Adresse 123.45.67.9 und dem Hostnamen mail2.anderefirma.com zur Verfügung (vgl. Abb. 4.10).

Das lokale DNS-System in der Domain »meinefirma.com« muss zunächst selbst im Internet nachforschen und den MX-Eintrag der Zieldomain herausfinden. Ist diese Information nicht in einem darüber liegenden DNS gecached, muss das DNS-System der anderen Domain direkt gefragt werden. Im Folgenden sehen Sie ein Beispiel für die externe DNS-Konfiguration der Domain »anderefirma.com«. (In der externen DNS-Konfiguration werden nur die IP-Adressen bekanntgegeben, die notwendig sind. Die internen IP-Adressen werden nur im internen Netzwerk verwaltet. Das Beispiel beruht auf Abb. 4.10.)

Externe DNS-Konfiguration:

```
anderefirma.com.
      IN MX     10 mail.anderefirma.com
anderefirma.com.
      IN MX     30 mail2.anderefirma.com
*.anderefirma.com.
      IN MX     10 mail.anderefirma.com
*.anderefirma.com.
      IN MX     30 mail2.anderefirma.com
mail.anderefirma.com    IN A    123.45.67.8
mail2.anderefirma.com   IN A    123.45.67.9
hub.anderefirma.com     IN A     123.45.88.81
```

Interne DNS-Konfiguration:

```
anderefirma.com.        IN MX   10 hub.anderefirma.com
mail.anderefirma.com    IN A    123.45.67.8
hub.anderefirma.com     IN A    123.45.88.81
mail1.anderefirma.com   IN A    123.45.88.82
mail2.anderefirma.com   IN A    123.45.88.83
```

Damit ein *smtp*-basiertes Mailnetzwerk funktioniert, muss das DNS-System die folgenden Informationen liefern:

MX – Mail-eXchange-Eintrag für eine Domain. Diesem Eintrag ist eine Priorität zugeordnet, die umso höher ist, je kleiner die Zahl ist.

A – Address-Record-Eintrag, der die Zuordnung eines Full Qualified Hostname zu einer IP-Adresse übernimmt.

Beim Domain-Teil einer Mailadresse ist nicht von vornherein klar, ob es sich um eine (Sub-)Domain oder um den Fully Qualified Host Name eines Rechners handelt. Deshalb betrachten die meisten Mailserver diesen Teil vorläufig als Domain, d.h. sie suchen nach einem MX record. Ist diese Suche erfolglos, wird die Information als a record gewertet, mit dem versucht wird, die Zieladresse über das DNS ausfindig zu machen. Schlägt auch diese Suche fehl, wird die Adresse als falsch abgelehnt.

5. Ist der Server, der auf der anderen Seite die Mails entgegennimmt, identifiziert, wird über *smtp* eine Verbindung zu diesem Rechner aufgebaut. Für die Aktion gibt es zwei Varianten.

   Variante A: Bevor er die eingehende Mail akzeptiert, überprüft der Server in der Domain »anderefirma.com«, ob es sich beim Adressaten um eine gültige Mailadresse handelt. Ist dies der Fall, akzeptiert er die Mail, andernfalls nicht. Hierfür verwendet man den Begriff »Post-Accept-Modus«.

   Variante B: Der Server in der Domain »anderefirma.com« nimmt erst die Mail an und überprüft anschließend die Gültigkeit der Mailadresse. Hierbei handelt es sich um die »Pre-Accept«-Variante.

   Tritt an einer Stelle der Zulieferung der Nachricht ein Fehler auf, der vom Mailsystem nicht bearbeitet werden kann, wird die Nachricht »gebounced«, ein anglisierter Begriff für eine Aktion, die durchgeführt wird, wenn es bei einem Server zu Problemen mit einer Mail kommt. »Bouncen« kann das Löschen der Mail sein, aber auch das Zurückschicken zum Postmaster des Servers, der entscheiden soll, was damit passiert, oder die Mail wird ohne weitere Aktionen an den Absender zurückgeschickt. Was immer in einem solchen Fall geschehen soll, wird unter diesem Begriff zusammengefasst.

6. Der Server in der Domain »anderefirma.com« sucht die Mailbox des Benutzers heraus, um diese dort abzuliefern. Dies kann im einfachsten Fall der Server selbst sein, andernfalls wird nachgeschaut, welcher Rechner hierfür verantwortlich ist. Dieser Eintrag kann in einer lokalen »Datenbank« am Mailserver hinterlegt sein oder die Information stammt aus einem Directory Server.

7. Die Mailbox hält die Mail für den Adressaten vor. Das heißt, wenn sich Susi Sorglos beim nächsten Mal am Mailserver die eingegangenen Nachrichten abholt, ist die von Joe User dabei.

Diese Vorgehensweise ist recht allgemein beschrieben und weicht abhängig vom jeweiligen Hersteller an der einen oder anderen Stelle etwas ab. Die meisten Mailserver arbeiten jedoch nach diesem Prinzip.

4.7 Sicherheit im Mailsystem

Heutzutage kann es sich kaum ein Unternehmen mehr leisten, gegenüber Kunden oder Interessenten im Internet keine Schnittstelle für den Informationsaustausch bereitzustellen. So wie man eine Webseite erwartet, über die man sich über die Produkte und Dienstleistungen informieren kann, ist eine Mailanbindung selbstverständlich, über die man einen unternehmensinternen Ansprechpartner kontaktieren kann. Umgekehrt gibt es kaum noch Schreibtischarbeitsplätze, die nicht über einen Internetanschluss verfügen. Selbst wenn der Zugriff auf Webseiten nur begrenzt möglich ist, so gilt doch der uneingeschränkte Versand von Mails sowohl intern als auch ins Internet als selbstverständlich.

Mit den beiden Anwendungen »Web« und »Mail« ist normalerweise der Informationsbedarf eines Angestellten an seinem Arbeitsplatz abgedeckt. Aus Sicht der unternehmensinternen Sicherheitspolitik besteht zwischen beiden Technologien ein großer Unterschied. Webzugriffe sind Anforderungsaktionen, die vom einzelnen Mitarbeiter aus dem internen Netzwerk heraus initiiert werden. Dieser entscheidet, welche Seiten er sich ansieht, während das Unternehmen über die Reglementierung dieses Zugriffs bestimmt. Der Informationsfluss ist weitgehend unidirektional.

Das Mailsystem dagegen wird von diesseits wie jenseits der Firewall gleichermaßen betrieben. Mails gehen ins Internet hinaus und werden von außen in das Unternehmen hinein gesendet. Gerade Letzteres stellt die Sicherheitsmaßnahmen vor entscheidende Probleme: Welche externen Mailaktionen fügen dem Unternehmen Schaden zu und welche können unbedenklich durchgelassen werden?

Daten lassen sich am einfachsten über das Mailsystem von außen in das interne Unternehmensnetz einschleusen. Anwender erhalten beispielsweise Mails aus dem Internet mit schädlichen Attachments, die das Sicherheitssystem aushebeln können. Dies muss durch einen geeigneten Maßnahmenkatalog weitgehend verhindert werden. Die folgenden Abschnitte befassen sich mit den verschiedenen Arten von Angriffen auf ein Mailsystem – und den entsprechenden Strategien zu deren Abwehr.

4.7.1 Mailsystem: Angriffe und Missbrauch

Es gibt mehrere Möglichkeiten, das Mailsystem für Attacken auf Personen oder Unternehmen zu missbrauchen:

- **Mailbomben**

  So genannte Mailbomben haben das Ziel, das Mailsystem des Angriffsopfers lahmzulegen. Dies wird dadurch erreicht, dass eine oder mehrere Nachrichten mit großem Volumen an die Zielperson versendet werden bzw. dass eine hohe Zahl von kleinen Mails verschickt werden, um die Mailbox des Adressaten überlaufen zu lassen. Dies gelingt vor allem,

wenn Letzerer nur über eine schmalbandige *pop3*-Verbindung Zugriff auf seine Mailbox hat. Bei dieser Konstellation hat der User keine Chance zu entscheiden, ob und welche Mail er vom Server auf sein System herunterladen möchte, er muss den kompletten Download aller neu eingegangenen Nachrichten abwarten. Da Angreifer zur Erzeugung einer umfangreichen Mail gerne die frei kopierbare Betriebssystemerweiterung »X11« mit einem Umfang von etwa 80 Mbyte verwenden, werden derartige Mailbomben mitunter auch als »X11-Bomben« bezeichnet.

Das Versenden einer großen Mail an einen anderen Account belastet aber auch den Maildienst des Angreifers, so dass diese Form des Angriffs weniger häufig zum Einsatz kommt.

Angriffe mittels zahlreicher kleinerer Mails basieren häufig auf einem verteilten Angriff, in dessen Rahmen möglichst viele Wirtsrechner Nachrichten versenden. Der »I Love You«-Virus beispielsweise hatte eine solch verheerende Wirkung, weil in einem infizierten Windows-System sämtliche Adressbucheinträge gelesen und an jede dieser Mailadressen eine Nachricht verschickt wurde. Diese hatten die Betreffzeile »I Love You« und waren mit einer Kopie des Virus versehen.

- **Anonyme Drohungs-/Belästigungsmail**

  Der Absender nutzt einen offenen Mailrelay, um unerkannt belästigende Mails zu versenden. Dies ist unter *smtp* relativ einfach möglich, da beim Versand der Mail in Standardkonfigurationen keine weiteren Überprüfungen insbesondere auf die Richtigkeit der Absenderangaben erfolgen. Dadurch ist es schwer herauszufinden, wer der Urheber der Nachrichten ist.

  Verfolgt man den Verfasser dieser Mails, so muss man zusammen mit den Administratoren der Server in den Logfiles die Einträge identifizieren, in denen die Mailverbindungen des Urhebers protokolliert sind.

- **Import von Viren**

  Durch die Möglichkeit, Personen in einem internen Netzwerk direkt adressieren und Daten übermitteln zu können, bietet sich das Mailsystem geradezu an, um über Attachments in Mails Viren und Würmer in ein Unternehmen einzuschleusen. Abhilfe schaffen Virenscanner, die den eingehenden Datenstrom analysieren und gegebenenfalls vor Viren warnen. Solche Scanner können aber nur die Signaturen bekannter Schädlinge entdecken und sind weitgehend machtlos, wenn ein neuer Virus auftaucht, der dem Scanner noch nicht bekannt ist. Zudem wird die Suche nach Viren erschwert, wenn beim Nachrichtenaustausch Verschlüsselungstechniken zum Einsatz kommen.

- **Spam-Nachrichten**

  Nachrichten, die unaufgefordert versandt werden und deren Inhalt entweder Werbezwecken dient und/oder geringen bis keinen Informationswert für den Empfänger bedeutet, werden als Spam (oder auch »Junkmail«) bezeichnet. Spam tritt im Mail- und im News-Service in zunehmenden Maße auf.

Im Unterschied zum Werbebrief ist die elektronische Form um ein Vielfaches effizienter, da sie keine Portogebühren und Materialkosten verursacht. Dies ist der Grund, warum Werbemails in der letzten Zeit so dramatisch zugenommen haben.

Spam kann in Netzwerken zu Beeinträchtigungen der Netzwerkbandbreite führen. Verschickt ein Spammer eine Werbemail, die eine Größe von etwa 2 Kbyte hat, an 200.000 Mailadressen, entsteht ein Datenaufkommen von etwa 400 Mbyte. Mit jeder neuen Spam-Mail werden weitere Netzwerkressourcen beansprucht und Bandbreiten verbraucht.

Das Versenden von Spam-Mails ist in höchstem Maße unsozial, da die meisten Adressaten keinen Wert auf diese Nachrichten legen und dennoch Zeit und Geld investieren müssen, um diese Mails herunterzuladen und auszusortieren. Im Kampf gegen diese Art von Belästigung gehen seit einigen Jahren die ersten Staaten aktiv gegen Spam vor. So hat in Österreich im Juli 1999 das Parlament mit Zustimmung aller fünf Parlamentsparteien eine Novelle zum Telekommunikationsgesetz beschlossen, die das Spamming als Delikt einordnet.

- **Reale Mailinglisten**

  Der Angreifer meldet sein Opfer an möglichst vielen Mailinglisten an. Von jeder Liste bekommt der Anwender nun mehr oder weniger regelmäßig Nachrichten, was in der Summe eine riesige Mailflut entstehen lässt. Um von diesen Listen wieder entfernt zu werden, muss man jede einzelne mit der Bitte um Abmeldung kontaktieren.

- **Falsche Mailingliste als Köder**

  Adressenjäger sind an aktiven Mailaccounts interessiert. Sie sammeln in Newsgroups alle verfügbaren Mailadressen auf und versenden an jede eine Willkommensmail mit der Information, dass der Adressat von nun an in einer Mailingliste eingetragen ist, ohne dass dem wirklich so ist. Um von der vermeintlichen Mailingliste wieder entfernt zu werden, muss man eine »unsubscribe«-Mail zurücksenden, die den Adressenjäger wiederum darüber informiert, dass es sich bei der Adresse um einen aktiven Account handelt. Damit ist die Mailadresse des Anwenders qualifiziert und kann für Werbemails verwendet werden.

  Aus diesem Grund verwenden die meisten Anwender immer weniger reale und insbesondere primäre Mailadressen, wenn sie in öffentlichen Newsgroups Artikel posten. Vielmehr greift man auf einen der unzähligen Webmail-Accounts zurück, die man sich nach Belieben einrichten kann, oder man gibt beim Posten eines Diskussionsbeitrags von vorneherein eine falsche Mailadresse an.

- **Unberechtigtes Verwenden des Mailservers (Relaying)**

  Wird aus einem fremden Netzwerk der Mailserver des eigenen Unternehmens verwendet, um Mails in andere fremde Netzwerke zu versenden, spricht man von »Relaying«. Aus zwei Gründen will man dies verhindern:

- Ressourcenschonung: Dürfen externe Anwender den eigenen Server für den Mailversand in andere Domains verwenden, so schränkt man die verfügbaren Ressourcen für die eigenen Anwender ein.
- »Relaying« kann Spammern helfen, ihre eigene Identität zu verschleiern. Der Server wird so zum Ausgangspunkt vieler Spam-Mails und bei der Suche nach den Verursachern der Massenmails wird man auf den unschuldigen Betreiber des Mailservers zurückgreifen. Im schlimmsten Fall kommt ein »missbrauchter« Mailserver auf die schwarze Liste der unerwünschten Mailserver im Internet, von denen regelmäßig Spam-Mail zu erwarten ist. Andere Mailserver laden sich in regelmäßigen Abständen solche schwarzen Listen herunter und ignorieren Mails von Servern, die auf dieser Liste stehen. Dadurch kann das Mailsystem des Unternehmens nicht mehr im vollen Umfang am internationalen E-Mail-Austausch teilnehmen.

- **E-Mail-Betrug/Fälschung (Spoofing/Forging)**

  Verschleiert man (aus welchem Grund auch immer) die Absenderadresse einer Mail, liegt eine E-Mail-Fälschung vor. Der Adressat bekommt eine Nachricht scheinbar von einer anderen Person, der eigentliche Absender bleibt verborgen.

Allen Angriffen gemein ist, dass sie zunächst auf die externe Schnittstelle des Mailsystems zielen. Während das interne Mailsystem durch die Firewall weitgehend vor externen Angriffen geschützt ist, befinden sich die Mailserver, die als primäre Anlaufstelle für die eingehende Mail dienen, entweder vor der Firewall oder in einer Demilitarisierten Zone. (DMZ: ein abgesichertes Netzwerk, das alle Dienste umfasst, die in direktem Kontakt mit dem Internet stehen.) Diese Server müssen sowohl von der Hardware als auch von der Software (Betriebssystem und Mailservice) abgesichert sein (es werden nur die notwendigsten Dienste betrieben, alle anderen werden abgeschaltet). Gegenüber dem internen Netzwerk darf über die Firewall hinweg nur von den externen Servern her eine *smtp*-Verbindung aufgebaut werden.

Weitere Maßnahmen zur Absicherung des Mailsystems betreffen die einzelnen Server selbst: Virenscanner untersuchen die eingehenden Mails auf schädlichen Inhalt, über Filterregeln lassen sich Mails aus unerwünschten Netzwerken abweisen, TCP-Wrapper verhindern bereits auf Hardware-Ebene den Verbindungsaufbau aus unerwünschten Netzwerken.

4.7.2 Aufspüren von Spammern

Man kann Werbemail ignorieren oder aber versuchen, den oder die Urheber ausfindig zu machen, um Gegenmaßnahmen einzuleiten.

Wie kann man herausfinden, welchen Weg eine Nachricht durch das Internet hin zum Adressaten genommen hat? Jede einzelne Mail enthält eine Menge entsprechender Informationen, man muss nur einmal die Header-Informationen analysieren.

Beispiel:

Die Mailbox enthält eine Werbemail, die im Folgenden genauer untersucht werden soll. (Beim vorliegenden Beispiel handelt es sich um eine reale Werbemail, deren Netzwerkdaten nachträglich verändert wurden.)

```
Date:   Wed, 12 Dec 2001 06:01:01 EST

From:   yesitsfree.2@reply.ad-vert.com

Message-Id:   <200112121336.FAA20135@s1011.colo2.ad-vert.com>

MIME-Version:   1.0

Received:   from  rly-na03.mx.meinprovider.com
(rly-na03.mail.meinprovider.com  [172.18.151.232]) by
air-na02.mail.meinprovider.com  (v82.22) with ESMTP id MAILINNA24-
1212073655; Wed, 12 Dec 2001 07:36:55 1900

Received:   from  s1011.colo2.ad-vert.com (s1011.ad-vert.com
[207.33.16.17]) by rly-na03.mx.meinprovider.com  (v82.22) with ESMTP
id MAILRELAYINNA32-1212073638; Wed, 12 Dec 2001 07:36:38 1900

Received:   (from pmguser@localhost)by s1011.colo2.ad-vert.com
(8.8.8pmg/8.8.5) id FAA20135for :include:/usr/home/pmguser/pmgs/users/
yesitsfree/delivery/1007748625.23824/rblk.511; Wed, 12 Dec 2001
05:36:55 -0800 (PST)

Return-Path:   <yesitsfree@ofr.ad-vert.com>

Subject:   FREE BURGERS + 1/2 Price Omaha Steaks!

To:   joe.user@meinefirma.com

This message is being sent to you as a member of YesItsFree.com. If
you feel you have received this message in error or do not wish to
receive future messages, please see the unsubscribe instructions at
the bottom of this email.

Just look at what you'll save this holiday season:

…

 The Gourmet Grillers

   4  (5 oz.) Filet Mignons  (628BYD)

   8  (5 oz.) Gourmet Burgers

Reg. $84.00, Now Only $42.00, SAVE 50%!
```

Sie könnten nun eine Mail an »yesitsfree.2@reply.ad-vert.com« schreiben, um nachzufragen, warum diese Mail an Sie geschickt wurde. Für den Fall, dass eine Fehlermail von Ihrem Mailserver zurückkommt, wüssten Sie, dass es sich um eine gefälschte Mailadresse handelt. Eine Reaktion auf eine Spam-Mail kann aber dem Sender auch übermitteln, dass es sich bei Ihrem Account

um eine aktive Mailadresse handelt. Unübersehbar ist die Information, dass »diese Nachricht an Sie als Mitglied der YesItsFree.com-Domain versendet wurde (war ich noch nie!) [..] und falls Sie das Gefühl haben, dass diese Mail irrtümlich an Sie versandt wurde, [...]«. Man wird in diesem Fall aufgefordert, dem Absender eine Nachricht zu schicken, dass man nicht Mitglied dieser Domain ist. Spätestens dann ist aber Ihre Mailadresse als gültige und aktive Adresse qualifiziert und Sie sind erst recht Ziel zukünftiger Werbemails.

Hier kann man mit dem »finger«-Befehl sein Glück versuchen:

```
>finger yesitsfree.2@reply.ad-vert.com
```

Im Allgemeinen ist dieser Dienst bei den Internet Providern aus Sicherheitsgründen abgeschaltet und dieser Versuch bringt Sie auch hier nicht viel weiter.

Untersuchen Sie stattdessen die Header-Informationen. Bereits in der dritten Zeile findet sich ein interessanter Hinweis auf den Absender der Mail: die Message-ID.

```
Message-Id:   <200112121336.FAA20135@s1011.colo2.ad-vert.com>
```

Die Message-ID ist eine eindeutige Kennung einer Mail. Sie wird von dem Server generiert, bei dem die Nachricht zum ersten Mal erscheint, also beim Versenden der Mail beim *smtp*-Server des Absenders. Da diese Kennung Rückschlüsse auf den Verfasser zulässt, ist ein Spammer daran interessiert, sie zu fälschen oder zu entfernen.

Jede Mailserver-Implementierung hat ihren eigenen Stil, eine Message-ID zu definieren. Allgemein hat sie das Format:

```
Message-Id:   <unique string>@<mailserver>
```

Im Beispiel hat der Spammer seine Mail offensichtlich aus der Domain »ad-vert.com« heraus versandt.

Betrachten Sie den Weg, den die Mail zurückgelegt hat. Die Beispielmail hat drei verschiedene »Received:«-Felder. Jeder MTA, der die Mail weitertransportiert, fügt ein neues »Received:«-Attribut hinzu, das jüngste ist zuoberst. Der erste Server, der die Nachricht empfangen hat, befindet sich am weitesten unten in der Liste der »Received:«-Einträge in den Header-Zeilen der Mail.

Der erste Eintrag stammt von dem Server, der die Mail zugestellt hat, also Ihr Internet Service Provider. Es ist relativ unwahrscheinlich, dass dieser Eintrag nicht glaubwürdig ist.

```
Received:   from  rly-na03.mx.meinprovider.com
(rly-na03.mail.meinprovider.com  [172.18.151.232])
by air-na02.mail.meinprovider.com  (v82.22) with ESMTP id
MAILINNA24-1212073655; Wed, 12 Dec 2001 07:36:55 1900
```

Die Domain »meinefirma.com« wird vom Provider »meinprovider.com« gehostet, also ist der Mail-Exchange-Point dieses Providers für die eingehende Mail verantwortlich. Der angeführte »Received:«-Eintrag beschreibt, wie die Nachricht vom eingehenden Mailserver »rly-na03.mx.meinprovider.com« zum Server »air-na02.mail.meinprovider.com« weitergeleitet wird, auf dem unsere Mailbox hinterlegt ist. Die Übertragung erfolgte über *esmtp*.

Der vorangegangene »Received:«-Eintrag ist interessanter.

```
Received:   from  s1011.colo2.ad-vert.com (s1011.ad-vert.com
[207.33.16.17]) by rly-na03.mx.meinprovider.com  (v82.22) with ESMTP
id MAILRELAYINNA32-1212073638; Wed, 12 Dec 2001 07:36:38 1900
```

Hier ist der Weg beschrieben, wie die Mail von einem Server aus der Domain »ad-vert.com« an den Mail-Exchange-Punkt Ihres Providers geleitet wurde. In der Message-ID ist diese Domain bereits aufgetaucht und so sollten Sie sich diese etwas näher anschauen.

```
>whois ad-vert.com

Registrant:

     Advertizing Design, Inc.
        15 S. First Street Suite 123
        San Jose, CA 95113
        US

        Domain Name: AD-VERT.COM

        Administrative Contact, Billing Contact:

         Advertizing Design Hostmaster
             hostmaster@AD-DESIGN.COM
         Advertizing Design, Inc.
         1234 Forrest Hill Rd.
         San Jose, CA 95124
         US
         408-1234567 Fax 408-1234568

        Technical Contact:

         ISP Network Systems Hostmaster
                    hostmaster@ISPNET.NET

         ISP Network Systems
         1234 East 123 North
         Orem, UT 84097
         US
         801-1234567
         Fax 801-1234568
```

```
Record last updated on 08-Oct-1999.
Record expires on 08-Oct-2001.
Record created on 08-Oct-1999.

Db last updated on 4-Dec-2000 00:30:51 EST.

Domain servers in listed order:

NS1.ISPNET.NET192.41.3.10

NS2.ISPNET.NET161.58.2.10
```

Auf den ersten Blick ist diese Domain von einem Internet Provider mit dem Namen »ispnet.net« gehostet. Interessanterweise ist »ispnet.net« eine Sackgasse, ein Blick auf den Webserver »www.ispnet.net« macht deutlich, dass diese Domain zum Verkauf steht.

Die verlässlichste Information ist der Eintrag des Mailservers Ihres Providers, der die IP-Adresse protokolliert hat, von der er die Mail bekommen hat: »201.33.16.11«. Der *tracert*-Befehl verrät einiges mehr:

```
tracert 207.33.16.17

Routenverfolgung zu s1011.ad-vert.com [207.33.16.17]  über maximal 30
Abschnitte:

  1   771 ms   912 ms   781 ms  srv47.meinprovider.net [62.33.207.93]
  2   701 ms   781 ms   912 ms  rmws-meinprovider.net [195.72.233.1]
  3   560 ms   141 ms   631 ms  62.35.96.8
  4   811 ms   922 ms  1041 ms  ge-1.linx.some-isp.net
[195.69.224.139]
  5   911 ms   912 ms  1041 ms  p4-1-0-0.r02.some-isp.net
[129.251.4.181]
  6  1042 ms   911 ms  1052 ms  p4-4-0-0.r04.some-isp.net
[129.251.5.11]
  7  1031 ms   911 ms   912 ms  p16-4-0-0.r01. some-isp.net
[129.251.5.12]
  8   951 ms  1042 ms   911 ms  p16-3-0-0.r00. some-isp.net
[129.251.4.21]
  9  1082 ms  1121 ms  1112 ms  p16-6-0-0.r00. some-isp.net
[129.251.5.112]
 10  1192 ms   921 ms  1051 ms  p16-4-0-0.r06. some-isp.net
[129.251.5.68]
 11  1112 ms  1122 ms  1111 ms  ge-1-0-0.a02. some-isp.net
[129.251.15.16]
 12  1112 ms  1122 ms  1111 ms  p4-1-2-0.a00. some-isp.net
[129.251.122.150]
 13  1112 ms  1121 ms  1142 ms  ge-1-1.a03. some-isp.net
[129.251.122.219]
 14     *        *        *     Zeitüberschreitung der Anforderung.
 15  1402 ms  2013 ms  2063 ms  s1011.ad-vert.com [207.33.16.17]
```

Die Route zu unserem Spammer führt also über einen anderen großen Internet Service Provider, was seine Zuordnung zu diesem Netzwerk sehr viel wahrscheinlicher macht.

Auf diese Weise kann man Schritt für Schritt die Route der Werbe-Mail durch das Internet nachvollziehen und eventuell gefälschte Informationen identifizieren.

Man kann nun auf Spam-Mail verschieden reagieren. Am besten schickt man von einem unkritischen Mailaccount aus eine Beschwerde an den Internet Service Provider mit »cc:« an den Absender der Spam-Mail – in der Hoffnung, dass der Service-Provider seinen Spam-Kunden zur Rechenschaft zieht und künftigen Spam-Aktionen vorbeugt.

4.7.3 Aufbau einer sicheren Mailtopologie

Das Mailsystem ist normalerweise das einzige System, dem der Aufbau einer Verbindung von außen und das Senden von Informationen in das Intranet ohne interne Anforderung gestattet ist. Und eben deshalb ist es oft Ziel externer Angriffe, da hier die Schwachstellen des Sicherheitssystems vermutet werden.

Um einen sicheren Mailtransfer zu gestatten, ohne dass sich Lücken im Sicherheitsgeflecht des internen Unternehmensnetzwerks auftun, geht man im Allgemeinen von einer zweistufigen Hierarchie der Mailtopologie aus.

Die erste Stufe der Absicherung ist der Aufbau eines Puffernetzes: Das Netzwerk, das dem Internet am nächsten steht, ist wie bereits angesprochen die Demilitarisierte Zone (DMZ). Diese bildet einen Übergang zwischen dem Gefahrenpotenzial aus dem Internet und dem zu schützenden internen Netz. In dieser DMZ werden für die Kommunikation nach außen Dienste eingerichtet, deren Übernahme durch Hacker die Datensicherheit des Unternehmens noch nicht unmittelbar bedrohen würde. Das DMZ besitzt jeweils eine Firewall zum Internet und zum internen Netzwerk. Es steht unter besonderer Beobachtung in Hinblick auf Angriffe von außen.

Innerhalb dieses Netzwerks wird nun der Mailserver für die eingehende Internetmail installiert. Für die äußere Firewall richtet man eine Regel ein, die einen *smtp*-Connect von überall her auf diesen Server auf Port 25 zulässt. Für die innere Firewall wird festgelegt, dass *smtp*-Verbindungen nur von diesem einen Mailserver in der DMZ zu einem internen Server gestattet sind. Die Regeln hierzu werden durch Einträge in der Routing-Tabelle in den Konfigurationsdaten der Mailserver hinterlegt. Ein solcher Server für den Mailtransfer in der DMZ wird als Internet Mail Relay bezeichnet.

Im internen Netzwerk befindet sich ein Mailserver, der vom Internet Mail Relay die Nachrichten entgegennimmt und sie weiter verteilt. Ein solcher Mailhub (»smtp in« oder »Inbound Mailhub«) sortiert wie bei einer Poststelle die Nachrichten und routet sie weiter ins interne Netzwerk. Er ist so

konfiguriert, dass er nach außen keine Mails versenden kann. Dies ist die Aufgabe eines anderen Dienstes, dem Outbound Mailservice (»smtp out«).

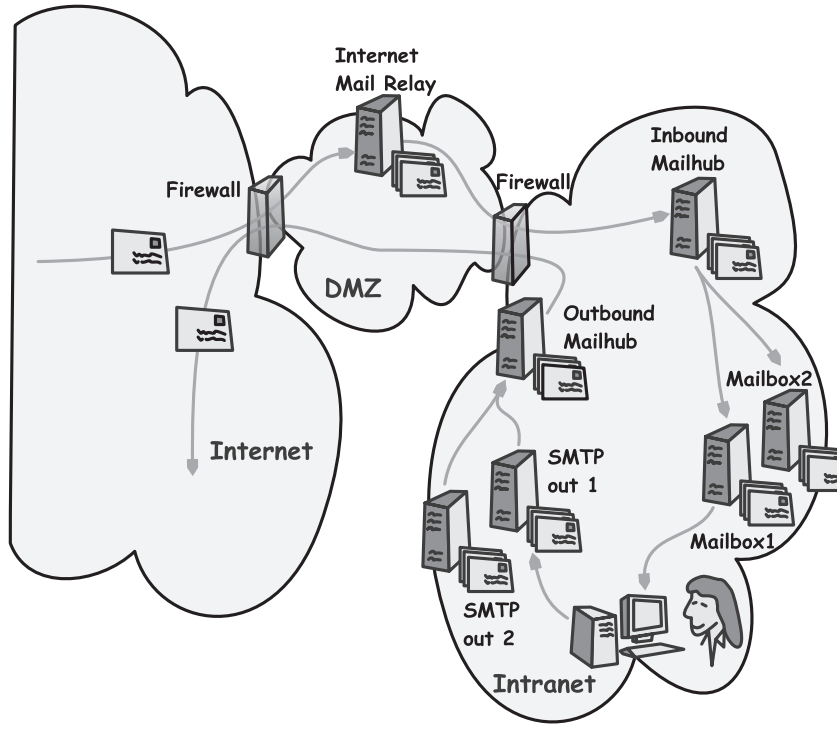


Abbildung 4.11: Verschiedene Rollen innerhalb einer Mailarchitektur

Für die ausgehende Mail kann man einen Outbound Mailhub vorschalten, der als einziger Dienst durch die beiden Firewalls hindurch Mails über *smtp* versenden kann. Dies unterstützt das Sicherheitskonzept: Es muss nicht für jeden neuen Mailserver, der ins Internet Nachrichten versendet, eine neue Regel an den Firewalls eingerichtet werden. Alle internen *smtp*-Dienste versenden ihre Nachrichten an diesen einen internen Dienst, der sie dann direkt ins Internet weiterleitet.

Hier wurde ein Szenario entworfen, in dem verschiedene Rollen von MTAs dargestellt wurden. Die tatsächliche Implementierung kann vorsehen, dass alle oder zumindest einige dieser Dienste auf einem Rechner über einen einzigen Service umgesetzt werden. Das Rollenprinzip sollte aber bei der Planung der Umsetzung eines sicheren Mailkonzepts herangezogen werden, um die Aufgabenverteilungen der Dienste übersichtlich zu halten und Sicherheitslücken zu isolieren und zu beheben.

Antirelay-Regel

Der Mailserver benötigt als weitere Sicherheitsmaßnahme eine Regel für Antirelay. Darunter versteht man die Konfiguration des Dienstes in der

Form, dass er nicht von außen kontaktiert werden kann, um über ihn Mails nach außen zu verschicken.

Folgende Maßnahmen und Konfigurationen verhindern, dass ein Mailserver zu einem Relay wird:

- In der internen Netzwerktopologie müssen alle Domänen und Subnetze identifiziert werden, die von einem Mailserver als Berechtigte für den Nachrichtenversand akzeptiert werden. Ein großes Unternehmen kann mehrere Subdomains hosten, die alle denselben Mailserver verwenden, um ins Internet Nachrichten zu versenden. Diese müssen dem Server bekannt gemacht werden. Ein Client, der nicht zum internen Netzwerk gehört, darf keine Mails ins Internet versenden.
- Die Realisierung des Antirelay basiert auf Mailfiltern, über die der Server sowohl DNS- als auch IP-Address-basiert entscheidet, ob es sich bei dem sendenden Client um einen internen oder einen externen User-Agent handelt.
- Das Antirelay-System, das definiert und umgesetzt wurde, muss schließlich getestet und ausfallsicher gemacht werden.

Antirelay-Regeln müssen nicht unbedingt für ausschließlich externe Clients gelten, schließlich kann es im internen Netzwerk ebenfalls Spammer geben, die man über diese Regeln abfangen möchte.

Üblicherweise handelt es sich bei den Antirelay-Filtern um Einstellungen auf *smtp*-Level. Das heißt, der Client hat bereits eine Verbindung über dieses Protokoll an den Server und versucht, seine Mail zu versenden:

```
elho spammer.com

250 smtp.meinefirma.com

Hello spammer.com [123.45.6.7], pleased to meet you
mail from: spammer@spammer.com
250 spammer@spammer.com... Sender ok
rcpt to: tom@ganz-andere-firma.de
550 tom@ganz-andere-firma.de.. Relaying denied
```

Anti-Spam-Filter

Eine weitere Regel soll dem Mailserver eine Aussortierung von unerwünschter Mail ermöglichen. Diese »Unsolicited Bulk E-Mail«-Filter (UBE) werden vom Mailadministrator konfiguriert und verhindern, dass Nachrichten aus einer bestimmten Domain, von einer bestimmten Mailadresse oder mit einem bestimmten Betreff das interne Netzwerk erreichen. Im einfachsten Fall verhindert man damit Spam-Mails, schwerer wiegt aber die Möglichkeit der Abwehr von Viren, die mit einem bestimmten Betreff (z.B. »I love you«) verschickt werden. Diese Filter unterbinden nicht die Übermittlung von Viren, da dies die Aufgabe eines Virenscanners ist. Dieser durchforscht den Inhalt einer Mail und alarmiert beim Auffinden der ent-

sprechenden Signatur eines Virus den Administrator. Aber wie im Fall des »I Love You«-Virus lassen sich Folgemails mit dem entsprechenden Betreff mit Hilfe des UBE-Filters unterdrücken.

TCP-Wrapper

Man kann den Verbindungsaufbau aus unerwünschten Netzwerken auf niedrigeren Protokollebenen verhindern, indem man einen TCP-Wrapper vorschaltet. Dieser kontrolliert auf IP-Level die Initiierung einer Socket-Verbindung.

Authenticated smtp

Der vorherige Abschnitt hat gezeigt, wie man über eine falsche Absenderadresse eine Mail absenden kann. *smtp* beinhaltet im normalen Einsatz keine Schutzmaßnahmen gegen diesen Missbrauch des Systems. Es ist als offenes Protokoll vorgesehen, was einen einfachen und effizienten Transfer von Nachrichten erlaubt. Aber gerade diese Lücke wird genutzt, um Mails versenden zu können, ohne dass der Urheber seine Identität preisgeben muss. Dies zu unterbinden, ist die Aufgabe des Authenticated *smtp*-Protokolls, eine Erweiterung der einfachen *smtp*-Version. Eine Beschreibung dieser Technik ist in RFC 2554 [15] beschrieben. Dort finden sich eine Reihe von weiteren Ergänzungen zum ursprünglichen Protokoll.

Über das Authenticated *smtp*-Protokoll authentifiziert sich der Sender einer Nachricht gegenüber dem Mailserver, dem er die Mail zum Weiterleiten übergibt. Dies kann gegenüber einer lokalen Datenbank am Server oder über ein zentrales Verzeichnis geschehen. Bevor die Mail akzeptiert wird, verlangt der Server eine User-ID und ein Passwort, das er mit seinen Benutzerdaten vergleicht. Nur bei einer erfolgreichen Authentifizierung wird die Mail weitergeleitet. Folgende Schritte sind im Handlungsablauf vorgesehen:

1. Der Benutzer schreibt eine Mail, die er mit Hilfe des Client-Programms erstellt.
2. Wenn der Benutzer die Mail versendet, fragt das Client-Programm eine User-ID und ein Passwort des Benutzers ab, das im Mailsystem eine gültige Kennung ist.
3. Der Client authentifiziert sich gegenüber dem Server mit den User-Daten. Dies geschieht im Protokoll mit Hilfe des AUTH-Befehls, der unmittelbar im Anschluss an den Verbindungsaufbau über Port 25 und dem EHLO (der neueren HELO-Version des *esmtp*-Protokolls) folgt. Normalerweise hat der Client die User-Daten im Cache zwischengespeichert, um nicht jedes Mal beim Versenden einer Mail vom Benutzer die gleichen Daten abzuverlangen.
4. Der Mailserver akzeptiert die Mail und versendet sie. Dies kann er entweder über *smtp* ohne weitere Authentifizierung tun oder aber die Benutzerprüfung wird von einem Sender zum nächsten weitergetragen. Im ersten Fall ist sichergestellt, dass der Server nicht von Spammern

missbraucht wird, die weitere Übermittlung der Mail erfolgt ungesichert. Wir verfolgen hier aber den zweiten Weg, in dem die Authentifizierung des Benutzers innerhalb einer Kette von Servern weitergetragen wird. Hierzu hat der erste Server mit dem folgenden eine Authentifizierung vereinbart, die auf dem Austausch einer Server-ID und einem vereinbarten Passwort beruht. Da ein Outgoing Mailserver allerdings nicht mit allen Mailservern im Internet eine solche Vereinbarung treffen kann, wird diese Authentifizierung oft nur innerhalb eines Unternehmens oder in einem überschaubaren Netzwerk eingesetzt.

5. Das *esmtp*-Protokoll kann damit die Information, dass die Mail authentifiziert wurde, bis hin zum Empfänger der Nachricht übermitteln. Dessen Client wertet die Nachricht aus und stellt sie dar.

esmtp ist in dieser Anwendung der erste Schritt in Richtung eines abgesicherten Mailsystems. Wir werden uns in Folge mit weiteren Sicherungen befassen, die in einem solch offenen Kommunikationsmedium Integrität und Privatsphäre ermöglichen.

Kryptografische Absicherung der Mail

Mails, die in einem Netzwerk versendet werden, werden im Klartext übertragen. Dies gilt sowohl für Nachrichten, die in einem kleinen Netzwerk versendet werden, als auch für den länderübergreifenden Austausch. Mit einfachen Netzwerkmitteln, wie zum Beispiel »snoop«, werden die Daten, die von einem Rechner zum anderen gesendet werden, mitprotokolliert, oft ohne dass dies der Adressat oder der Absender bemerkt. Gerade im internationalen Mailverkehr werden die ausgetauschten Nachrichten an bestimmten Knoten durch Filter klassifiziert: Sind bestimmte Schlüsselwörter in der Mail enthalten, werden diese auf die nächst höhere Beurteilungsebene gesetzt. Eine Mail, in der beispielsweise die Wörter »President« und »attack« gleichzeitig vorkommen, ist verdächtig und wird zunächst von automatischen Filtern, in letzter Konsequenz aber von Mitarbeitern von Nachrichtendiensten gelesen und auf ihr Gefahrenpotenzial hin untersucht. Das von den Amerikanern betriebene Echelon steht trotz Dementis der Betreiber im Verdacht, den internationalen Datenaustausch zu belauschen, sei es, um Terroranschlägen vorzubeugen oder um Wirtschaftsspionage zu betreiben.

Um das Belauschen von Mails zu verhindern, verwendet man die Technik des Verschlüsselns. Kapitel 6 wird sich näher mit den entsprechenden Architekturen befassen. Für den Augenblick genügt die Feststellung, dass man mit Hilfe der Verschlüsselung Daten unlesbar machen kann. Diese lassen sich dann nur mit geeigneten Schlüsseln wiederherstellen. Geschieht das Verschlüsseln vor dem Absenden der Nachricht auf Seiten des Absenders und entsprechend die Entschlüsselung auf Seiten des Empfängers, kann ein Lauscher auf Netzwerkebene nicht mehr (oder zumindest nicht ohne zusätzlichen Aufwand) die ausgetauschten Informationen mitlesen. Wird beispielsweise eine vertrauliche Information vor dem Versenden über ein kryptografisches Verfahren verschlüsselt, so können zwar die verschlüssel-

ten Daten immer noch mitgelesen werden, die Interpretation dieser Informationen ist aber nicht mehr unmittelbar möglich. Es sei hier betont, dass das Ausmaß der Sicherheit von der Stärke des Verschlüsselungsverfahrens abhängt.

4.8 Zusammenfassung

Im Laufe der Zeit haben sich eine Reihe von verschiedenen Mailsystemen entwickelt. Mit den Anforderungen der User, auch bezüglich der Datensicherheit, wuchsen die Möglichkeiten der verschiedenen Lösungen. Die Einführung multimedialer Inhalte jenseits der reinen textbasierten Nachrichtenübermittlung führte zu neuen Standards, die die Übertragung und die Handhabung von Dateien als Attachments in Mails vereinfachten. Auf der anderen Seite stiegen auch die Anforderungen an die Sicherheit des Gesamtsystems, da diese Erweiterungen zusätzliche Möglichkeiten für Angriffe aus dem Internet boten. So steht man nun vor der Herausforderung, einerseits die Möglichkeiten der neuen Architekturen zu nutzen und andererseits keine Lücken im Sicherheitssystem des Intranets entstehen zu lassen.

Die Umsetzung einer effizienten Mailtopologie orientiert sich stark an den aktuellen und zukünftigen Gegebenheiten. Eine gut geplante Mailarchitektur kann mit den Anforderungen des Unternehmens wachsen, ohne sofort eine grundlegende Architekturänderung zu erfordern.

5 News

Der News-Dienst steht eher im Schatten der populären Dienste wie Web oder E-Mail. Ein Grund dafür ist sicherlich die eingeschränkte Verfügbarkeit, die die Service-Provider gestatten. Oft kann der Anwender nicht auf das volle Angebot der weltweiten Diskussionsforen zugreifen. Zum anderen liegt es auch daran, dass die inhaltliche Qualität der Beiträge oft sehr zu wünschen übrig lässt. Darüber hinaus wird dieses Medium häufig von Adressensammlern missbraucht, die aus den Beiträgen heraus Mailadressen für Werbeaktionen sammeln oder die Diskussionsforen mit Spam-Beiträgen fluten. Damit ist es teilweise recht mühsam, im weltweiten News-Geflecht sinnvolle Informationen zu sammeln. Trotzdem wird dieser Service – zwar von weniger Anwendern, dafür aber umso intensiver – genutzt.

5.1 Einführung

Um gezielt Informationen über Netzwerke an eine oder mehrere Personen zu verteilen, verwendet man üblicherweise Mails. Handelt es sich bei den Adressaten nicht nur um einzelne Personen, sondern um ganze Interessengemeinschaften, werden so genannte Mailinglisten unter einer einzigen Mailadresse eingerichtet. Eine Nachricht an diese Adresse wird automatisch an alle verteilt, die zur Mailingliste gehören. Mailinglisten haben aber den Nachteil, dass jede Mail an alle Mitglieder dieser Liste verteilt wird. Damit wächst das Mailaufkommen und die damit verbundene Netzwerklast direkt proportional mit der Teilnehmerzahl. Insbesondere bei Mailinglisten, in denen sehr intensiv diskutiert wird, kann es dazu kommen, dass sehr viele Nachrichten an viele Personen verteilt werden. Auf der anderen Seite ist es sehr wahrscheinlich, dass diese Beiträge nicht immer von allen Personen unbedingt gelesen werden, weil beispielsweise das einzelne Thema oder der Beitrag des Absenders nicht interessiert.

Beispiel: Joe User ist Mitglied der Mailingliste »Marketing«. Dieser Mailingliste gehören viele Mitglieder an und Susi Sorglos möchte innerhalb dieser Gruppe eine Diskussion ihrer Werbestrategie anregen, die allerdings für Joe vollkommen uninteressant ist. Trotzdem bekommt er alle Mails zu diesem Thema in seine Mailbox.

Newsserver gehen einen anderen Weg. Sie stellen Beiträge einzelner Personen in Diskussionsgruppen zur Verfügung. Der Anwender kann nun selbst entscheiden, welche Beiträge er lesen möchte. Das hier zum Einsatz kommende Prinzip ist also keine passive Informationszuweisung, sondern eine aktive Informationsbeschaffung.

Um sich fach-, raum- und betriebssystemübergreifend in Diskussionsgruppen auszutauschen, wird nicht nur ein einzelner Server in diesen Datenaustausch involviert, vielmehr existiert ein weltweites Netz von Diskussionsservern, die untereinander die Diskussionsbeiträge austauschen und in den verwalteten Diskussionsgruppen einsortieren.

Ein News-System ist ein Netzwerk von verschiedenen Servern, die weltweit Diskussionsgruppen wechselseitig aktualisieren und es gestatten, sich mit Personen aus aller Welt auszutauschen. Es besteht aus vielen Rechnern, die Nachrichten zu einem bestimmten Thema über ein dediziertes Netzwerkprotokoll auszutauschen.

Ein News-Beitrag umfasst einen Header und einen Body, wie man es von der Mailarchitektur her kennt. Die Netzwerkkommunikation geschieht über ein Protokoll, das viele Ähnlichkeiten mit dem des Mailsystems aufweist. Auf der anderen Seite ist die Intention eines Diskussionsbeitrags eine vollkommen andere:

- Während eine Mail an eine bestimmte Person oder Personengruppe adressiert ist, von der man erwartet, dass sie die Nachricht liest, ist ein Diskussionsbeitrag viel weniger an direkte Wahrnehmung geknüpft. Ein Beitrag kann kommentiert werden, muss aber nicht unbedingt eine Reaktion auslösen.
- Ein weiterer Unterschied ist der Fokus der Nachricht. Mails sind direkt (privat) für die Adressaten bestimmt. Ein News-Beitrag dagegen ist für alle gedacht, die sich dafür interessieren und an dieser Diskussionsgruppe angemeldet sind.
- Mails werden entgegengenommen, bearbeitet und anschließend üblicherweise gelöscht. News dagegen werden auf dem Server in chronologischer und thematischer Ordnung für einen bestimmten Zeitraum archiviert, so dass man jederzeit frühere Nachrichten lesen und die Entwicklung der Diskussion verfolgen kann.
- Schließlich stellt die Mail einen Informations-»Push« dar, während News einem »Pull« entsprechen: Mails werden an die Adresse des Empfängers gesendet, ohne dass er diese Informationen angefordert hat. News dagegen sind ein Informationsangebot eines Servers, das man annehmen kann, aber nicht muss.

Diese unterschiedliche Zielsetzung bedingt eine vollkommen andere Verteilungsarchitektur mit einem eigenen Dienst und einem eigenen Kommunikationsprotokoll.

Ein News-Service ist ein asynchroner Informationsdienst, über den nach Themen sortiert Diskussionen stattfinden. Durch einen Netzwerkverbund können diese Debatten über lokale Grenzen hinweg weltweit geführt werden. Im Gegensatz zu Mail (Informations-Push) werden in Newsgruppen die Beiträge aktiv angefordert (Informations-Pull).

5.2 Architektur

So wie der »*http*-Teil« des Internets als »Web« bezeichnet wird, hat sich für das News-Netzwerk eine eigene Bezeichnung durchgesetzt: das »Usenet«. Es ist ein verteiltes Kommunikationssystem, mit dem Diskussionsbeiträge in Diskussionsgruppen, so genannte Foren, auf Servern hinterlegt sind. Man kann sich entweder passiv, also nur lesend, oder aktiv – lesend und schreibend – am weltweiten digitalen Meinungsaustausch beteiligen.

Betrachten wir eine Diskussion zum Thema »Der Sinn des Lebens«. Ein Newsserver stellt dieses Thema zur Diskussion und man kann diese Newsgruppe abonnieren. »Abonnieren« ist vielleicht nicht unbedingt ein treffender Begriff, da ein reales Abonnement eine passive Bereitstellung von Informationen und Daten bedeutet – bei einem Zeitschriftenabonnement bekommt man das Magazin frei Haus. Beim News-Dienst dagegen muss man aktiv die Nachrichten »abholen«. Normalerweise hat ein Abonnement auch einen kommerziellen Aspekt, der beim Newsserver in der Regel entfällt. Unter »Abonnement« im Zusammenhang mit dem Newsserver versteht man die Funktion, dass man mit seiner News-Clientsoftware die gewünschte Diskussionsrunde schnell und einfach findet.

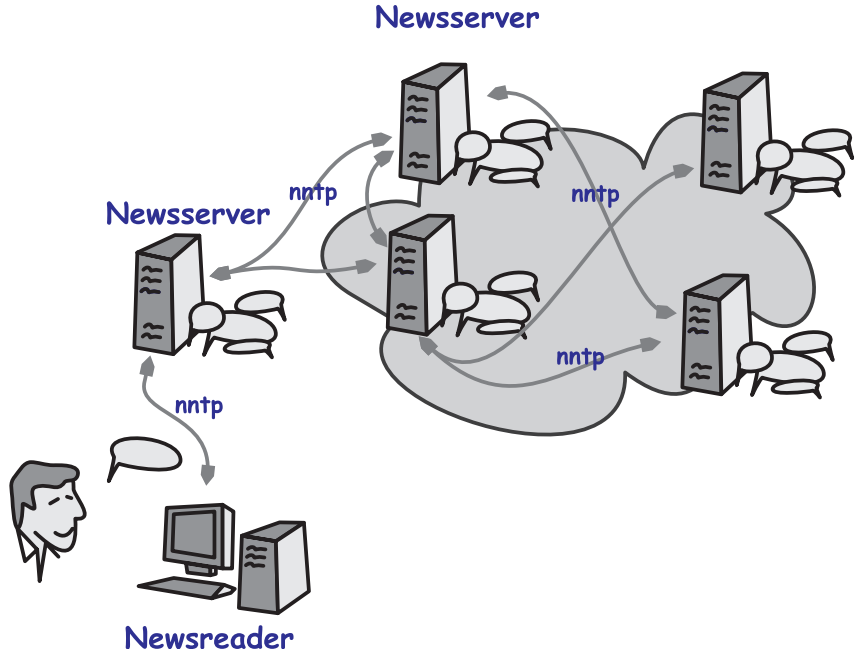
Der Vorteil an der News-Architektur ist, dass die Diskussionsbeiträge, immer kategorisiert nach den einzelnen Foren, in die ganze Welt repliziert werden. Diese Verteilung bedingt, dass nicht ein einziger Server eine bestimmte Diskussion vorhält, sondern alle Beiträge an alle Newsserver verteilt und in den einzelnen Gruppen konsolidiert werden. Jeder weitere Newsserver, der am weltweiten Usenet partizipiert, bekommt ein Abbild der Foren und der darin enthaltenen Beiträge. Entsprechend einfach findet man einen Newsserver in seiner »Nähe«, der die Möglichkeit der Teilnahme an den Diskussionen anbietet. Diese Art der Topologie bringt es mit sich, dass die Informationen innerhalb des News-Netzes keinem einzelnen Server mehr gehören, sondern Allgemeingut sind.

Natürlich trifft das bisher Gesagte nur für das Ideal einer verteilten Diskussion zu. In der Realität muss man Abstriche machen. Newsserver werden nicht nur öffentlich, sondern auch innerhalb von Intranets betrieben, es finden interne Diskussionen statt, die nie an die Öffentlichkeit gelangen. Andere Newsserver verlangen für den Zugriff eine Authentifizierung, um den eigenen Server vor zu vielen Zugriffen zu schützen. Insbesondere Internet Service Provider achten darauf, dass sich nicht jeder Anwender aus dem Internet auf dem eigenen Dienst tummelt und damit die Performance für die eigenen Kunden reduziert. Andere Anbieter beschränken sich darauf, aus dem Kaleidoskop der Diskussionsforen nur einzelne Gruppen anzubieten und nicht den vollen Umfang aller weltweit existierenden Foren. Dies geschieht auch zum Schutz vor manchen Diskussionsgruppen mit recht fragwürdiger Natur, beispielsweise Foren, in denen Dateien ausgetauscht werden, die Copyright-Bestimmungen unterliegen. Im Falle von Foren, in

denen radikale, politisch fragwürdige oder illegale Materialien ausgetauscht werden, macht sich ein Betreiber unter Umständen nur aufgrund der Tatsache strafbar, dass er diese über seinen News-Dienst anbietet.

Das Idealbild von einer weltweiten Diskussion über News unterliegt in der Realität den Reglementierungen durch den Newsprovider. Einschränkungen ergeben sich beispielsweise aus technischen, kommerziellen oder juristischen Rahmenbedingungen.

Abbildung 5.1: News-Topologie



Beiträge veralten, d.h., sie werden innerhalb einer Diskussionsrunde nach einer gewissen Zeit gelöscht. Andernfalls würden die gesammelten Beiträge im Laufe der Zeit den gesamten Speicher auf einem Newsserver auffüllen. Das Löschen der Beiträge erfolgt auf den verschiedenen Servern unterschiedlich, so dass beispielsweise auf einem Server eine bestimmte Nachricht noch vorhanden ist, während sie auf einem anderen bereits entfernt wurde.

Weiterhin verfolgen die Administratoren der einzelnen Server unterschiedliche Auffassungen von »Zensur«. So kann man davon ausgehen, dass die Betreiber ein kritisches Auge auf den Inhalt der Beiträge und Foren haben und besonders augenfällige Beiträge zügig entfernt werden.

5.3 Das Network News Transfer Protocol

5.3.1 Überblick

Innerhalb des News-Umfelds haben sich eigene Begriffe eingebürgert: Die Nachrichten sind »Postings« oder »Articles«, das Veröffentlichen der News wird als »posten« bezeichnet, Diskussionsgruppen sind »Foren« oder »Newsgroups« und wie bereits beschrieben bilden die über das Internet verbundenen Newsserver das »Usenet«. Genauer gesagt war das Usenet nicht von Beginn an im Internet enthalten. Es ist im Umfeld der Entwicklung des Unix-Betriebssystems entstanden und wurde als Netzwerk betrieben, das von Zeit zu Zeit über Wählverbindungen Daten austauschte. Das Übertragungsprotokoll war das betriebssystemeigene »Unix To Unix Copy« (UUCP). Im Laufe der Zeit entwickelten sich aber die teilnehmenden Newsserver im Usenet immer mehr in Richtung Internet, in dem sie ständig miteinander verbunden waren.

Isolierte Usenet-Server stellen die komplette, für den Betrieb von Foren notwendige Funktionalität bereit. Darüber hinaus können sie an einem News-Verbund partizipieren, indem sie lokal gepostete Artikel an die anderen Server im Verbund verteilen. Umgekehrt kann der gleiche Server News von anderen akzeptieren oder abweisen. Für die Kommunikation der Server untereinander sowie eines Servers mit einem Client steht ein besonderes Protokoll zur Verfügung: das Network News Transfer Protocol (*nntp*). Dieses Usenet-Protokoll wurde in RFC 977 [19] spezifiziert, in RFC 1036 [20] wurde das News-Format festgelegt.

Innerhalb eines News-Netzwerks kann jeder Server von jedem anderen beliebig viel von dem bereitgestellten Angebot übernehmen. Ein einzelner Server lässt sich so konfigurieren, dass er einzelne oder alle Diskussionsgruppen von einem benachbarten Server übernimmt. Im Falle einer kompletten Übernahme des News-Inhalts eines benachbarten Servers spricht man von einem »Full Feed« des Servers. Dabei hängt der Umfang dieses Feed vom Angebot des anderen Servers ab. Je nachdem, wie dessen Administrator die Bereitstellungskriterien definiert, kann es sein, dass ein Teil der weltweit existierenden Diskussionsgruppen nicht importiert werden kann, weil der lokale Server keinen News-Anbieter findet, der diese bereitstellt.

Umgekehrt teilt der lokale Server sein eigenes Angebot den benachbarten Servern mit, die dieses übernehmen können. Newsgruppen, die intern vorgehalten werden sollen, sind von diesem Angebot ausgeschlossen.

Jeder Beitrag, der auf einem Newsserver hinterlegt wird, besitzt eine eindeutige Message-ID. Mit Hilfe dieser Kennung pflegt jeder Server eine Liste seiner vorhandenen Beiträge. Dadurch wird das mehrfache Laden von gleichen Beiträgen verhindert. Über eine »Trace-List« ist der News-Server über das eigene Angebot und das der fremden Server informiert.

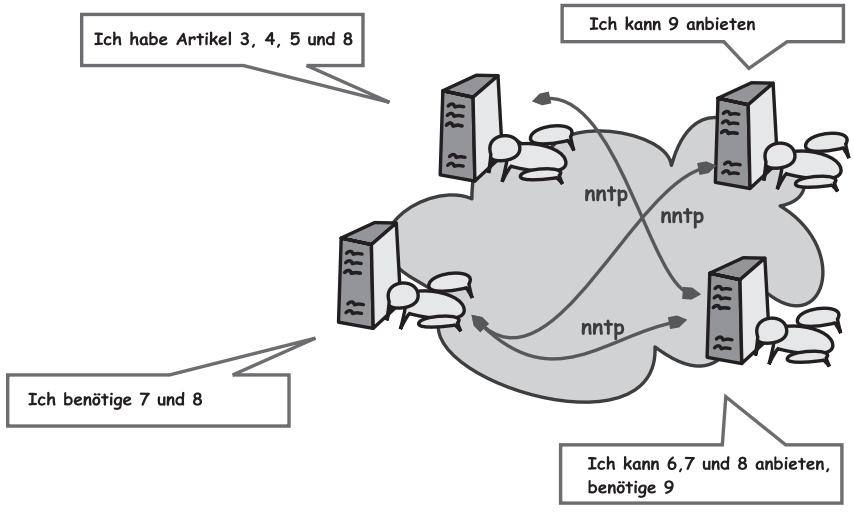


Abbildung 5.2: Austausch von Diskussionsbeiträgen

Die Verbreitung der Beiträge geschieht durch die Kommunikation der Server untereinander: Benachbarte Newsserver »unterhalten« sich untereinander und informieren sich gegenseitig, welche Nachrichten bzw. Foren sie anbieten bzw. wünschen. Diese Kommunikation erfolgt in den meisten Fällen bidirektional.

Bietet ein Nachbar eine neu eingetroffene News an, so wird sie vom anderen Server angefordert. Wird die gleiche News von einem dritten Server erneut angeboten, so wird der Beitrag abgewiesen. Diese Update-Prozedur kann sich im Bereich von Minuten bis hin zu Tagen abspielen. In welchen zeitlichen Abständen der Server Updates an seine Nachbarn verteilt, liegt in der Verantwortung des Betreibers. Dieser kann einzelne Diskussionsgruppen priorisieren oder vernachlässigen.

Die Verteilung in einem großen Verbund von Newsservern geschieht wie das Ausbreiten einer Welle nach einem Steinschlag im Wasser: Die Nachricht verteilt sich von einem zentralen Server aus in alle Richtungen.

Newsserver-Lexikon:

Article, Posting: Beitrag

Posten: Auf einem Newsserver veröffentlichen

Usenet: Netzwerk der Newsserver, die über ein entsprechendes Kommunikationsprotokoll Diskussionsforen und Beiträge austauschen

Crosspost: Ein Beitrag, der in mehr als einer Newsgruppe veröffentlicht wird

Flame: Ein beleidigender Beitrag, der meistens als Reaktion auf eine vorangegangene Meinungsäußerung zustande kommt

Moderator: Ein Administrator, der den Inhalt einer Diskussionsgruppe kontrolliert und unerwünschte Beiträge entfernt

5.3.2 Das nntp-Protokoll

Um auf einem Newsserver an den Diskussionen teilzunehmen, benötigt man einen Client, der *nntp* interpretieren kann. Dieses Protokoll ist – wie *pop3*, *imap4* oder *smtp* – textbasiert. Der Standard-Port für dieses Protokoll ist 119. Wie bei allen bisherigen Applikationsprotokollen sitzt auch hier *nntp* als Kommunikationsprotokoll auf dem TCP/IP-Stack auf.

Befehlsumfang des nntp-Protokolls

nntp verwendet eine Reihe von Befehlen, die Clients und Server zur Kommunikation nutzen. Die folgende Liste erhebt keinen Anspruch auf Vollständigkeit, da die Hersteller der verschiedenen Newsserver-Software leicht unterschiedliche Implementierungen unterstützen. Die Abweichungen vom Standard werden entsprechend markiert.

`ARTICLE [NUMMER]:`

Liefert den [NUMMER]-ten Beitrag in der selektierten Newsgroup.

`AUTHINFO [USERID] [PASSWORD]:`

Authentifizierung des Clients. Es wird die User-ID und das Passwort übermittelt. Die Bereitstellung dieser Information ist notwendig, wenn der Server vom Client eine Authentifizierung verlangt. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`BODY:`

Liefert den Inhalt eines Beitrags.

`DATE:`

Aktuelles Datum des Newsservers. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`GROUP [NEWSGROUP]:`

Selektiert ein bestimmtes Diskussionsforum mit dem Titel [NEWSGROUP].

`HEAD [NUMMER]:`

Liefert den News-Header des [NUMMER]-ten Artikels zurück.

`HELP:`

Es wird eine Übersicht über die unterstützten Befehle ausgegeben.

`IHAVE:`

Dieser Befehl wird zwischen zwei Servern übermittelt. Damit wird die Information ausgetauscht, welcher Artikel beim Server (der das Kommando abgibt) vorliegt, so dass dieser von einem anderen Server abgeholt werden kann.

`LAST:`

Innerhalb der aktuell selektierten Newsgruppe wird der zuletzt gelesene Artikel selektiert.

`LIST:`

Listet alle verfügbaren Diskussionsforen des Newsservers auf.

`LISTGROUP:`

Listet die Nummern der vorhandenen Diskussionsbeiträge innerhalb einer Newsgroup auf. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`MODE READER:`

Dieses Kommando wird von einem Newsclient übermittelt, der damit dem Server mitteilt, dass es sich hier um einen Newsreader und nicht um einen anderen Server handelt. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`NEWGROUPS [DATUM]:`

Listet die neu hinzugekommenen Newsgroups seit einem bestimmten Zeitpunkt ([DATUM]) auf.

`NEWNEWS [NEWSGROUP] [DATUM]:`

Listet die neu hinzugekommenen Artikel in einer Diskussionsgruppe ([NEWSGROUP]) seit einem bestimmten Zeitpunkt ([DATUM]) auf.

`NEXT:`

Selektiert den nächsten Artikel innerhalb der aktuellen Diskussionsgruppe.

`POST:`

Fügt einen neuen Diskussionsbeitrag zur aktuell selektierten Diskussionsgruppe hinzu.

`QUIT:`

Beendet die Newsserver-Verbindung.

`SLAVE:`

Teilt dem Server mit, dass er nicht mit einem Newsclient, sondern einem anderen Server verbunden ist. Dieses Kommando wird für den Abgleich der Newsgruppen zwischen verschiedenen Servern eingesetzt.

`STAT [NUMMER]`:

Selektiert den [NUMMER]-ten Diskussionsbeitrag innerhalb einer Newsgruppe.

`XGTITLE [NEWSGROUP-PATTERN]`:

Eine Alternative zum Befehl »List«. Der Befehl liefert eine Beschreibung der mit [NEWSGROUP-PATTERN] beschriebenen Newsgruppe. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`XHDR [ARTICLE-PATTERN]`:

Liefert den mit [ARTICLE-PATTERN] spezifizierten Teil des Headers. »XHDR Subject« liefert zum Beispiel die Betreffzeile des derzeit selektierten Diskussionsbeitrags. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`XOVER`:

Fordert eine Übersichtsdatei für die aktuelle Nachricht oder eine Gruppe von Nachrichten an. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

`XINDEX [NEWSGROUP]`:

Fordert die Indexdatei für die angegebene Newsgruppe an. (Dieses Kommando ist nicht im RFC 977 festgelegt und wird nicht von allen Servern unterstützt.)

nntp im Einsatz: Server-Modus

Da es sich beim Network News Transfer Protocol um ein Plaintext-Protokoll handelt, kann man über Telnet auf den Port 119 die vorgestellten Befehle direkt übermitteln.

Beispiel:

```
>telnet news.meinefirma.com 119
Trying 172.16.1.1...
Connected to localhost.
Escape character is '^]'.
200 news.meinefirma.com InterNetNews server INN 1.7.2 03-Aug-2001
ready
```

Der Server übermittelt eine kurze Grußmitteilung und einen Serverstatus.

Mit einem einfachen »help«-Befehl wird eine Liste von Kommandos angezeigt, die der Server akzeptiert. Der Inhalt dieser Liste hängt von dem Modus ab, in dem der Server kontaktiert wird. Beim initialen Zugriff über Telnet erfolgt die Kommunikation im Server-Modus. Dieser wird auch aktiviert, wenn der Server eine IP-Adresse erkennt, von der er normalerweise einen Newsfeed erhält. Der Modus wird über den Befehl »mode« umgeschaltet. Der Abschluss einer Übertragung erfolgt durch einen Punkt:

```
help
100 Legal commands
 authinfo
 help
 ihave
 check
 takethis
 list
 mode
 xmode
 quit
 head
 stat
 xbatch
 xpath
 xreplic

For more information, contact "usenet" at this machine.
.
```

Wie schon beim ersten Aufruf liefert der Server einen Statuscode. Nach dem erfolgreichen Verbinden über Telnet übermittelt der Server den Code 2xx mit folgender Bedeutung:

- 200 server ready – posting allowed
- 201 server ready – no posting allowed

Der Code 100 bedeutet, dass ein Help-Text übermittelt wird.

Da im jetzigen Umfeld die Verbindung im Server-Modus läuft, können wir einen anderen Newsserver simulieren, der einen Diskussionsbeitrag anbietet:

```
ihave <22334455@gw.abcde.somedomain.com>
335
From: isabella@gw.abcde.somedomain.com
Subject: Skydiving is great!
Newsgroups: rec.sports.skydiving
Distribution: world
Path: gw.abcde.somedomain.com
Date: 03 August 2001
Message-ID: <123456@gw.abcde.somedomain.com >
Body:
Hi
I want to tell the world, that I love skydiving
.
235
```

Der Ablauf der Kommunikation sieht vor, dass mit dem Ausdruck »ihave <Posting-ID>« dem Zielserver mitgeteilt wird, dass der sendende Server (in diesem Beispiel sind wir als Initiator der Verbindung in der Rolle des Servers) ein Posting hat, das eine bestimmte und eindeutige ID besitzt.

Die Statuscodes bedeuten:

- 335 send article to be transferred. End with <CR-LF>.<CR-LF>
- 235 article transferred ok

Nachdem der sendende Server das Posting angeboten hat, signalisiert der Zielserver mit dem Code 335, dass er diese Message gerne übermittelt haben möchte. Die Nachricht wird anschließend übertragen und mit einem Punkt terminiert. Der abschließende Servercode 235 quittiert dem sendenden Server, dass das Posting erfolgreich übertragen wurde.

Der empfangende Server kann auch mit einer Reihe von anderen Statuscodes antworten, je nachdem, wie er auf das Posting reagiert. Die Codes aus der Gruppe 4xx stellen allesamt ein abweisendes Verhalten oder einen Übertragungsfehler fest:

- 435 article not wanted – do not send it
- 436 transfer failed – try again later
- 437 article rejected – do not try again

nntp im Einsatz: Reader-Modus

Mit dem mode-Befehl wird zwischen Server-Modus und Reader-Modus umgeschaltet. Beide Modi kennen unterschiedliche Befehlssätze, die auf den jeweiligen Betrieb hin optimiert sind. Im Reader-Modus nimmt der Server an, dass hinter der Clientverbindung ein Newsreader verwendet wird, dessen User das Angebot des Servers abfragen möchte.

Dieser Modus kennt entsprechend andere Befehle, die mit dem »help«-Befehl aufgelistet werden können:

```
mode reader
```

```
200 news.meinefirma.com InterNetNews server INN 1.7.2 03-Aug-2001
ready (posting ok).
```

```
help
```

```
100 Legal commands

authinfo user Name|pass Password|generic <prog> <args>
article [MessageID|Number]
body [MessageID|Number]
date
group newsgroup
head [MessageID|Number]
help
```

```
ihave
last
list [active|active.times|newsgroups|distributions|
distrib.pats|overview.fmt|subscriptions]
listgroup newsgroup
mode reader
newgroups yymmdd hhmmss ["GMT"] [<distributions>]
newnews newsgroups yymmddhhmmss ["GMT"][<distributions>]
next
post
slave
stat [MessageID|Number]
xgtitle [group_pattern]
xhdr header [range|MessageID]
xover [range]
xpat header range|MessageID pat [morepat...]
xpath MessageID

Report problems to <newsmaster@news.meinefirma.com>

.
```

Die vom Newsserver unterstützte Liste der Kommandos ist im vorliegenden Beispiel herstellerabhängig und kann variieren.

Um eine Liste der vorhandenen Newsgroups zu erhalten, veranlasst man den Server mit dem List-Befehl, alle Diskussionsgruppen auszugeben:

```
list active

215 Newsgroups in form "group high low flags".

control 0000000000 0000000001 y
junk 0000000003 0000000001 y
alt.drugs 0000000000 0000000001 y
rec.sports.skydiving 00000000000000 0000000001 y
rec.sports.diving 00000000000000 0000000001 y
rec.travel.indonesia 00000000000000 0000000001 y
rec.travel.usa 00000000000000 0000000001 y

.
```

Diese Liste hat ein bestimmtes Format:

- Das erste Feld benennt die Diskussionsgruppe.
- Das zweite Feld beschreibt den ältesten Beitrag in dieser Gruppe.
- Das dritte Feld beschreibt den jüngsten Beitrag in dieser Gruppe.
- Der vierte Parameter hat zwei Einstellungen (y/n) und teilt mit, ob das Veröffentlichen von eigenen Beiträgen in dieser Gruppe erlaubt ist oder nicht.

Aus diesen Daten kann ein Newsclient-Programm dem Anwender die verfügbaren Diskussionsgruppen und die Anzahl der darin enthaltenen Beiträge in einer Übersicht präsentieren. Aufgrund dieser Informationen kann der Anwender nun entscheiden, welche Gruppen er abonnieren möchte.

Mit der vorliegenden Liste hat der Anwender noch keinen Einblick in die Übersicht der Einzelbeiträge. Diese erhält er erst mit dem Abonnement einer Diskussionsgruppe. Entscheidet er sich für eine bestimmte Gruppe, so werden die Header-Informationen der einzelnen Beiträge vom Server heruntergeladen und dem Anwender präsentiert. Anhand dieser Informationen kann er nun entscheiden, welchen Beitrag er lesen möchte.

Angesichts der hohen Zahl von Diskussionsbeiträgen in den Foren erscheint es für einen Newsclient auf Seiten des Anwenders sinnvoll, die bereits geladenen Beitragsinformationen aus vergangenen Sessions vorzuhalten. Damit erspart man dem User den erneuten Download aller Titel, Autoren und Zeitstempel von News, die beim letzten Kontakt mit dem Newsserver übertragen wurden.

Beispiel: Auf einem Newsserver wird eine Gruppe »de.comp.security.misc« geführt. Hier treffen im Laufe der Zeit neue Postings ein. Wurde diese Gruppe abonniert und vor einiger Zeit bereits kontaktiert, so werden die bereits geladenen Header-Informationen auf dem Client gespeichert. Nur die neu hinzukommenden Header werden in Folge vom Newsclient geladen (die neu hinzugekommenen Artikel sind in dieser Newsclient-Simulation fett markiert).

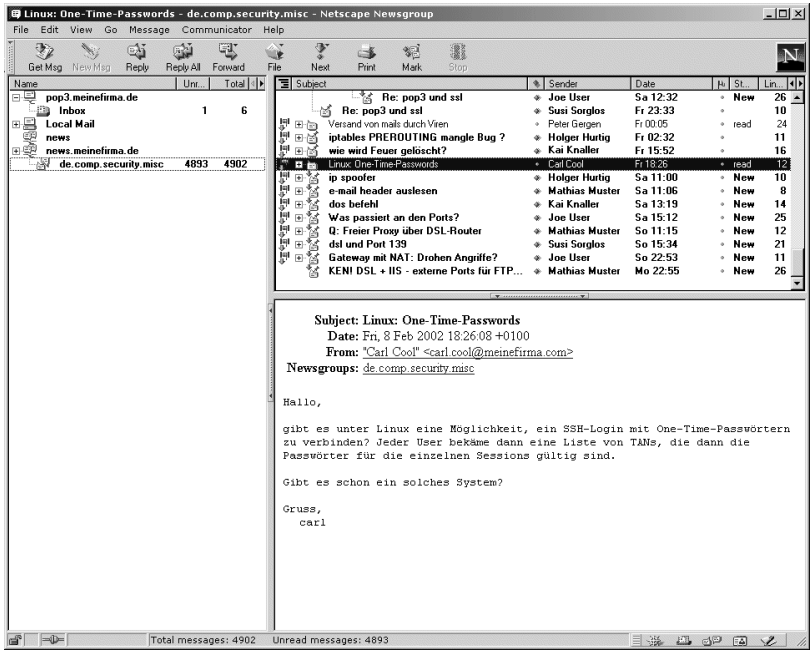


Abbildung 5.3: Gängiger News-Client mit grafischer Benutzeroberfläche

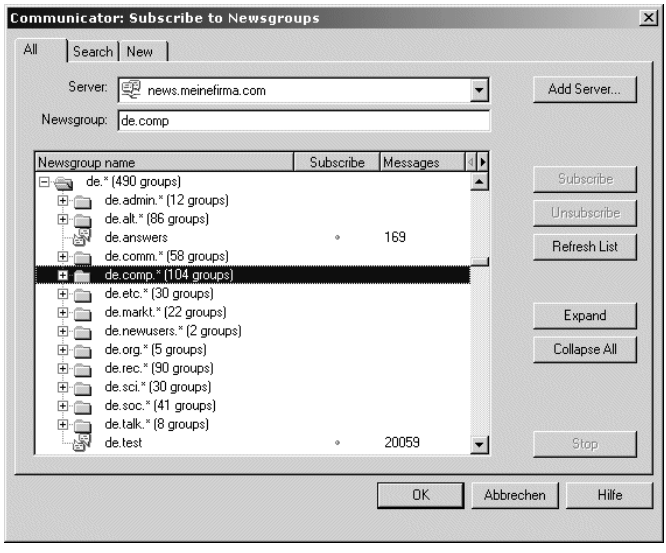


Abbildung 5.4: Auswahlbox für die Selektion der angebotenen Diskussionsgruppen

Um dies zu realisieren, trägt ein Newsreader die Liste der bereits geladenen Subjects pro abonnierter Diskussionsgruppe in Protokolldateien ein. Bei einem erneuten Kontakt derselben Gruppe werden diese Daten ausgelesen. Anschließend werden nur noch die seit dem letzten Kontakt neu hinzugekommenen Posting-Header vom Newsserver angefordert. Dies geschieht mit dem »newnews«-Befehl:

```
newnews de.alt.music 020208 000000
230 New news follows
<6f5sfd$6f$@news.meinefirma.com>
<6f7gad$4f$@news.anderefirma.com>
<6f8ofk$9f$@news.anderedomain.com>
.
```

Die Syntax dieses Befehls lautet:

```
newnews <Diskussionsgruppe> <Startdatum im Format yymmdd hhmmss>
```

Mit dieser Sequenz werden alle Subjects von Beiträgen dieser Gruppe auf den Newsreader geladen und dem User zur Ansicht präsentiert.

Wir haben bereits gesehen, wie ein Newsserver Diskussionsbeiträge akzeptieren oder abweisen kann. Es wurde auch dargestellt, wie ein Server ein Posting anbietet und der empfangende Server dieses Posting entgegennimmt. Bei dem besprochenen Beispiel handelte es sich um einen Beitrag, der bereits von einem anderen Server akzeptiert wurde und als vollständiger Diskussionsbeitrag im News-Netzwerk vorliegt.

Ein Newsclient nimmt zwar den Text und den Absender eines Postings auf und übersendet diesen Beitrag an den Newsserver, dieser aber macht aus diesen »Rohdaten« erst ein gültiges Posting. Dazu gehört unter anderem

eine Message-ID und ein Zeitstempel, die serverseitig dem vom Anwender gelieferten Text beigefügt werden.

Betrachten wir das Posten eines Artikels im Reader-Mode:

```
post
340 Ok
From: patrick@news.meinefirma.com
Subject: looking for diving partner
Newsgroups: rec.sports.diving
Body:
I would like to dive in Australia. I'm looking for a partner.
.
240 Article posted
```

Mit dem abschließenden Punkt terminiert der Anwender über den Newsclient die Eingabe und übergibt das Posting an den Server. Dieser generiert eine Message-ID, fügt weitere notwendige Header-Informationen hinzu und sortiert dieses Posting in der entsprechenden Newsgroup. Beim nächsten Kontakt mit einem anderen Server wird dieses Posting, sofern es sich nicht um einen Beitrag einer internen Diskussionsgruppe handelt, zur weiteren Verbreitung im Usenet angeboten.

Der Server kann aber auch bei einem Übertragungsfehler oder im Falle anderer Einschränkungen mit einer entsprechenden Serverstatusmeldung reagieren. Im Folgenden sehen Sie eine Liste der möglichen Servercodes:

- 240 article posted ok
- 340 send article to be posted. End with <CRLF>.<CRLF>
- 440 posting not allowed
- 441 posting failed

Diese Servercodes werden von der entsprechenden Newsclient-Software ausgewertet und dem Anwender als Fehlermeldung präsentiert.

Um eine Newsgruppe zu lesen, kann man sie über den Befehl »group« auswählen. Die Antwort des Servers ist entweder

- 211 n f l s group selected

  (n = Anzahl der angebotenen Beiträge innerhalb der Gruppe, f = Nummer des ersten Beitrags innerhalb der Gruppe, l = Nummer des letzten Beitrags innerhalb der Gruppe, s = Name der Gruppe) oder
- 411 no such news group

Über den Telnet-Client sieht dieser Befehl folgendermaßen aus:

```
GROUP local.test
211 20 1 21 rec.sports.basketball
```

Nachdem man eine Diskussionsgruppe selektiert hat, kann man sich die Header-Zeilen der Artikel anzeigen lassen. Dies erleichtert dem Anwender die Suche nach interessanten Beiträgen. Insbesondere die Betreffzeile eines Diskussionsbeitrags gibt Aufschluss über das Posting: ob es ein neuer Diskussionsbeitrag ist und was dessen Inhalt ist oder ob es sich um eine Antwort auf ein vorheriges Posting handelt. Wie im obigen Beispiel beschrieben werten gängige Newsreader diese Header-Informationen aus und listen diese dem Anwender auf: Subject, Date, Organization, Lines und From werden angezeigt, um den Beitrag zu charakterisieren.

Insbesondere das Attribut »Lines« gibt Aufschluss darüber, wie groß der Beitrag ist. Dieser Wert ist nicht die Größe des Beitrags in Byte selbst, vielmehr gibt er die Zahl der <Return> und damit der Zeilen des Beitrags wieder.

Hier der Befehl:

```
head 3
221 3 <4fidf1$8j$7@news.someserver.com> head
Path: news.someserver.com!not-for-mail
From: tony@carmel.somedomain.uk
Newsgroups: rec.travel.spain
Subject: Spain is beautiful
Date: 22 Jun 2001 13:01:39 GMT
Organization: Some Organization
Lines: 2
Message-ID: <4fidf1$8j$7@carmel.somedomain.uk >
NNTP-Posting-Host: localhost
X-Server-Date: 22 Jun 2001 13:01:39 GMT
.
```

Auf die gleiche Weise wird der Body eines Beitrags mit einem ähnlichen Befehl ermittelt:

```
body 3
222 3 <4fidf1$8j$7@news.someserver.com> body
I love Spain.
Who wants to join me for a trip to Granada?
.
```

Mit dem Befehl »article <Artikelnummer>« erhält man den Gesamtbeitrag mit Header und Body.

Mit all diesen Befehlen kann man über Telnet die gleichen Aktionen durchführen, wie sie für gewöhnlich ein Newsreader mit grafischer Oberfläche oder zumindest zeilenorientierter Benutzerführung ebenfalls verwendet.

nntp ist ein klartext-orientiertes Protokoll, das in zwei Betriebsmodi mit unterschiedlichen Befehlssätzen arbeitet: Reader und Server. Sowohl die Client/Server-Verbindung wie auch die Server/Server-Verbindung nutzen dieses Protokoll. Der Standard-Port ist 119.

5.4 News-Inhalt

Die Newsgruppen, die mit der Einführung des Usenet entstanden sind, waren in der ersten Zeit recht ungeordnet. Etwa sieben Jahre nach der Einführung dieser Architektur begann man mit dem Aufräumen des Wildwuchses. Von Juli 1986 bis etwa März 1987 wurde eine Struktur von insgesamt sieben Toplevel-Foren eingeführt. Diese waren:

comp	vorwiegend computerorientierte Themen
news	Themen in eigener Sache: Usenet und Newsgroups
sci	Science: wissenschaftliche Themengebiete
rec	Recreation: Themen zu Freizeit und Erholung
soc	soziale Themen
talk	von Polemik bis Politik alles diskussionswürdige
misc	all das, was sich in den übrigen Gruppen nicht unterbringen lässt

Unterhalb dieser Toplevel-Gruppen konnten mehrstufig Untergruppen gebildet werden, z.B.: rec.sports.soccer.fc-bayern.

So wie in vielen anderen Bereichen des Internets wurde auch hier die scheinbar willkürliche Reglementierung des Newsgruppen-Bestands nicht überall hingenommen. Nachdem der Wildwuchs einigermaßen geordnet war, wurde allgemein die Bildung von Diskussionsgruppen aus dem Bereich »Sex«, »Drugs« und »Rock'n Roll« – insbesondere unter der Toplevel-Gruppe »rec« – unterbunden. Gegner dieser »Zensur« bildeten eine eigene Toplevel-Group: »alt« für »alternative«. Heute findet sich innerhalb dieser Toplevel-Gruppe der größte Anteil des News-Aufkommens.

Dass Newsgruppen eine schier unerschöpfliche Quelle von Kontakten aus aller Welt sind, hat mittlerweile auch die kommerzielle Werbung entdeckt. So werden diese Diskussionsgruppen passiv und aktiv für solche Zwecke missbraucht. Hier werden jeden Tag Tausende von Werbepostings veröffentlicht, die vom Topic her scheinbar ein Diskussionsbeitrag sind, bei denen es sich aber in Wirklichkeit um gezielte Werbung handelt. Ein User, der in der Diskussionsgruppe »rec.travel.australia« seine Reiseerfahrungen austauschen möchte, kann von Reiseagenturen nun direkt und sehr zielorientiert beworben werden. Das Ergebnis sind oft jede Menge Spam-Mails, die unaufgefordert in der Mailbox dieses Users landen.

Um diesen unerwünschten Nebeneffekt wenigstens in Ansätzen zu bekämpfen, hat man im Laufe der Zeit Filter entwickelt, die anhand bestimmter Schlüsselwörter im Posting entscheiden, ob es sich um Spam-Beiträge oder ernst gemeinte Diskussionsbeiträge handelt. Leider scheitern aber auch seriöse Beiträge oft an diesen Filtern. Ein anderer Ansatz ist das Moderieren der Gruppen. Hier übernehmen ein oder mehrere Administratoren die Kontrolle der Postings. Sie entscheiden, welche Postings zugelassen bzw. abgewiesen werden. Je nach Diskussionsgruppe kann dies zur Sisiphus-Aufgabe wer-

den. Außerdem handelt es sich um eine Art von Zensur, die unter Umständen keine objektive Diskussion innerhalb dieser Gruppe mehr zulässt.

Die Diskussion über Zensur und Meinungsfreiheit wird nirgendwo so kontrovers geführt wie im Internet. Diese Debatte zu verfolgen, würden den Rahmen dieses Buchs sprengen. Anhang A beschreibt jedoch kurz die Argumentationen beider Seiten.

5.4.1 Anwendungsgebiete

»Wissensdatenbank«

Mit Hilfe des News-Dienstes kann man unternehmensintern eine »Wissensdatenbank« aufbauen. So werden Foren nach Themengebieten geordnet eingerichtet, um einzelne Probleme zu erörtern.

Beispiel: In der Consulting-Firma »meinefirma.com« wird der News-Dienst eingesetzt, um alle Mitarbeiter europaweit miteinander zu verbinden. Hier finden Diskussionen zu Problemen, Hilfestellungen oder auch Bugs einer Software statt. Neue Mitarbeiter finden hier eine wichtige Datenquelle, die dokumentiert, welche Probleme bereits auftraten und wie sie gelöst wurden. Der Newsserver liefert zunächst eine Übersicht der vorgehaltenen Themen. Diskussionsgruppen mit neuen Beiträgen stellt der Newsclient gesondert dar. So liegen in der Newsgruppe »intern.consult.produkte.myoffice« insgesamt 528 Beiträge auf dem Newsserver vor, von denen der Anwender fünf noch nicht gelesen hat:

```
intern.consult.produkte.catalog      230
intern.consult.produkte.clients     1022
intern.consult.produkte.mail        1021    21
intern.consult.produkte.peoplesoft   592
intern.consult.produkte.sap          290     2
intern.consult.produkte.myoffice     528     5
```

In dieser Diskussionsgruppe werden alle Probleme, Tricks und Hinweise rund um das Thema »myoffice« besprochen.

Öffnet der Anwender besagte Diskussionsgruppe, so bekommt er eine Übersicht über alle vorhandenen Beiträge. Die neuen und noch ungelesenen Beiträge sind fett markiert.

```
Problem Inst. Vers. 1.1          sammy    01.02.2002
Re: Problem Inst. Vers. 1.1       JoeU    01.02.2002
Re: Re: Problem Inst. Vers.      Tommy    02.02.2002
Suche Version 1.2 beta          Master    01.02.2002
Re: Suche Version 1.2  beta      Joshie    02.02.2002
Update v. 1.1 beta auf 1.2a      Zombie    03.02.2002
Re: Update v. 1.1 beta auf        sammy    03.02.2002
Version 1.2 auf Win2000?        Master    03.02.2002
Re: Version 1.2 auf Win2000      Gandalf    03.02.2002
```

Hier findet man meist den richtigen Ansprechpartner für ein Problem.

Solche Diskussionsgruppen sollten langfristig angelegt sein. Das heißt, die Verfallszeit der Beiträge sollte nicht zu kurz gewählt werden, damit man als Hilfesuchender auch nach längerer Zeit noch auf diese Datenquelle zurückgreifen kann.

Einen solchen Hilfe-Pool kann man intern führen, es existieren aber auch zahlreiche öffentliche Diskussionsgruppen, in denen man auf kompetente Hilfe zählen kann.

Kontaktquelle und Meinungsbildung: die digitale Kleinanzeige

Diskussionsgruppen können weltweit Menschen mit gleichen Interessen oder Vorlieben zusammenführen. Wer nach Griechenland reisen möchte und eine Reisebegleitung sucht, wird sie in einer einschlägigen Gruppe finden.

Die Vorhaltezeit der Beiträge kann hier kürzer gewählt werden, da der Inhalt der Postings meist keinen so hohen Stellenwert bzw. keine langfristige Bedeutung hat. Im Mittelpunkt steht die Kommunikation, der Newsserver ist die Plattform hierfür.

Datenaustausch

In den letzten Jahren hat sich das Usenet zunehmend als Plattform für den Datenaustausch etabliert. So werden vor allem in den alternativen Foren Bilder und Videos ausgetauscht, wobei es häufig zu Verletzungen von Copyright-Bestimmungen kommt, indem Daten, die dem Lizenzrecht oder der Geheimhaltung unterliegen, veröffentlicht werden.

In diesen Foren werden aber auch Passwörter für Dienste und Webseiten ausgetauscht oder Kontakte hierfür angeboten, ftp-Server für Raubkopien (slang: »warez«) und deren »Öffnungszeiten« angekündigt und Seriennummern mitgeteilt.

Leider erstreckt sich das Angebot auch auf das Veröffentlichen von illegalem Material. So waren einschlägige Newsgroups lange Zeit eine der ersten Quellen für Kinderpornografie.

Werbefläche

Eine weitere bedauerliche Entwicklung, die das öffentliche News-System erfährt, ist der Missbrauch für Werbezwecke. Kaum eine Diskussionsgruppe hat nicht ein Posting, in dem Werbung enthalten ist. Aus manchen Gruppen haben sich die Anwender bereits komplett zurückgezogen und das Feld den unzähligen Spammern überlassen, die die Gruppen mit ihren Beiträgen fluten und eine sinnvolle Nutzung unmöglich machen. Eine Gruppe, die eine solche Entwicklung erfahren hat, stirbt aus und wird vom Newsserver entfernt.

5.5 Zusammenfassung

Über einen News-Dienst können Personen aus verschiedenen Orten, Ländern oder Zeitzonen miteinander diskutieren. Der Service bietet eine Reihe von Themen und Diskussionen an, zu denen man Beiträge verfassen kann. Diese Beiträge werden über einen gewissen Zeitraum vorgehalten, um die chronologische Entwicklung der Diskussionen verfolgen zu können.

Die offene Architektur eines solchen News-Dienstes bietet aber auch viel Spielraum für Missbrauch. Grundsätzlich kann jeder beliebige Diskussionsbeitrag veröffentlicht werden. Meist existiert keine Kontrollinstanz, die darüber entscheidet, welcher Beitrag aufgenommen wird und welcher nicht. Moderationen von Gruppen sind zwar möglich, häufig ist aber der mit dem Überwachen einer öffentlichen Diskussionsgruppe betraute Administrator angesichts der Flut eintreffender Beiträge überfordert.

Setzt sich die Tendenz weiter in Richtung Werbefläche und Datenmissbrauch fort, so wird der öffentliche News-Dienst wahrscheinlich aussterben. In einem unternehmensinternen Netzwerk ist aber der Einsatz von News als Kommunikationsmittel ein wichtiger Beitrag, dessen Implementierung man in Erwägung ziehen sollte.

6 Certificate Authority

Das Internet ist in seiner Architektur auf offene Kommunikation ausgelegt. Dieser Ansatz war und ist einer der Faktoren, die den Erfolg dieser Technologie ausmachen. Die Kehrseite ist allerdings, dass diese offenen Protokolle selten die Privatsphäre wahren. Eine Information, die im weltweiten Netz unterwegs ist, kann praktisch überall abgefangen und untersucht werden, sei es nun berechtigt oder nicht. Für einen Schutz vor solchen Angriffen müssen vielfältige Maßnahmen ergriffen werden, die zu einem großen Teil auf Techniken aus der Mathematik basieren. Zusammengefasst werden sie unter dem Begriff »Kryptografie«.

Zunächst aber soll dargestellt werden, wo eigentlich die Gefahren liegen, die uns aus dem Netzwerk drohen.

6.1 Gefahrenpotenziale im Internet

Im Rahmen dieses Buchs beziehen sich die Begriffe Sicherheit und Risiken auf das Netzwerk und die Daten. Hier lässt sich bereits eine grobe Unterteilung vornehmen in das, was zu schützen ist:

1. Daten

   Zu Daten gehören Informationen jeder Art. Man kann sie in öffentliche, geschützte und private Daten unterteilen. Während öffentliche Daten jedem zugänglich sind und nach Belieben verwendet werden können, unterliegen geschützte Daten einem eingeschränkten Nutzungsrecht. Private Daten stehen ausschließlich einem bestimmten Benutzerkreis zur Verfügung.

   Öffentliche Daten: Zu öffentlichen Daten gehören beispielsweise Wetterinformationen und Verkehrsnachrichten, sie sind Allgemeingut und gehören niemandem. Sie können von jedem verwendet und modifiziert werden.

   Geschützte Daten: Geschützte Daten sind Informationen, bei denen der Urheber oder die veröffentlichende Instanz die Eigentümer sind. Betrachten wir dieses Buch: Der Käufer erwirbt den Datenträger, also das Papier und die Druckerschwärze. Der Inhalt, also Texte, Bilder, Grafiken, Ideen, Formulierungen und Informationen sind Eigentum des Autors und des Verlags. Damit sind diese Daten nicht »geheim« und werden nicht vor den Augen der Öffentlichkeit verborgen, sie sind lediglich geschützt. Das heißt, niemand darf sie ohne Genehmigung des Eigentümers kopieren oder in seinem Namen weiterverwenden.

Gerade in der Musikbranche machte man die schmerzliche Erfahrung, dass mit CD-Brennern und der Einführung des »mp3«-Standards immer mehr Raubkopien erzeugt werden. Im Jahr 2001 wurde erstmals mehr Musik kopiert als gekauft.

Der Inhalt eines Buchs, ein Film oder ein Musikstück stellt so genanntes geistiges Eigentum dar und unterliegt dem Urheberrecht. Um Verletzungen gegen dieses Recht vorzubeugen, müssen Schutzmaßnahmen ergriffen und umgesetzt werden.

Private Daten: Private Daten unterliegen dem Persönlichkeitsrecht, sie können nur durch den Eigentümer selbst veröffentlicht werden. Meist aber sind sie vor der Öffentlichkeit sowohl im lesenden als auch schreibenden Zugriff geschützt. Persönliche Daten müssen nicht unbedingt zu einer Person gehören. Auch Unternehmensdaten unterliegen dem Schutz, der über Geheimhaltungsvereinbarungen gegenüber anderen umgesetzt wird. Gerade diese Daten aber sind für andere sehr interessant.

Beispiele:

- Versicherungen sammeln Strukturdaten über die Bevölkerung, analysieren regionale Gehaltsprofile und Gewohnheiten der Bevölkerung.

  Vorteil: Die Versicherer können regional abgestimmte Policen anbieten, das eigene Risiko minimieren und den Profit erhöhen.

  Nachteil: Menschen in ungünstigen Bedingungen werden gegenüber den anderen benachteiligt.

- Zeitungsreporter der Regenbogenpresse versuchen permanent in die Privatsphäre von Prominenten einzudringen, um entsprechende Informationen an die interessierte Klientel weiterzugeben.

  Vorteil: Eine breite Masse an Interessenten für diese Art von Informationen wird unterrichtet und die Auflagen dieser Zeitschriften steigen.

  Nachteil: Das Persönlichkeitsrecht des Betroffenen wird massiv verletzt und für ihn ist es schwer, den allgemeinen Eindruck einer Meldung zu korrigieren, sei sie nun wahr oder nicht.

- Dem Verbraucher werden immer häufiger Bonuspunkte-Programme angeboten. Man kann beim Kauf von bestimmten Produkten Punkte sammeln, die man bei Erreichen einer bestimmten Menge gegen Bargeld oder Produktgutscheine eintauschen kann.

  Vorteil: Der Kunde bekommt Rabatte ausbezahlt. Für Unternehmen bietet dieses Prinzip die Möglichkeit, das Käuferverhalten jedes Teilnehmers genauestens zu verfolgen, Interessen und Hobbys zu erkennen und zielgruppengerecht werbewirksam Angebote zu unterbreiten.

Nachteil: Das Verhalten des Teilnehmers unterliegt genauer Beobachtung und Kategorisierung, Neigungen und Benachteiligungen können ausgewertet, Personendaten profitabel an Dritte weiterverkauft werden.

Unternehmen wie Doubleclick haben ihr Geschäftsmodell auf das Sammeln von persönlichen Daten ausgerichtet. Mit Hilfe von Cookie-Informationen, versteckten Feldern auf Webseiten oder anderen Funktionalitäten wird dabei das Surfverhalten des Anwenders analysiert. Die gesammelten Daten dienen dann dazu, die Hobbys und Neigungen der Person zu identifizieren, um sie zielorientiert zu bewerben.

Für den Einzelnen wie für Unternehmen sollte der Datenschutz eine der ersten Prioritäten im Umgang mit den Medien, insbesondere dem Internet haben. Fallen private Daten in falsche Hände, kann dies der einzelnen Person oder dem Unternehmen sehr schaden. Und doch scheitert die Vorsorge gegen diese Gefahren oft an der Fahrlässigkeit oder am Unverständnis vieler Anwender.

2. Die Ressourcen

   In Bezug auf Systemsicherheit denkt man meist an die Daten und weniger an die Ressourcen. Und doch ist hier ein hohes Gefahrenpotenzial vorhanden, das in einem Sicherheitskonzept berücksichtigt werden muss.

   Bei den Ressourcen handelt es sich nicht um Informationen, sondern um Arbeitsmittel, die den Zugriff auf die Daten ermöglichen. Ein Rechner, der als Web-Terminal an einer Autobahnraststätte steht, erhebt weniger einen Anspruch auf Daten-, als auf Ressourcenabsicherung. Er darf nicht von Unberechtigten abmontiert und wegtransportiert werden, die Tastatur darf über die angebotene Funktionalität hinaus keinen Zugriff auf das Betriebssystem gestatten. Selbst die Spannungsversorgung muss geschützt sein, um nicht Angreifern die Möglichkeit zu geben, das Betriebssystem neu zu starten.

   Bei der Sicherheit der Ressourcen möchte man in erster Linie einen Investitionsschutz erreichen, damit autorisierte Benutzer damit arbeiten können und Unberechtigte außen vor bleiben. Der Schutz vor einem unberechtigten Zugriff auf den Rechner mit den Daten steht vor dem Schutz der Daten selbst.

3. Reputation
   Als immaterieller Wert ist die Reputation, der gute Ruf, nur schwer über eigene Mechanismen zu schützen. Trotzdem muss sich ein Unternehmen auch in diese Richtung absichern. Finden sich in den Logfiles eines Webservers mit fragwürdigem Inhalt die Zugriffsdaten eines angesehenen Unternehmens und wird diese Information veröffentlicht, gerät nicht nur der Ruf des einzelnen Mitarbeiters in Gefahr, sondern auch der der gesamten Firma. Ein ähnlicher Fall war das viel diskutierte Compu-

Serve-Urteil, als sich illegale Daten auf einem Newsserver des Providers fanden. Plötzlich war das gesamte Unternehmen in den Negativschlagzeilen.

Um einen effizienten Schutz der Reputation zu erreichen, müssen die ersten beiden Punkte, also Schutz der Daten und der Ressourcen, effizient umgesetzt und in die allgemeine Sicherheitspolitik eingebettet sein.

Die nächsten Abschnitte haben das Ziel, die notwendigen Werkzeuge zum Schutz dieser drei Bereiche zu erarbeiten.

6.2 Aufklärung der Anwender

Einer der ersten Schritte bei der Einführung eines Sicherheitskonzepts ist die Aufklärung des Benutzers. Selbst die ausgefeiltesten technischen Gegenmaßnahmen zum Datenschutz scheitern an der mangelnden Kooperation der Mitwirkenden.

Beispiel: In einem Unternehmen wird mit viel Geld und Aufwand eine Firewall im Netz installiert, die das Unternehmen vor unberechtigtem Datenzugriff schützt. So sind Telnet-Zugriffe über die Firewall nicht gestattet, Mail darf nur über einen dedizierten Internet-Mailrelay über die Firewall ins interne Netz geroutet werden, ftp-Zugriffe von außen sind nicht erlaubt.

Ein Mitarbeiter erhält eine Nachricht mit einem Attachment mit der Bezeichnung »Klick_mich_und_du_siehst_Gruene_Maennchen.exe«. Dieser Aufforderung kommt er auch gleich nach. Er sieht grüne Männchen, die sich in rote Kleckse verwandeln. Unbemerkt hat sich aber mit diesem Programm ein Virus installiert, das die geheimen Produktentwicklungspläne an die Konkurrenz mailt.

Seit dem »I Love You«-Virus ist sicherlich den meisten Benutzern klar, dass man Programme unbekannter Herkunft in Mails nicht ausführt. Das allein reicht aber nicht aus. Viele Passwörter finden sich immer noch auf Post-its unter der Tastatur. Zahlreiche Internetseiten mit Spielen und Gimmicks sind virenverseucht und werden dennoch ohne Bedenken angeklickt. Solche Zustände können die Sicherheitspolitik eines Unternehmens komplett in Frage stellen.

Es ist sicherlich sinnvoll, dem Anwender ein umfangreiches technisches Verständnis für sein Arbeitsmittel, den Computer, zu vermitteln. Viel wichtiger aber ist es, ihn für potenzielle Sicherheitsrisiken zu sensibilisieren und ihm vor allem eine gesunde Portion Misstrauen gegenüber Applikationen nahezulegen, die von ihm sicherheitsrelevante oder geheime Daten abverlangen.

Die Methode, wichtige Daten unter Vorspiegelung falscher Tatsachen zu erlangen, bezeichnet man als »Social Engineering«:

- Etliche AOL-Mitglieder erhielten vor einiger Zeit eine Mail, die angeblich vom AOL-Operating-Team stammte. Der Anwender wurde aufge-

fordert, sein Passwort an eine bestimmte Adresse zu mailen, da sonst der Account deaktiviert werden würde. Viele Benutzer kamen dieser Aufforderung nach und mussten anschließend den Missbrauch ihres Accounts feststellen.

- Auf manchen Webseiten werden kostenlose Bilder offeriert. Der Zugriff darauf ist aber nur möglich, wenn man einen Bilderbrowser oder eine Zugangssoftware installiert. In Wirklichkeit baut dieses Programm die bis dahin bestehende Internetverbindung ab und eine neue, viel teurere Verbindung auf.
- Internetseiten lassen sich so programmieren, dass sie Grafiken erzeugen, die vorgeben, vom Betriebssystem zu stammen.

  Beispiel: Ein Anwender surft auf eine Internetseite und plötzlich wird ein Fenster angezeigt, das ihn auffordert, sein Kennwort des Betriebssystems oder für den Internetzugang zu ändern. Die Aufmachung der Grafik suggeriert, dass es sich um eine Aufforderung des Betriebssystems handelt. In Wirklichkeit wird die Eingabe des Anwenders über das Internet an eine andere Person übertragen, die sich mit dieser Information Zugriff auf den Rechner oder den Internetaccount verschaffen möchte.

In solchen Fällen kann man kaum mit Hilfe von Soft- oder Hardware Schutzmechanismen aufbauen, hier hilft nur die Aufklärung des Anwenders.

6.3 Steganografie

Im Laufe der Geschichte wurden die unterschiedlichsten Strategien zur Geheimhaltung von Daten entwickelt. Eine der ersten ist im eigentlichen Sinne keine Verschlüsselung, bei der Steganografie handelt es sich vielmehr um ein Verstecken der Daten.

Eine Information wird an einer Stelle hinterlegt, wo sie niemand vermuten würde. In der Antike wurden Sklaven die Haare abrasiert, eine Botschaft auf den Kopf tätowiert und so lange gewartet, bis die Haare wieder nachgewachsen waren, bevor man sie durch feindliche Linien schickte. Heute gibt es Algorithmen, die Daten mit Hilfe von geringsten, für das menschliche Auge kaum wahrnehmbaren Nuancen der Farbtiefe in Bildern verstecken. Die Daten werden aber nicht weiter verschlüsselt, sondern liegen im Klartext vor.

Allen gemein ist, dass ein aufmerksamer oder gewissenhafter Beobachter bei diesen Verfahren die Daten entdecken kann. Legt man eine geheime Nachricht ins Dachgestühl der Frauenkirche in München unter einen Ziegel im linken Turm, wird sie im Sinne der Steganografie versteckt. Wird sie stattdessen durch mathematische Algorithmen unkenntlich gemacht und mitten auf den Marienplatz gelegt, spricht man von Verschlüsselung.

Wo kann nun diese Art des Datenschutzes eine Rolle spielen? Betrachten wir hierzu die Datenübertragung im Internet: Daten werden von einem Server

zu einem Client und umgekehrt übertragen. Beim Homebanking ist die Intention der Verbindung klar und jeder aufmerksame Beobachter einer solchen Verbindung würde vermuten, dass es sich hierbei um eine Transaktion von Geldmitteln handelt. Damit ist das Verstecken der Daten bei dieser Verbindung nicht möglich und Steganografie stellt keine Lösung für dieses Problem dar.

Plant man einen Banküberfall oder möchte man den Grundriss von Fort Knox per E-Mail austauschen, kann man die Daten verschlüsseln, bevor man sie versendet. Wird man allerdings vom Geheimdienst überwacht, während man verschlüsselte Daten austauscht, gerät man schnell in den Verdacht, etwas Illegales zu planen. Mit Hilfe der Steganografie kann man stattdessen Urlaubsbilder austauschen, die in Wirklichkeit die Schwachpunkte der Tresore kennzeichnen.

6.4 Verschlüsselung

Im Rahmen der Verschlüsselung werden Informationen reversibel aus einem lesbaren in einen unlesbaren Zustand gebracht. Ziel ist es, den Informationsgehalt vor Unbefugten zu verstecken und für Berechtigte zugänglich zu machen. Dies alleine ist die Aufgabe der Kryptografie, die sich seit mehr als zweitausend Jahren mit dieser Aufgabe befasst und dafür bis heute noch keine hundertprozentig befriedigende Lösung gefunden hat.

Im Folgenden werden die aktuellen Verfahren der Kryptografie beschrieben, die im Internet Verwendung finden. Für die Chiffrierung sind sehr viele Architekturen und Algorithmen auf dem Markt, die unterschiedlich bewertet werden müssen. Einige der Produkte sind absolut unsicher und schützen die verborgenen Informationen praktisch gar nicht. Andere erscheinen stark, sie widerstehen einem aufwändigeren Angriff auf die damit verschlüsselten Daten mehr oder weniger lang. Die stärksten Algorithmen sind die, für die bis heute noch kein Hinweis auf eine Entschlüsselung von Geheimnissen vorliegt.

Produkte von Anbietern, deren Sicherheitskonzept auf der Geheimhaltung der zugrunde liegenden Algorithmen basiert, werden häufig sehr schnell als unbrauchbar entlarvt. Die allgemein anerkannte Philosophie ist, dass die Chiffriermechanismen öffentlich sein müssen, damit

1. niemand aus dem Unternehmen des Herstellers einen geheimen Algorithmus publizieren kann, was alle darauf basierenden Sicherheitskonzepte ad absurdum führen würde,
2. jeder sich davon überzeugen kann, dass die verwendete öffentliche Architektur keine Hintertüren hat, die es Dritten ermöglichen, an damit geschützte Daten zu gelangen,
3. jeder die Algorithmen testen kann, um möglichen Schwächen auf die Spur zu kommen.

Allgemein gilt, dass die am ausgiebigsten getesteten Verfahren die sichersten sind. Neue Verfahren müssen sich erst einer langjährigen Prüfung unterziehen und viele erfolglose Angriffe auf ihre Architektur vorweisen, bevor sie in der Öffentlichkeit als vertrauenswürdig eingestuft und akzeptiert werden.

6.4.1 Algorithmen

Verschlüsselungstechniken wurden bereits in der frühen Antike zur Nachrichtenübermittlung verwendet. Man verbirgt Informationen vor den Augen Dritter, indem man sie in ein Format bringt, das den Sinn der Information nicht mehr erkennen lässt.

Bevor die Daten verschlüsselt werden, liegen sie im Klartext vor (selbst wenn sie akustisch übertragen werden, spricht man von Klartext). Mittels eines vorher vereinbarten Algorithmus werden sie verschlüsselt. Das Ergebnis bezeichnet man als Cipher-Text oder Chiffrat.

Eine Verschlüsselung ist reversibel, das heißt, ein Chiffrat wird mittels einer Umkehrfunktion wieder entschlüsselt und eindeutig in den Klartext überführt.

Eine Art, eine Nachricht zu chiffrieren, ist die Transposition: Die ersten elf Wörter der amerikanischen Unabhängigkeitserklärung in deutscher Übersetzung (also: »Wenn es im Lauf menschlicher Gegebenheiten fuer ein Volk noetig wird«) kann überführt werden in »wnei lumn clce ggbn etuu rivl neiw rens mafe shih ree hief eeno kotg id«. Dieser Term erscheint zunächst vollkommen sinnfrei, die Entschlüsselung ist aber recht einfach: Der Chiffretext wird in zwei Teile geteilt und untereinander gelegt.

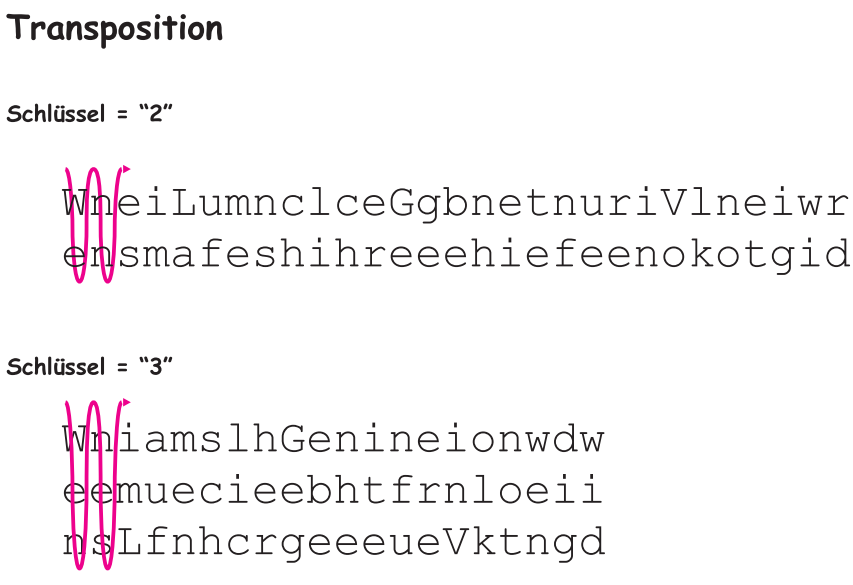


Abbildung 6.1: Transpositionsverfahren

Verwürfelungen dieser Art werden als Anagramme bezeichnet. Einige sehr schöne Beispiele werden in [29] beschrieben.

Eine andere Methode, Informationen zu verschlüsseln, ist die Substitution: Der gleiche Satz »Wenn es im Lauf menschlicher Gegebenheiten...« wird in »mudd uiyc bqkv cudi sxby sxuh wuwu rudx uyju dvku huyd leba deuj ywmy ht« umgewandelt. Dies geschieht, indem man in einer fest definierten Form einen Buchstaben gegen einen anderen austauscht.

Im vorliegenden Beispiel haben wir zwei Alphabete um zehn Buchstaben gegeneinander verschoben:

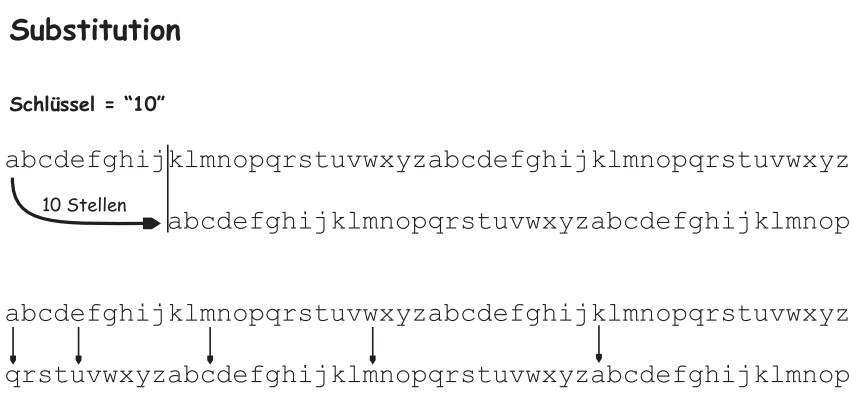


Abbildung 6.2: Substitution bei einer monoalphabetischen Verschlüsselung

Ausgehend vom Klartext werden dessen Buchstaben individuell durch die zugehörigen Buchstaben des verschobenen Alphabets ersetzt.

Diese Form der Verschlüsselung hat einen berühmten Namen: Cäsar-Chiffrierung. Der römische Feldherr soll sich dieser Chiffriermethode bedient haben, um geheime Daten an seine Generäle zu senden. Da man hierfür nur ein Substitutionsalphabet verwendet, bezeichnet man sie auch als monoalphabetische Verschlüsselung.

Man erkannte sehr schnell, dass diese Art der Geheimhaltung recht unsicher ist. Jemand, der das Chiffrat abfängt, braucht lediglich 26 Verschiebungen des Alphabets auszuprobieren und man hat den Klartext hergestellt.

Diesem Schwachpunkt widmete sich ein Verfahren des französischen Diplomaten Blaise de Vigenère (*1523). Im Rahmen der gleichnamigen Verschlüsselung wählt man ein Code-Wort, beispielsweise »Hund«. Man bringt die Buchstaben in alphabetische Reihenfolge (»dhnu«), erstellt die Alphabetreihen in dieser Reihenfolge und substituiert den Klartext Buchstabe für Buchstabe durch den jeweils darauf folgenden im verschobenen Alphabet.

Mit unserem Satz »Wenn es im Lauf menschlicher Gegebenheiten...« kommen wir damit zu »zlah hzvg ohhz qlam foyc forl jlty elab hqgy qmhy ulvi yvye qvrn lnjc uk«.

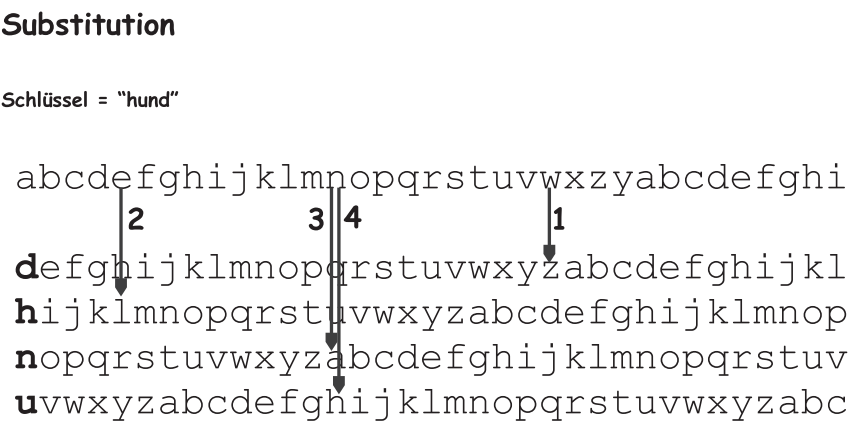


Abbildung 6.3: Vigenère-Verschlüsselung am Beispiel des Schlüsselworts »Hund«

Zur damaligen Zeit galt diese Verschlüsselung als sicher. Es schien, als könne man ohne Kenntnis des Code-Worts das Chiffrat nie mehr in den Klartext zurückführen.

Hier soll nun nicht auf die verschiedenen Methoden eingegangen werden, die rund um diese Art der Verschlüsselung entstanden. Hierfür gibt es in der Literatur sehr viele gute Referenzen, die dies detailliert beschreiben (Beispiele: [26], [29], [30]).

Vielmehr soll hier ein Prinzip dargestellt werden, das in allen Beispielen eine Rolle spielte: Ausgangspunkt war in allen Fällen ein Klartext, der über einen bestimmten Algorithmus in ein Chiffrat überführt wurde. Der Algorithmus wurde von einem so genannten Schlüssel voreingestellt, bevor der Klartext darauf angewandt wurde. Bei der Transposition war der Schlüssel die Zahl 2 (jeder zweite Buchstabe rekonstruierte den Klartext). Die Caesar-Verschlüsselung arbeitete in unserem Beispiel mit der Zahl Zehn. Und bei der Vigenère-Verschlüsselung kommt man über den Schlüssel »Hund« wieder zurück zum Klartext.

Ein bestimmter Verschlüsselungsalgorithmus hat damit immer das gleiche Aussehen. Die Individualisierung geschieht über den verwendeten Schlüssel, der die offenen Parameter einer solchen Architektur einstellt. Man kann sich das anhand der Analogie eines Haustürschlosses vorstellen, das immer nach dem gleichen Prinzip arbeitet. Die Voreinstellung des Schließzylinders ist individualisiert und der Mechanismus funktioniert nur mit einem bestimmten Schlüssel.

6.4.2 Schwachpunkte

Die Qualität eines Verschlüsselungsalgorithmus entspricht dem Aufwand, der erforderlich ist, um den Algorithmus zu brechen oder das Chiffrat zu entschlüsseln. Hier handelt es sich sehr wohl um einen Unterschied. Man kann jedes Chiffrat entschlüsseln, wenn man nur die verschiedenen Möglichkeiten aller möglichen Schlüssel ausprobiert. Gehen wir von der

Vigenère-Verschlüsselung aus und probieren alle möglichen Kombinationen aus Buchstaben, beginnend bei a, aa, ab, ... huna, hunb, hunc, ... so gelangen wir nach 1.095.486.336 Versuchen zu dem Wort »Hund« und der Klartext erscheint. Dieses simple Ausprobieren von Schlüsseln bedeutet nicht, dass der Algorithmus selbst gebrochen ist, sondern nur, dass man mit viel Fleiß irgendwann einmal auf den Klartext kommt. Diese Art des Angriffs auf ein Chiffrat wird als »Brute Force Attack« bezeichnet. Jeder Algorithmus liefert den Klartext mit dem richtigen Schlüssel. Die Kunst des Chiffrierens ist es, ihn in der Mannigfaltigkeit der Schlüsselmenge zu verstecken.

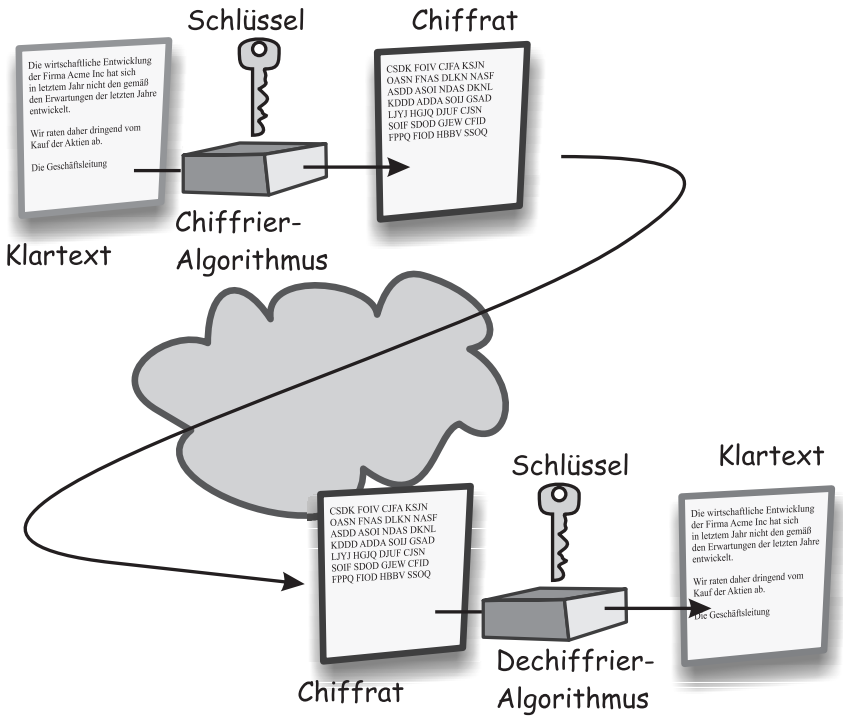


Abbildung 6.4: Chiffrierungsmodell

Ein Algorithmus gilt dann als gebrochen, wenn es einen Weg gibt, die Menge der möglichen Schlüssel, die für das Dechiffrieren in Frage kommen, zu reduzieren. Auf diese Weise lässt sich der Klartext mit einem endlichen Aufwand wieder herstellen. Dies ist die Kunst der Kryptoanalyse.

Häufigkeitsanalyse

Bei der Caesar-Chiffrierung lohnt sich fast keine Kryptoanalyse, da die Menge der verschiedenen Schlüsseln mit 26 sehr überschaubar ist. Trotzdem schauen wir uns einmal die Charakteristik des Chiffrats an.

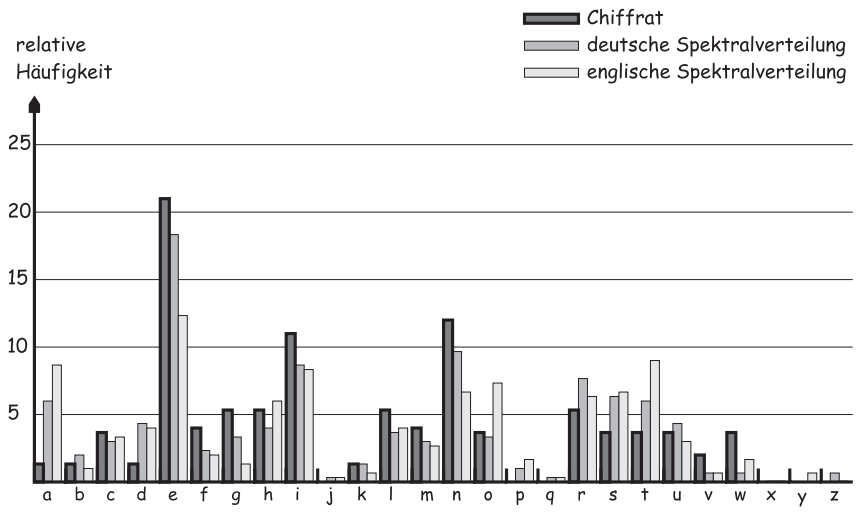


Abbildung 6.5: Häufigkeitsanalyse– Vergleich des Beispieltextes anhand einer Statistik der deutschen und englischen Sprache

Ein simpler Trick ist das Zählen der Buchstaben. In jeder Sprache gibt es relative Häufigkeiten, die sehr charakteristisch sind. Philippinische Sätze kommen fast ohne Konsonanten aus, Deutsch weist einen hohen Anteil der Buchstaben »e« und »n« auf. Abbildung 6.5 zeigt, dass die Häufigkeitsverteilung der Buchstaben bereits bei einem sehr kurzen Satz verrät, dass es sich um einen deutschen Text handelt. Würde man die amerikanische Originalfassung nehmen, käme das Spektrum der englischen Charakteristik näher.

Bei einer 1-zu-1-Ersetzung der Buchstaben in der Caesar-Chiffrierung verrät die Statistik der relativen Häufigkeiten sehr schnell, dass bei einer Verschiebung der Alphabete um zehn Buchstaben das »e« gegen das »u« und das »a« gegen das »q« vertauscht wurden. Mit dieser Information wird das Dechiffrieren eines Textes sehr einfach.

Bei der Vigenère-Verschlüsselung scheint diese Charakteristik verwischt zu sein, aber die Auflösung ist auch hier sehr einfach:

1. Man macht die Häufigkeitsanalyse wie bei der Caesar-Chiffrierung, kommt aber bei diesem Beispieltext, der mit »hund« vigenère-verschlüsselt wurde, nicht weiter.

2. Man macht eine Häufigkeitsanalyse jedes zweiten (dritten, vierten ...) Buchstabens.

Im Falle einer Chiffrierung mit einem Schlüssel von vier Buchstaben Länge liefert eine Häufigkeitsanalyse, die jeden vierten Buchstaben berücksichtigt, ein charakteristisches Spektrum einer bekannten Sprache, da alle diese Buchstaben mit dem gleichen Alphabet verschlüsselt wurden. Damit sind die Schlüssellänge und die Sprache bekannt. Der Rest ist einfach.

In EDV-Zeiten werden solche Analysen automatisiert. Die Häufigkeitsverteilung liefert dabei ein sehr gutes Entscheidungskriterium für einen Rechner, wenn er einen Klartext gefunden hat.

Schlüssellängen

Bereits bei der Caesar-Chiffrierung war zu sehen, dass bei einer geringen Menge an verfügbaren Schlüsseln die Dechiffrierung relativ einfach wird. Dies soll nun noch deutlicher dargestellt werden.

Jeder verschlüsselte Text ist dechiffrierbar, wenn man den Schlüssel kennt. Je mehr mögliche Schlüssel vorhanden sind, umso einfacher wird es, den einzig richtigen zu verstecken. Im Folgenden wird gezeigt, wie man die Menge an möglichen Schlüsseln abschätzen kann.

Nehmen wir einen Schlüssel in der EDV-Terminologie. Die kleinste Informationseinheit ist ein Bit. Dieses hat zwei mögliche Einstellungen: Eins und Null. Der Umfang der Schlüsselmenge beträgt damit 2, die Wahrscheinlichkeit, bei einem Griff in diese Menge den richtigen Schlüssel zu finden, beträgt 1/<Umfang> = 1/2 = 0,5.

Nehmen wir als Schlüssel einen Buchstaben (jedes alphanumerische Zeichen ist mit 7 Bit darstellbar), finden sich 2^7 = 128 verschiedene Schlüssel. Die Wahrscheinlichkeit, den richtigen auf Anhieb zu finden, beträgt 1/128 = 0,0078125, ein Aufwand, den man noch von Hand bewältigen kann.

Wählt man fünf Buchstaben als Schlüssel, wobei nun alle darstellbaren Zeichen zur Verfügung stehen (indem 8-Bit-Zeichen verwendet werden), hat man 109.951.627.776 verschiedene Schlüssel zur Verfügung. Das entspricht in etwa dem zehnfachen Alter des Universums in Jahren, ist aber für moderne Computer noch keine wirkliche Herausforderung.

Heutige Datenübertragungen zwischen Kunden und Banken im Homebanking-Bereich beginnen etwa bei 128 Bit, einer Schlüssellänge von 16 Zeichen. Damit hat man etwa $3{,}4 * 10^{38}$ verschiedene Schlüssel zur Verfügung. Nimmt man nun eine Million Rechner mit je einer CPU, die eine Million Rechenoperationen pro Sekunde ausführen kann, würde man etwa 10^{18} Jahre benötigen, um alle Schlüssel auszuprobieren. Nun ist aber unser Universum erst $12{,}5 * 10^9$ Jahre alt.

Hier noch ein paar Zahlenbeispiele:

Schlüssellänge	Anzahl der Schlüssel, die damit darstellbar sind
170 Bit	etwa Anzahl der Atome auf der Erde
190 Bit	etwa Anzahl der Atome auf der Sonne
223 Bit	etwa Anzahl der Atome in unserer Galaxie
265 Bit	etwa Anzahl der Atome im Universum

Wirklich wichtige Daten werden heute mit 1024, andere sogar mit 2048 Bit verschlüsselt, für die entsprechenden Schlüsselmengen bietet sich kein physikalischer Vergleich mehr an.

Leider handelt es sich hier um eine Idealvorstellung. Angenommen, man lässt den Rechner einen zufälligen Schlüssel wählen. Schon von diesem Ansatz her kann man keinen wirklich sicheren Schlüssel erwarten, da der Computer immer auf eine mathematische Funktion zurückgreifen muss, die auf den ersten Blick recht zufällige Ergebnisse liefert, aber einer genaueren Kryptoanalyse nicht standhält. Für die Erzeugung tatsächlich zufälliger Zahlen sind Computer denkbar ungeeignet. Als deterministische Geräte liefern sie bei einem definierten Ausgangszustand und den voreingestellten Randbedingungen reproduzierbare Ergebnisse, was bereits als Ausgangspunkt für einen Angriff auf ein Chiffrat dienen kann.

Wirklich zufällige Zahlen kann die Physik liefern: Der radioaktive Zerfall von Atomkernen, die brownsche Molekulardiffusion von Molekülen in Flüssigkeiten oder das thermische Rauschen von Widerständen sind nicht reproduzierbar. Messmethoden dieser Größen können als Schlüsselwerte herangezogen werden.

Known-Plaintext-Angriffe

Im Zweiten Weltkrieg verwendeten die deutschen U-Boote zur Übermittlung ihrer Standorte und der Angriffspläne einen symmetrischen Algorithmus, der den Alliierten sehr viel Kopfzerbrechen bereitete: die Enigma. Hierbei handelte es sich um eine mechanische Apparatur, die mit drei, später vier Walzen bestückt war, in denen sich intern eine Verdrahtung befand, um die einzelnen Buchstaben zu verwürfeln. Mit jedem Buchstaben der Chiffrierung veränderte sich die Lage der Walzen untereinander, so dass es zu keiner monoalphabetischen Verschlüsselung kam. Vor dem Auslaufen wurde zwischen dem Hauptquartier in Berlin und dem U-Boot die Voreinstellung der Walzen vereinbart, sozusagen ein Schüsselaustausch. Damit waren beide Seiten in der Lage, Daten zu ver- und entschlüsseln. Die Nachrichten wurden zwar über Radiowellen übertragen, konnten aber von den Alliierten zunächst nicht entschlüsselt werden. Dies änderte sich allerdings, als ihnen mehrere U-Boote und damit die ersten Enigma-Maschinen in die Hände fielen.

Hier zeigte sich zunächst die Stärke der Verschlüsselung: Mit den Maschinen hatten die Alliierten zwar den Verschlüsselungsalgorithmus (der auch zum Dechiffrieren benutzt wurde), aber ohne die Schlüssel über die Voreinstellungen der Walzen konnten sie noch nicht die abgefangenen verschlüsselten Meldungen nach Berlin dechiffrieren.

Der erste Ansatz war natürlich, zu verstehen, wie die Maschinen funktionierten, um offensichtliche Schwachstellen anzugreifen. Man fand heraus, dass die Zahl der möglichen Schlüssel, die verwendet wurden, deutlich geringer als angenommen war. Entscheidende Hilfestellung erfuhren die Alliierten allerdings durch einen Bedienfehler der Deutschen: Bei jeder Initiierung des Funkverkehrs aus einem U-Boot heraus wurden Wetterdaten

übermittelt. Damit hatten die Alliierten eine Vorstellung davon, was im Chiffrat zu Beginn stand. Darauf basierend ließ sich ein Angriff auf die Chiffrierung starten. Entscheidend ist dabei, dass es eine Fülle von Daten gab, die man miteinander vergleichen konnte.

Einfache Anwendung eines Known-Plaintext-Angriffs: Daten, die mit gleichem Schlüssel und dem gleichen Algorithmus chiffriert werden, sollten den gleichen Geheimtext liefern. Wie ändern nun leicht abgewandelte Plaintext-Daten das Chiffrat? Auswertungen dieser Art bieten sich gerade im Internet beim Verbindungsaufbau zwischen Clients und Servern an, die Protokolle sind standardisiert und laufen immer nach dem gleichen Muster ab. Dadurch wird eine geschützte Verbindung angreifbar.

6.4.3 Symmetrische Verschlüsselung

Bisher wurden ausschließlich Verschlüsselungen beschrieben, bei denen der Chiffrier-Schlüssel der gleiche wie der Dechiffrier-Schlüssel war. Eine solche Architektur bezeichnet man als symmetrische Verschlüsselung.

Von den ersten Implementierungen in der Antike bis heute hat man immer auf diese Architektur zurückgegriffen. Die Algorithmen haben sich geändert, die zugrunde liegende Mathematik wurde immer komplexer, aber die Grundkonzeption sich bis heute nicht geändert.

Es gibt in diesem Zusammenhang auch eine symmetrische Verschlüsselung, die nachweislich sicher ist: das One Time Pad.

One Time Pad

Das One-Time-Pad-Verfahren basiert darauf, dass eine Klartext-Nachricht mit einem Schlüssel chiffriert wird, der genauso lang ist wie die Nachricht selbst und der nur ein einziges Mal verwendet wird.

Aus diesem Grund ist es für den Hacker nicht möglich zu entscheiden, welcher der sinnvollen bzw. sinnlosen Texte aus der Menge der dechiffrierten Nachrichten der richtige ist.

Abbildung 6.6 demonstriert die Ver- und Entschlüsselung eines Textes. Der Originaltext ist eine geheime Nachricht der Geschäftsleitung. Ein Lauscher, dem diese Notiz des Chiffrats in die Hände fällt, probiert alle möglichen Schlüssel aus. Er kann aber theoretisch nicht nur den Originaltext rekonstruieren, sondern auch jeden beliebigen anderen Text (im Beispiel die einleitenden Worte der amerikanischen Unabhängigkeitserklärung). Anders gesagt, es wird immer einen Schlüssel geben, der jeden beliebigen Klartext erzeugt.

Aus diesem Grund ist es für ihn nicht möglich zu entscheiden, welcher der sinnvollen bzw. sinnlosen Texte aus der Menge der dechiffrierten Nachrichten der richtige ist.

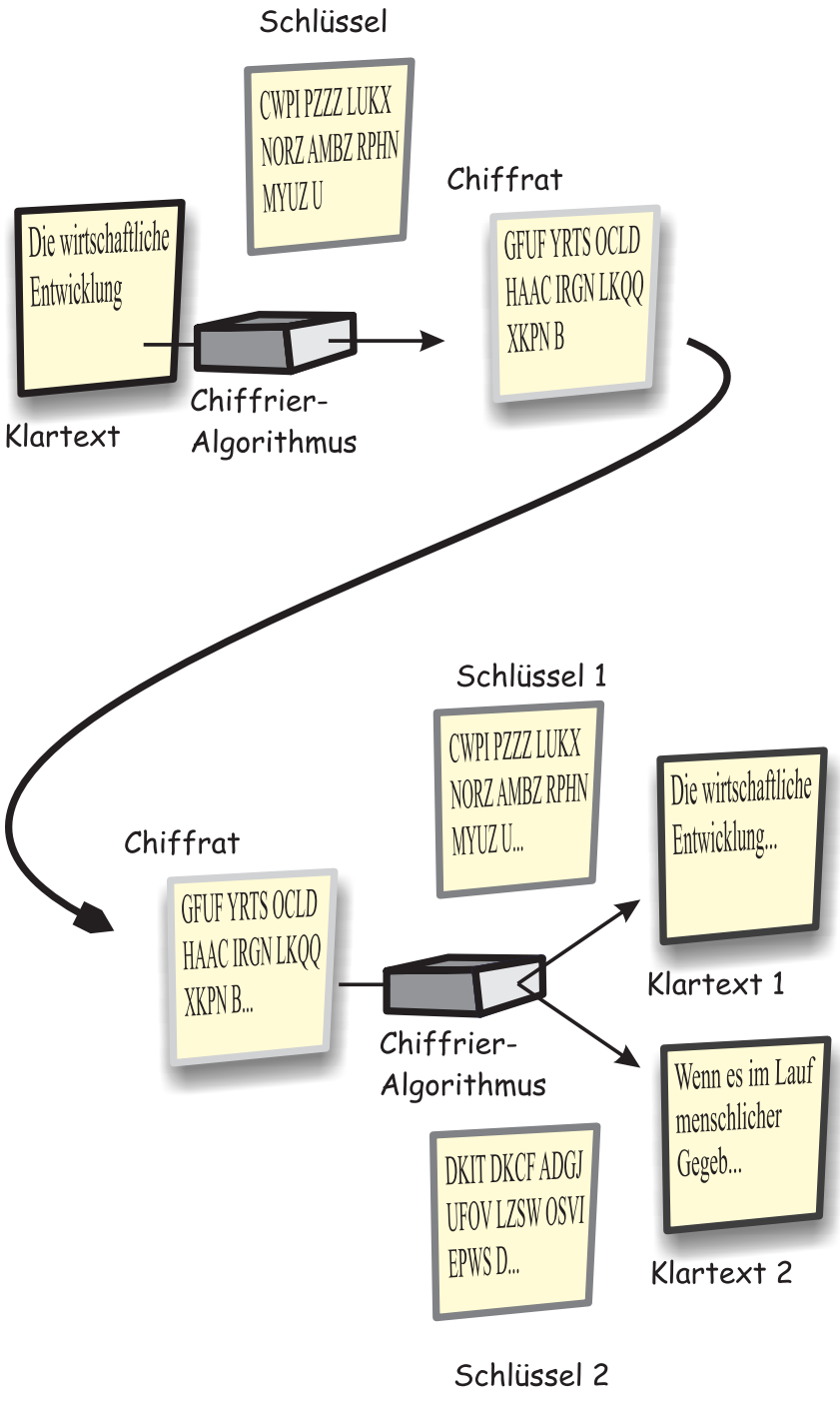


Abbildung 6.6: One-Time-Pad mit zwei möglichen Entschlüsselungen

Nachteil des One-Time-Pad ist, dass man immer einen neuen Schlüssel verwenden muss, der zudem noch die Länge der zu chiffrierenden Daten haben muss. Dies ist für Datenübertragungen im Internet fast nicht praktikabel. Demzufolge spielt das One-Time-Pad im täglichen weltweiten Datenaustausch auch keine Rolle.

Schüsselaustausch

Bei der symmetrischen Verschlüsselung ist ein Problem sehr lästig, das schon die Kryptografen in der Antike beschäftigte: der Schlüsselaustausch. Für den General im Feld bedeutete dies, dass er den Schlüssel von seinem Feldherrn vor Abmarsch direkt bekam. Jede andere Möglichkeit der Übertragung war zu riskant. Boten konnten bestochen oder ermordet, Nachrichten gefälscht werden.

Der heutige Homebanking-Anwender müsste für eine Transaktion zu seiner Bank gehen, um einen Schlüssel auszutauschen, bevor er online einen Überweisung initiieren kann. Würde er dies nicht tun, könnte jemand anderes sich für seine Bank ausgeben und wichtige Kundendaten erhalten.

Für den Internetsurfer bedeutet dies, dass er bei einer Online-Auktion zum Auktionshaus fahren muss, um dort einen gemeinsamen Schlüssel auszuhandeln. Andernfalls kann er seine Kreditkartendaten nicht übertragen, ohne Gefahr zu laufen, dass jemand diese sensiblen Daten abfängt und für seine Zwecke missbraucht.

Das grundsätzliche Problem bei der Verschlüsselung ist der Austausch eines geheimen Schlüssels. Ohne den gemeinsamen Schlüssel kann keine vertrauliche Kommunikation stattfinden. Sind beide Seiten räumlich voneinander getrennt, besteht zunächst keine Verbindung, die diesen Austausch ermöglicht: Telefon kann abgehört, Briefverkehr kann abgefangen werden. Der einzige Weg ist das physikalische Zusammenkommen an einem abhörsicheren Ort. Dies ist bei Datenübertragungen im Internet besonders lästig. Wohl niemand hat mit allen Kommunikationspartnern im Internet jeweils einen Schlüssel ausgetauscht.

Das Problem des Schlüsselaustauschs werden wir in einem späteren Abschnitt näher beleuchten.

Algorithmen der symmetrischen Chiffrierung

Die gebräuchlichsten symmetrischen Algorithmen sind:

- DES

  Der Data Encryption Standard ist gegenwärtig sicher die am häufigsten benutzte Architektur. Er wurde Mitte der 70er Jahre von der amerikanischen Regierung unter Mithilfe der National Security Agency eingeführt.

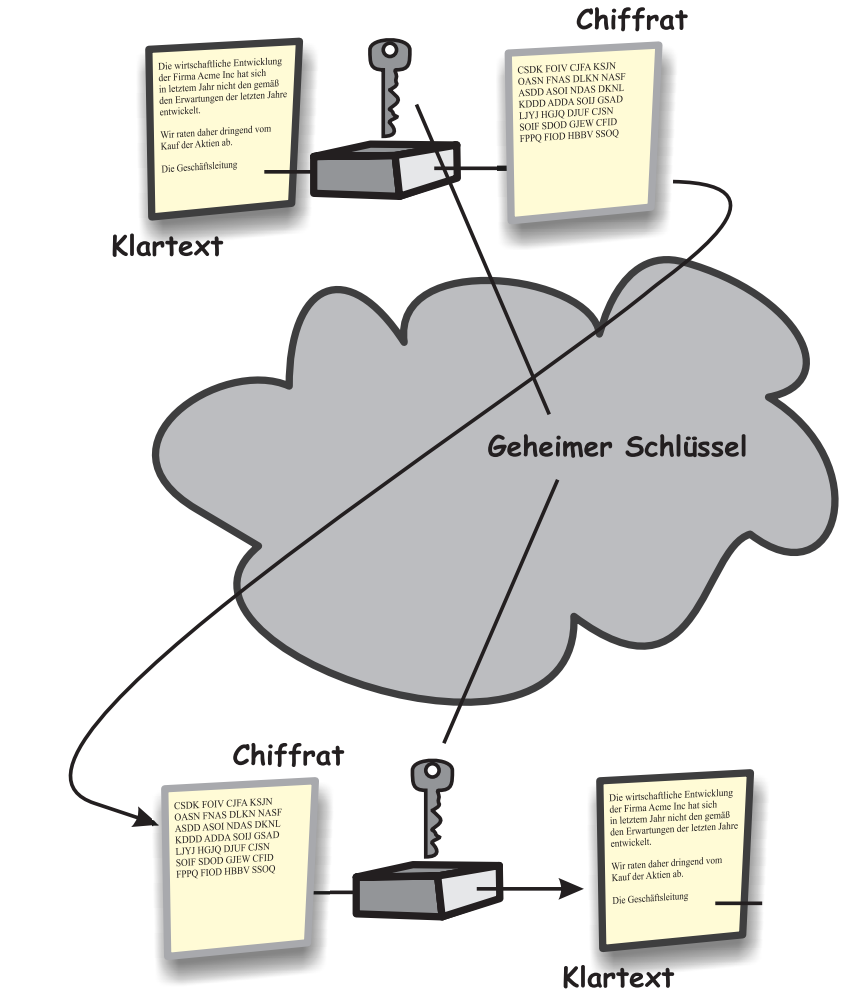


Abbildung 6.7: Symmetrische Verschlüsselung

- Triple-DES

  Triple-DES ist eine Erweiterung des DES-Standards und arbeitet nach dem gleichen Verfahren wie DES, wobei die Chiffrierung unter besonderen Bedingungen den Algorithmus dreimal durchläuft.

  DES ist eine Blockchiffrierung. Die Klartextdaten werden zu Blöcken von 64 Bit Chiffrat verschlüsselt. Die Schlüssellänge beträgt 56 Bit, der Algorithmus ist eine Mischung aus partieller Transposition und Substitution. Die Struktur der Verschlüsselung unterstützt die Implementierung des Algorithmus in Hardware.

DES gilt heute als gebrochen, wird aber dennoch in vielen Applikationen weiterverwendet. Weitere Informationen sind unter [26] ausführlich dokumentiert.

- CAST

  Dieses Verschlüsselungsverfahren wurde von Carlisle Adams und Stafford Tavares entwickelt [26]. Es verwendet einen 64-Bit-Schlüssel und wurde bisher von keinem Verfahren der Kryptoanalyse geknackt.

- IDEA

  Der International Data Encryption Algorithm wurde 1990 unter dem Namen PES (Proposed Encryption Standard) vorgestellt, weiterentwickelt zum IPES (Improved Proposed Encryption Standard) und 1992 in seine endgültige Form IDEA gebracht [27]. Er wurde von Xuejia Lai und James Massey von der Eidgenössischen Technischen Hochschule in Zürich entwickelt. Der Algorithmus basiert auf einer Blockchiffrierung von 64 Bit Klartext, die in je vier 16 Bit langen Blöcken verarbeitet werden. Es wird ein Schlüssel der Länge 128 Bit verwendet. Die Verschlüsselungsmechanik verwendet Transpositionen und Substitutionen.

  IDEA gilt als sehr starker Mechanismus und ist bis heute noch nicht gebrochen. IDEA unterliegt dem Patentschutz und wird derzeit in PGP (»Pretty Good Privacy«, siehe Abschnitt 6.9.1) als Alternative für DES verwendet.

  Weitere Informationen zur Funktionsweise und Architektur von IDEA sind unter [26] ausführlich dokumentiert.

Es existieren eine Reihe von weiteren symmetrischen Verschlüsselungsverfahren, die hier nicht mehr diskutiert werden sollen. Interessierte Leser mögen sich in der entsprechenden Fachliteratur weiter in diese Methoden der Kryptografie vertiefen.

6.4.4 Asymmetrische Verschlüsselung

Man hat lange nach einem Verfahren gesucht, das die Verbindungsaufnahme über ein öffentliches Kommunikationsmedium ermöglicht, ohne dass man physikalisch zusammenkommen muss, um einen geheimen Schlüssel auszutauschen. Dieses Problem könnte man lösen, indem man mit dem einen Schlüssel die Daten chiffriert und mit dem anderen dechiffriert. Mit diesem Ansatz könnte man den öffentlichen Schlüssel publizieren und jeder, der etwas Geheimes mitzuteilen hätte, würde diesen Schlüssel verwenden. Die Nachrichtenübertragung wäre sicher, da die einzige Möglichkeit zur Dechiffrierung der Information nur derjenige hat, der den privaten Schüssel besitzt.

Historische Entwicklung

Die Suche nach einem solchen Schlüsselpaar war zum ersten Mal 1973 von Erfolg gekrönt, als beim britischen Government Communications Headquarter (GCHQ) James Ellis die ersten Lösungen zu diesem Problem entwickelte. Aufgrund der Einstufung dieser Entdeckungen als geheim erfuhren die wirklichen »Erfinder« der Public-Key-Verfahren erst in neuerer Zeit eine angemessene Würdigung.

Ein von Ellis unabhängiges Public-Key-Verfahren wurde von Diffie und Hellman 1974 mit dem Diffie-Hellman-Verfahren erarbeitet. Ein insbesondere für den Anwender vereinfachtes Verfahren wurde von Rivest, Shamir und Adleman 1977 mit RSA vorgestellt (die Abkürzung steht für die Entwickler des Verfahrens). Damit war der Grundstein für einen sicheren Datenaustausch über ein öffentliches Kommunikationsmittel gelegt. Es dauerte aber bis Ende der achtziger Jahre, bis die Verschlüsselungsthematik ein wirkliches Anliegen der Öffentlichkeit wurde, da etwa zeitgleich das Internet zunehmend die Medien eroberte.

Die Verschlüsselung wird von vielen, die darauf angewiesen sind, über ein bestehendes öffentliches Medium private oder geheime Daten zu übermitteln, als ein Segen betrachtet. Das Problem ist dabei aber, dass nicht nur Datenübertragungen vor dem Zugriff Dritter geschützt sind, sondern auch Kommunikationskanäle illegaler Organisationen nicht mehr überwacht werden können. Diese Bedenken teilen insbesondere Staaten und Regierungen, die Terroranschläge befürchten. Deshalb sind Organisationen wie die National Security Agency (NSA) der USA sehr skeptisch gegenüber einer öffentlichen Einführung solcher Technologien. Deren Argumente lauten, dass sich internationale Kommunikationskanäle entwickeln, die man nicht mehr kontrollieren und insbesondere nicht mehr abhören kann. Die eigentliche Argumentation über Verletzungen des Patentrechts sind hier nur der Vorwand, um eine Kontrolle über diese Architektur zu bewahren. Und trotzdem hat sich die Einführung von starker Verschlüsselung auf der Basis von Public-Key-Algorithmen als wesentlicher Faktor bei der Entwicklung des internetbasierten Handels herausgestellt, gibt sie doch dem einzelnen Unternehmen ein Mittel in die Hand, Online-Business zu betreiben, das sowohl für den Kunden als auch für den Anbieter sicher und privat ist.

Zur ersten wirklichen Konfrontation der Öffentlichkeit mit den Kritikern der Verschlüsselung kam es mit der Verbreitung einer Software, die von Phil Zimmermann entwickelt wurde und sich unter dem Namen »Pretty Good Privacy« einen Namen machte. Das Verfahren sichert einen Anwender vor dem unberechtigten Zugriff Dritter auf die damit verschlüsselten Daten: In seinen ersten Versionen verwendete es die Architektur von RSA und wurde mit der Möglichkeit der Verschlüsselung mit einer Schlüssellänge von bis zu 1024 Bit ausgeliefert. Und diese Software konnte als Freeware von FTP-Servern im Internet geladen werden.

Da die amerikanische National Security Agency (NSA) eine solch starke Verschlüsselung nicht mehr kontrollieren konnte, wollte sie den Vertrieb dieser

Software reglementieren und die Verbreitung insbesondere in Länder des sozialistischen Machtbereichs unterbinden. Der rechtliche Hebel war die Verletzung des Patentschutzrechts mit diesem Produkt in Verbindung mit der RSA-Architektur.

Doch gerade diese Querelen rückten das Interesse an starker Verschlüsselung und Privatsphäre im Internet immer mehr ins Licht der Öffentlichkeit und man erkannte einen entsprechenden Bedarf insbesondere bei Geschäftsprozessen und Zahlungsverkehr. Aber auch viele Privatpersonen legten immer größeren Wert auf eine geschützte Kommunikation, die den Kompromissvorschlag der NSA, eine Chiffrierung mit kürzerer Schlüssellänge, nicht mehr akzeptierte.

Heute sind die Verschlüsselungsvorschriften in vielen Ländern weitgehend gelockert und man kann mit starker Chiffrierung seine Daten für die Übertragung im Internet nutzen, ohne rechtliche Schritten befürchten zu müssen.

Protokoll der Architektur

Während die symmetrische Verschlüsselung darauf basiert, einen einzigen, geheimen Schlüssel für Chiffrierung als auch Dechiffrierung zu verwenden, verwendet die asymmetrische Architektur ein Schlüsselpaar, das im Besitz einer Person oder eines Service ist. Bei dem Paar handelt es sich um einen öffentlichen und einen privaten Schlüssel (Public Key und Private Key).

Das Schlüsselpaar wird einmal erzeugt und für einen längeren Zeitraum benutzt. Die Funktionen, die die Keys erzeugen, sowie der Ver-/Entschlüsselungsmechanismus beruht auf einem öffentlichen Verfahren. (Anmerkung: Die mathematischen Algorithmen für die angesprochenen Verfahren sind veröffentlicht und mit ein wenig mathematischem Verständnis ist es relativ einfach, die zugrunde liegenden Funktionen nachzuvollziehen. Es ist aber schwer herauszufinden, ob das Produkt, das in einer kompilierten Version vorliegt, auch wirklich tut, was es verspricht. Selbst wenn eine Version eines BAD_PGP-Programms ein mathematisch korrektes Verschlüsselungsverfahren implementiert, ist man nicht dagegen gefeit, dass das gleiche Programm den geheimen Private Key an den Hersteller mailt.)

Für die Erzeugung des Schlüsselpaars wird eine Zufallszahl generiert, die als Eingabewert für den Schlüsselgenerator dient, der seinerseits zwei Schlüssel ausliefert. Die Zufallszahl soll so groß sein, dass weltweit die Wahrscheinlichkeit gegen Null geht, dass eine zweite Person das gleiche Schlüsselpaar erzeugt. Anschließend wird der private Schlüssel versteckt, während der öffentliche Schlüssel jedem zugänglich gemacht wird.

Wie sieht nun das Protokoll des Verbindungsaufbaus und der verschlüsselten Datenübertragung aus? Joe User ist in seinem Büro in Australien und besitzt ein Schlüsselpaar, dessen öffentlichen Part er jedermann zugänglich gemacht hat. Dies kann über eine Webseite oder einen Directory Server geschehen, auf dem jedermann nachschauen kann. Thomas Turbo, der im europäischen Büro sitzt, möchte Joe vertrauliche Informationen über das Internet übermitteln.

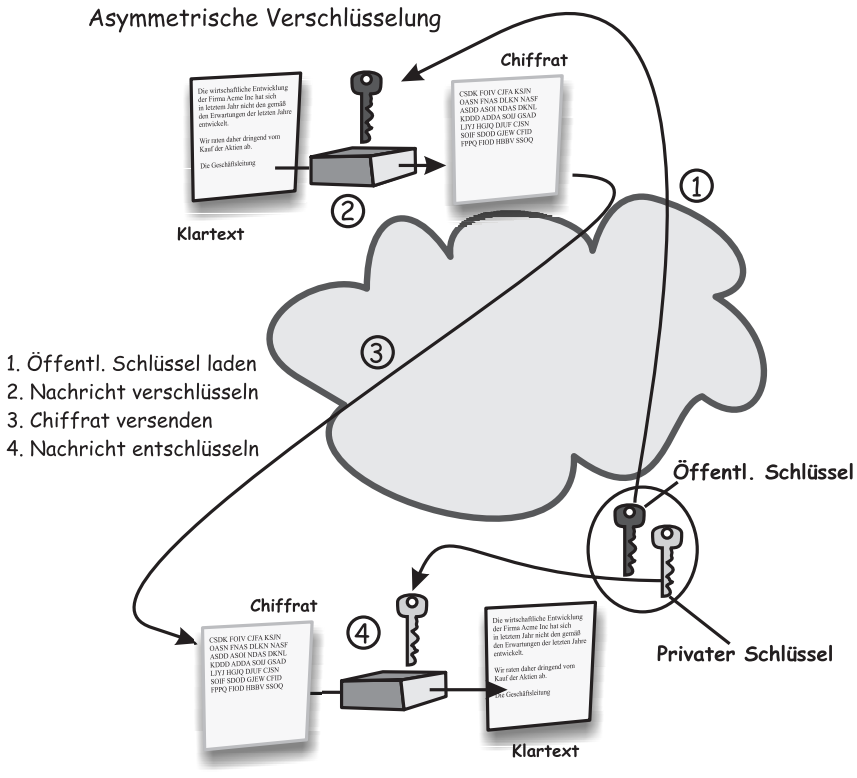


Abbildung 6.8: Asymmetrische Verschlüsselung

Im ersten Schritt verschafft sich Thomas den öffentlichen Schlüssel, den Joe auf seiner Webseite publiziert hat. Im zweiten Schritt verschlüsselt er die Daten mit diesem Schlüssel, anschließend überträgt er über das Internet die Daten an Joe. Niemand, der auf dem Weg zwischen Europa und Australien an der Leitung lauscht und die Daten mitprotokolliert, kann die übertragenen Informationen entschlüsseln (wir setzen hier einen Algorithmus voraus, der noch nicht gebrochen wurde). Die Daten erreichen Joe in Australien und da er als einziger den privaten Schlüssel besitzt, kann nur er alleine die Informationen wiederherstellen. In Abbildung 6.8 ist der Vorgang noch einmal illustriert.

Mit diesem System hat man erstmals das Problem des Schlüsselaustauschs gelöst. Es ist nicht mehr notwendig, dass zwei Kommunikationspartner zusammenkommen müssen, um einen geheimen Schlüssel auszutauschen. Der Partner, an den Informationen übermittelt werden sollen, muss im Besitz eines Schlüsselpaars sein, damit er Daten empfangen kann.

Informationen, die mit einem bestimmten öffentlichen Schlüssel chiffriert wurden, können demnach nur von einer bestimmten Person entschlüsselt werden, nämlich derjenigen, die den dazugehörigen privaten Schlüssel besitzt. Architekturen wie RSA lassen sich aber auch umgekehrt anwenden: Alles, was mit dem Private Key einer Person chiffriert wurde, kann mittels

des dazugehörigen öffentlichen Schlüssels wieder entschlüsselt werden: Joe User kann mit seinem privaten Schlüssel ein Dokument verschlüsseln, das jeder wieder entschlüsseln kann. Dem Sinn eines solchen Vorgehens widmet sich der übernächste Abschnitt.

6.4.5 Hybridverschlüsselung

Mit dem Public-Key-Verfahren sind wir nun in der Lage, eine Information an eine Person zu übermitteln, die einen Public und den dazu gehörenden Private Key hat. Der Nachteil des Verfahrens ist, dass es sehr rechenaufwändig ist und sich nicht für das Versenden von großen Datenmengen eignet.

Trotzdem haben wir mit diesem Verfahren einen entscheidenden Vorteil gewonnen: Wir können nun ein Geheimnis auf einem sicheren Weg übermitteln. Und warum sollte ein solches Geheimnis nicht ein geheimer Schlüssel sein.

Angenommen, Thomas Turbo möchte nun einen bidirektionalen Datenkanal aufbauen und mit Joe User in Australien kommunizieren. Bei der symmetrischen Verschlüsselung war dies nur möglich, wenn beide vorher zusammengekommen und einen Schlüssel ausgetauscht hatten. Nun ist folgendes Vorgehen möglich:

1. Thomas Turbo baut eine unverschlüsselte Verbindung mit Joe User auf und lädt sich dessen Public Key.
2. Er generiert eine Zufallszahl (der spätere geheime Schlüssel) und chiffriert diese mit dem öffentlichen Schlüssel von Joe User.
3. Er übersendet den chiffrierten Schlüssel an Joe User zusammen mit der Nachricht, dass er IDEA verwenden möchte.
4. Joe User entschlüsselt mit seinem Private Key das Chiffrat und erhält den geheimen Schlüssel.
5. Joe User schreibt eine Nachricht und verschlüsselt diese mit dem geheimen Schlüssel mit Hilfe des IDEA-Algorithmus.
6. Joe User versendet die Nachricht an Thomas Turbo.

Über asymmetrische Verschlüsselung wurde auf einem sicheren Weg ein geheimer Schlüssel ausgetauscht. Die weitere Kommunikation erfolgt anschließend über symmetrische Verschlüsselung. Diese Form wird als Hybridverschlüsselung bezeichnet, weil sie sowohl die asymmetrische als auch die symmetrische Architektur verwendet. Diese Art der Verschlüsselung ist je nach Implementierung etwa hundertmal schneller als die rein asymmetrische.

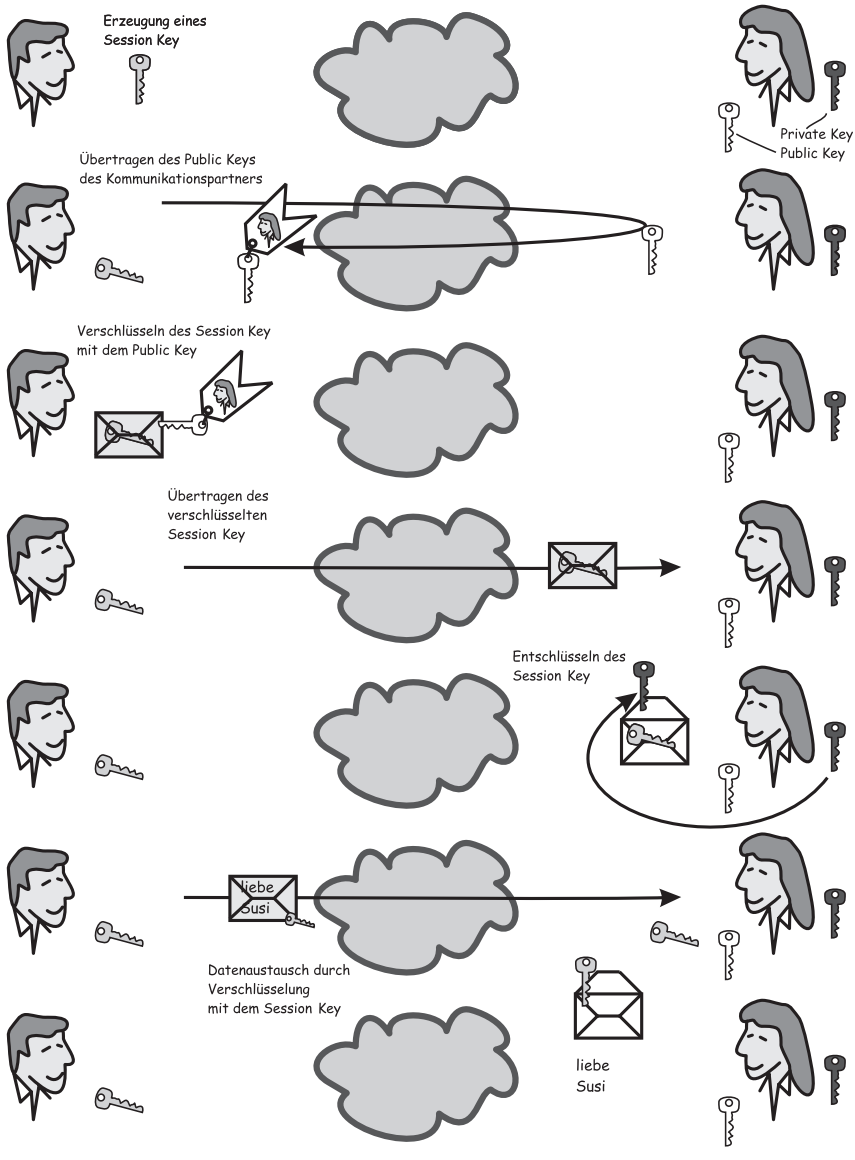


Abbildung 6.9: Hybridverschlüsselung

Asymmetrische Verschlüsselung/Public-Key-Verfahren

Vorteil: sicherer Verbindungsaufbau zwischen zwei Partnern, die vorher noch keinen gemeinsamen geheimen Schlüssel ausgetauscht haben.

Nachteil: mathematisch aufwändig und langsam.

Symmetrische Verschlüsselung/Private-Key-Verfahren

Vorteil: schnell, einfach zu implementieren.

Nachteil: Partner, die sicher kommunizieren möchten, müssen vor der Kommunikation physikalisch zusammenkommen, um den geheimen Schlüssel auszutauschen.

Hybridverschlüsselung

Nutzt die Vorteile beider Verfahren, indem es für den Austausch des geheimen Schlüssels die asymmetrische, für den Austausch der Nutzdaten die symmetrische Chiffrierung verwendet.

6.5 Digitale Unterschrift

Seit 1999 ist die Digitale Unterschrift vom deutschen Gesetzgeber als gerichtsverwertend anerkannt. Bedingung hierbei ist eine starke Verschlüsselung basierend auf der Verwendung einer anerkannten Public Key Infrastructure.

Bevor wir uns der rechtsverbindlichen Unterschrift über ein PKI-basiertes System (PKI – Public Key Infrastructure) widmen, müssen wir noch eine Technik betrachten, die hierfür notwendig ist: den Hash-Algorithmus.

6.5.1 Hash-Algorithmen

Ein Hash-Algorithmus ist eine Einweg-Funktion. Dies bedeutet, dass man zwar immer in eine Richtung die Daten berechnen kann, der umgekehrte Weg ist nicht möglich. Das Ergebnis einer solchen Funktion wird als »Hash« bezeichnet.

Betrachten wir hier ein simples Beispiel:

Die Zahl 157 ist als Binärzahl darstellbar als »1001 1101«. Eine mögliche »Hash-Funktion« könnte folgendermaßen aussehen:

Die Summierung über alle »1« mit anschließender Subtraktion der Summe aller »0« führt hier zur Zahl 2. Alleine aus der Kenntnis der Zahl 2 kann umgekehrt niemand auf die Zahl 157 schließen. Dies ist die Intention einer Hash-Funktion: Sie lässt die Berechnung der Charakteristik einer Information zu, aus der Kenntnis des Ergebnisses alleine lässt sich keine Aussage über die Ausgangsdaten machen.

Das vorgestellte Beispiel zeigt aber eine entscheidende Schwäche auf, die eine wirkliche Hash-Funktion nicht hat: Es existieren berechenbare Kollisionen. Für obiges Beispiel: Die Zahl 158 (1001 1110) hat die gleiche Charakteris-

tik wie die Zahl 157. Zwei (oder mehr) unterschiedliche Daten mit dem gleiche Hash-Wert werden als Kollision bezeichnet.

Eine starke Hash-Funktion sieht folgendermaßen aus:

1. Sie nimmt ein Datum an und liefert ein Ergebnis.
2. Sie lässt aus dem Ergebnis heraus keinen Rückschluss auf das Ausgangsdatum zu.
3. Sie ist eindeutig: Ein Ausgangsdatum liefert immer eindeutig das gleiche Ergebnis.
4. Das Ergebnis hat immer die gleiche Länge, unabhängig von der Länge der Ausgangsinformation.
5. Sie ist kollisionsfrei, das heißt, sie liefert zu zwei verschiedenen Ausgangsdaten keine identischen Ergebnisse.
6. Ihre zugrunde liegende Architektur ist öffentlich einsehbar.

Das heißt, jeder Text, jede Information hat eine hash-basierte Charakteristik, die eindeutig diesem Text zuzuschreiben ist.

Eine Analogie hierzu findet sich leicht im täglichen Leben: Der Fingerabdruck eines Menschen lässt zwar nicht auf dessen Hautfarbe, Größe oder Persönlichkeit schließen, ist aber eindeutig der betreffenden Person zuzuordnen und für diese leicht zu erzeugen. Deshalb spricht man im Datenumfeld auch vom elektronischen Fingerabdruck einer Information.

Hash-Funktionen arbeiten meist nach dem folgenden Prinzip:

Der Datensatz, dessen Hash gebildet werden soll, wird in Pakete aufgeteilt. Das erste Paket erhält durch den Hash-Algorithmus ein Ergebnis. Dieses wird – eventuell gekürzt, indem nur ein Teil des Ergebnisses verwertet wird – zusammen mit dem folgenden Paket des Ausgangsdatums verknüpft und wieder in den Algorithmus gesteckt. Diese Prozedur wird so lange durchlaufen, bis alle Pakete des Ausgangsdatums abgearbeitet sind. Das Endergebnis liefert den eigentlichen Hash-Wert.

Bekannte Hash-Algorithmen sind crypt, MD4, MD5 und SHA.

MD4: Message Digest in der Version 4 (MD4) ist eine Einweg-Hash-Funktion, die von Ron Rivest entwickelt wurde [31]. Sie erzeugt einen 128 Bit langen Hash-Wert. Dieses Verfahren wurde bereits kurz nach der Einführung erfolgreich angegriffen und gilt nicht mehr als sicheres Hash-Verfahren.

MD5: Die Weiterentwicklung von MD4 wurde von Rivest in MD5 umgesetzt [32]. Dieses Verfahren generiert einen 128 Bit langen Hash-Wert. MD5 ist erfolgreich angegriffen worden und gilt nicht mehr als sicheres Hash-Verfahren.

SHA: Secure Hash Algorithm. Dieses Hash-Verfahren wurde unter maßgeblicher Beteiligung der National Security Agency (NSA) und des National Institute for Standards and Technology (NIST) entwickelt [33]. Dieses Hash-Verfahren ist dem des MD4 sehr ähnlich. Es erzeugt Hash-Werte von 160 Bit Länge.

Crypt: Dieser Hash-Algorithmus wurde häufig zum Hinterlegen von Passwörtern insbesondere im Unix-Umfeld verwendet. Es ist dabei nicht notwendig, die Passwörter der Benutzer in Klartext oder verschlüsselt auf dem Betriebssystem zu hinterlegen, es genügt der Hash. Möchte sich ein Benutzer authentifizieren, so gibt er sein Passwort gegenüber dem Login-Mechanismus an, der seinerseits diese Daten durch den Hash-Algorithmus laufen lässt und das Ergebnis mit seinen eigenen Daten vergleicht.

Die Passwort-Daten stehen in Unix-Systemen normalerweise im Verzeichnis /etc in der Datei passwd.

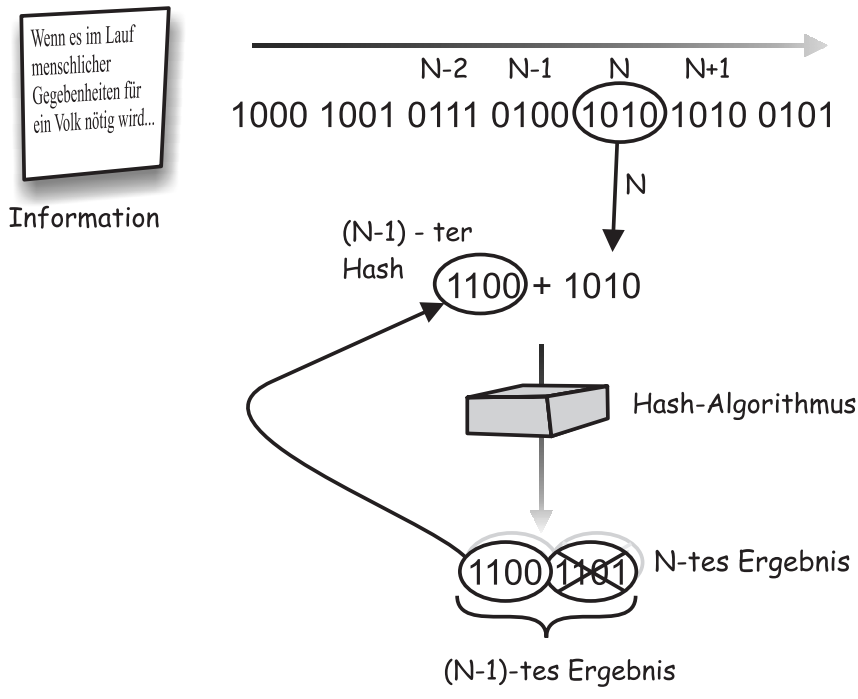


Abbildung 6.10: Beispiel für die Funktionsweise eines Hash-Algorithmus

Auch wenn der Algorithmus einer Hash-Funktion als sicher gelten kann, so ist diese Art der Authentifizierung unsicher. Angreifer machen sich hier die Bequemlichkeit der Benutzer zunutze und verwenden die öffentlich zugänglichen Hash-Funktionen, um sich über eine Wörterbuchattacke Zugang zu den Passwörtern der Anwender zu verschaffen. Der Angriff ist simpel: Man geht davon aus, dass viele Benutzer ein Passwort aus dem Wörterbuch wählen. Nimmt man nun eine Liste aller deutschen und englischen Wörter als Ausgangsdaten für die Hash-Funktion und vergleicht das Ergebnis mit der Liste aller Passwörter, so kann man unter Umständen recht schnell fündig werden. Beispiel: Der Benutzer hat das Passwort »Sonne«. Dieses Wort wird über kurz oder lang bei der Wörterbuchattacke als Ausgangsdatum für die Hash-Funktion erscheinen. Der Hash wird dann mit jedem Eintrag in der Liste der Passwörter aller Anwender verglichen. Da, wo sich eine Übereinstimmung ergibt, ist das Passwort des Benutzers enttarnt.

Es ist festzuhalten, dass damit die Funktion selbst nicht gebrochen wurde, es wurde lediglich eine Brute-Force-Attacke auf die Passwörter unternommen.

6.5.2 Digitale Signatur

Architektur

Wenn Sie im täglichen Leben eine Unterschrift unter ein Dokument setzen, erklären Sie dadurch, dass Sie die darin enthaltenen Informationen in dieser Weise anerkennen. Als Beispiel soll hier ein Testament dienen. Ein Großvater, nennen wir ihn Theo, möchte seinen letzten Willen dokumentieren und verfasst ihn auf einer Schreibmaschine. Zur Dokumentierung der Absichtserklärung unterschreibt er jedes Blatt des Testaments und hinterlegt es an einem leicht auffindbaren Platz.

In der digitalen Welt kann jedes Dokument geschrieben und verändert werden. Theo würde auf seinem Computer einen letzten Willen dokumentieren und auf der Festplatte speichern. Eine seiner Töchter, die mit diesem Abkommen nicht einverstanden ist, wäre bei Auffinden des Dokuments in der Lage, ohne Spuren die Daten des letzten Willens zu verändern, bevor das Testament in Kraft tritt. Scheinbar müsste sich der Großvater darauf verlassen, dass er seinen Töchtern eine hinreichende Moral anerzogen hat, damit keine von ihnen auf die Idee kommt, ein solches Dokument ohne sein Einverständnis zu ändern.

Aber er hat eine Alternative, die im Folgenden beschrieben wird.

1. Theo generiert ein Schlüsselpaar mit einem anerkannten Verfahren aus der PKI. Der eine Schlüssel ist der Public Key, der andere der Private Key.
2. Er veröffentlicht den Public Key, das heißt, jeder, insbesondere seine beiden Töchter, bekommt eine Kopie des Public Key.
3. Als Nächstes schreibt Theo seinen letzten Willen auf seinem Computer nieder.
4. Nachdem er fertig ist, erzeugt Theo einen Hash des Dokuments. Hierzu verwendet er einen allgemein anerkannten Hash-Algorithmus wie zum Beispiel SHA.
5. Den Hash des Dokuments verschlüsselt er nun mit seinem Private Key und hängt die so entstandenen Daten an das Testament als Unterschrift an.

Für den bedauerlichen Fall, dass Theo nun den Weg alles Irdischen geht, tritt das Testament in Kraft. Wie aber konnte er sicher sein, dass keine seiner Töchter das Testament manipuliert?

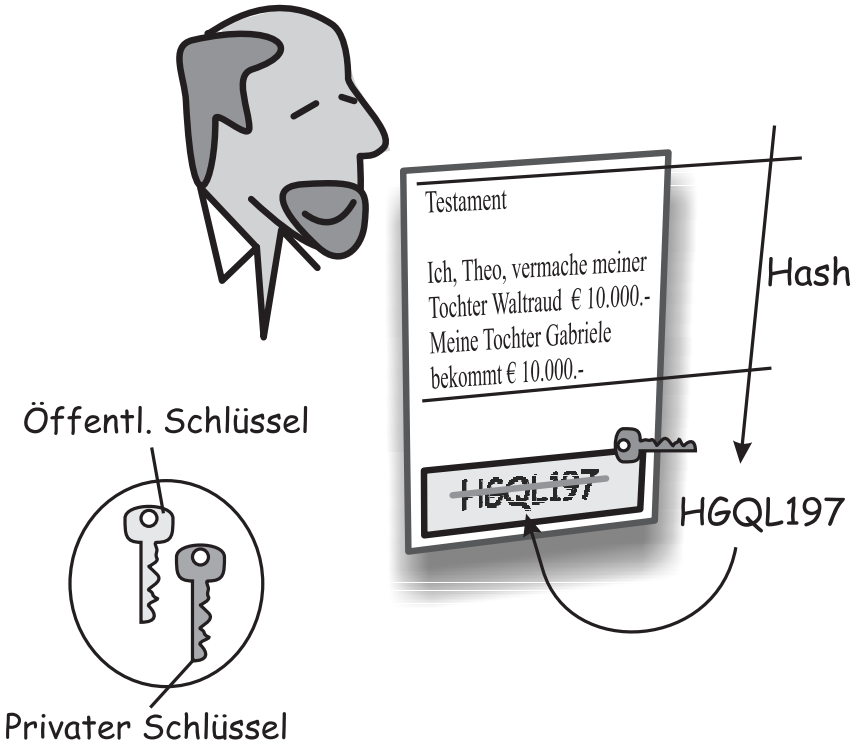


Abbildung 6.11: Erzeugen einer digitalen Unterschrift

Er verlässt sich hier auf die Kryptografie. In dem Dokument steht, dass beide Töchter je eine Summe von €10.000 erben. Angenommen, Tochter Waltraud findet das Dokument vor der Veröffentlichung und sie möchte ihren Anteil auf €15.000 verändern, so dass ihre Schwester Gabriele nur noch €5.000 erhält. Sie modifiziert die Daten auf dem Computer und gibt das Dokument bekannt. Leider hat Waltraud übersehen, dass dem Dokument eine digitale Unterschrift angefügt ist.

Gabriele, die das Testament auf Glaubwürdigkeit überprüfen möchte, geht wie folgt vor:

1. Sie nimmt das Dokument und generiert davon einen SHA-Hash.
2. Sie nimmt den Public Key von Theo und entschlüsselt die Unterschrift des Dokuments (der verschlüsselte Hash des Großvaters wird dechiffriert).
3. Sie vergleicht den Hash, den seinerzeit der bereits verstorbene Theo generiert hat, mit dem, den sie gerade erzeugt hat.
4. Sie stellt in diesem Fall keine Übereinstimmung fest und schließt daraus, dass das Dokument manipuliert wurde.

Betrachten wir die Leistung der Unterschrift und das nachträgliche Überprüfen etwas genauer. Das Dokument wurde vom Urheber generiert. In die-

ser Form wäre jede Manipulation am Dokument möglich, weshalb eine Unterschrift notwendig ist. Wir bezeichnen das (nicht unterschriebene Klartext-)Dokument mit »t«.

Theo hat ein Schlüsselpaar K_{pub} und K_{priv}, von dem er den öffentlichen Schlüssel (K_{pub}) an jede seiner Töchter verteilt hat. Beide haben damit je eine identische Kopie des originalen Public Key von Theo, beiden wurden diese Daten persönlich von ihm übergeben. Mit der Veröffentlichung des Public Key werden zwei Intentionen umgesetzt:

- Der Eigentümer des Public Key wird mit seiner Person in Verbindung gebracht.
- Der Public Key ist mit dem entsprechenden Private Key assoziiert, das heißt, derjenige, der den Public Key besitzt, hat auch den entsprechenden Private Key. Wer sich einen Public Key »aneignet«, hat nicht den zugehörigen Private Key. Den hat nur der wirkliche Eigentümer.

Theo generiert einen Hash $H_{SHA}(t)$ des Dokuments. Dies ist nicht zwingend notwendig. Theo könnte das eigentliche Testament mit seinem Private Key verschlüsseln und das Resultat an das ursprüngliche Dokument anhängen. Der Vorteil des Hash ist, dass er kurz ist und immer die gleiche Länge hat, egal ob das Dokument ein paar Zeilen lang ist oder es sich um eine mehrere tausend Seiten lange Abhandlung handelt.

Der Hash kann von jedem erzeugt werden, da es sich um ein öffentliches Verfahren handelt, dessen Algorithmus dokumentiert ist. Das Ergebnis der Hash-Funktion, angewandt auf das gleiche Dokument, ist immer dasselbe, unabhängig davon, wer dies durchführt. Im vorliegenden Beispiel ist der Hash-Wert, den Theo generiert, der gleiche wie der, den Gabriele erhalten würde, wäre das Testament nicht modifiziert.

Im nächsten Schritt hat der Großvater den Hash mit seinem Private Key verschlüsselt. Dies kann nur er tun, da nur er diesen privaten Schlüssel hat. Dies ist – wie bei einer handschriftlichen Unterschrift – das Alleinstellungsmerkmal, das die Originalität der Signatur sichert. Damit ist die digitale Unterschrift U für dieses Dokument geleistet. Formal geschrieben lautet sie:

$U(K_{priv,} t) = E_{K_{priv}} (H_{SHA}(t))$

wobei

- $U(K_{priv,} t)$ für »Digitale Unterschrift mit dem Private Key K_{priv} und dem Klartext t« steht,
- $E_{K_{priv}} (H_{SHA}(t))$ die Verschlüsselung mit dem privaten Schlüssel ist,
- $H_{SHA}(t)$ der Hash des Dokuments mit dem verwendeten Hash-Algorithmus SHA ist.

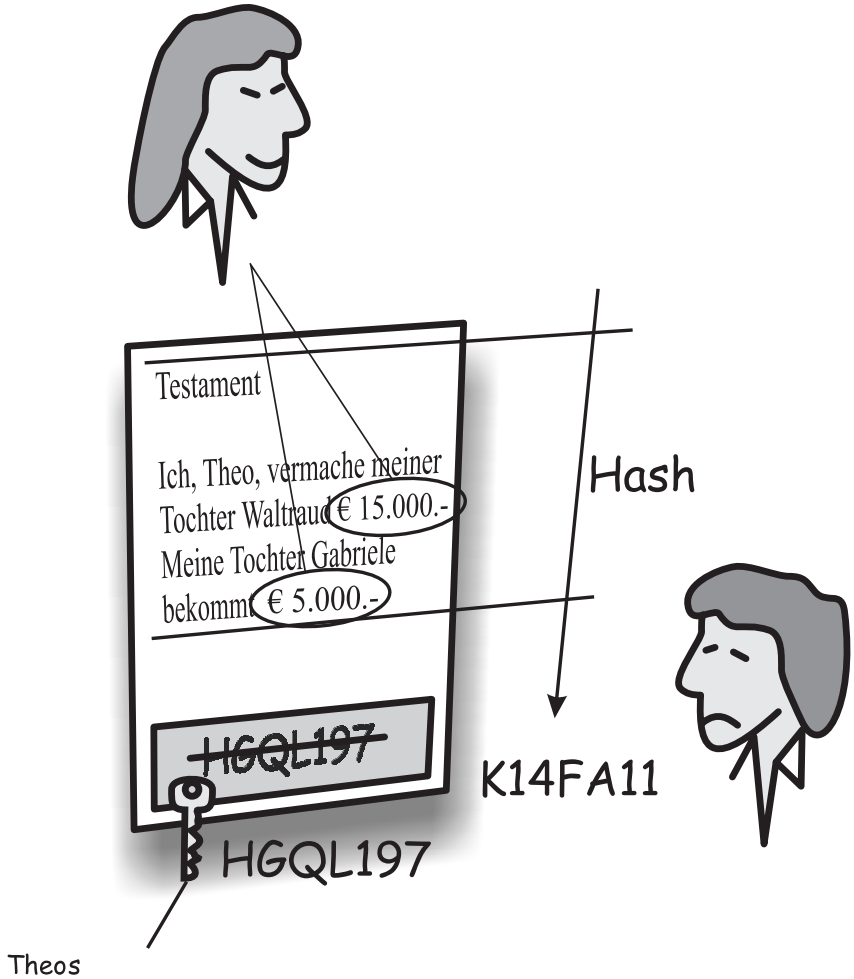


Abbildung 6.12: Verifikation eines Dokuments

Das komplette Dokument T ist nur mit der digitalen Unterschrift gültig:

$$T = t \mid U(K_{priv}, t)$$

Mit der digitalen Unterschrift ist die Fälschung des Dokuments T nicht mehr möglich.

Betrachten wir nun die Überprüfung der elektronisch unterschriebenen Daten. Das obige Beispiel hat gezeigt, dass das Dokument t nachträglich modifiziert wurde und wie der Betrug auffiel (wir bezeichnen in Folge das zu überprüfende Dokument mit t'). Eine Überprüfung der Unterschrift geschieht dadurch, dass zunächst ein Hash des Dokuments (des Gesamtdokuments T ohne die Signatur) erzeugt wird. Dieser Vorgang ist identisch mit der Erzeugung der Unterschrift:

$H_1 = H_{SHA}(t')$

Anschließend wird die digitale Unterschrift $U(K_{priv,} t)$ isoliert und entschlüsselt. Dies ist möglich, da jeder der Beteiligten einen anerkannten öffentlichen Schlüssel des Unterzeichners hat. Nochmals zur Wiederholung: Alles, was mit einem Private Key verschlüsselt wurde, kann mit einem dazu gehörenden Public Key entschlüsselt werden. Der entscheidende Punkt ist, dass man das Chiffrat zwar entschlüsseln kann, niemand aber in der Lage ist, das Chiffrat zu erzeugen. Dazu ist nur derjenige in der Lage, der im Besitz des Private Key ist. In unserem Beispiel ist dies Großvater Theo, der allerdings den Private Key mit ins Grab genommen hat.

Bei der Überprüfung der Originalität des Dokuments wird also der mit dem Private Key verschlüsselte Hash (also die digitale Unterschrift) mit dem öffentlichen Schlüssel dechiffriert:

$$H_2 = D_{K_{pub}}(U(K_{priv,} t)) = D_{K_{pub}}(E_{K_{priv}}(H_{SHA}(t))) = H_{SHA}(t)$$

Beide Werte, H_1 und H_2, werden anschließend miteinander verglichen. Sind sie identisch, dokumentiert die Unterschrift die Originalität der damit unterschriebenen Daten. Sind sie verschieden, wurde das Dokument manipuliert:

$$H_1 = H_{SHA}(t') = H_2 = H_{SHA}(t) \quad \Rightarrow t' = t$$

Die digitale Unterschrift ist möglich, wenn drei Voraussetzungen gegeben sind:

1. Der Unterzeichner hat ein Schlüsselpaar, das er sich mit einem anerkannten öffentlichen Algorithmus erzeugt hat und dessen Architektur nachweislich noch nicht gebrochen wurde. Mit dem Private Key wird die Unterschrift generiert, mit dem Public Key wird sie verifiziert.
2. Der öffentliche Schlüssel ist publiziert und die Zuordnung zwischen Public Key und dem Unterzeichner ist zweifelsfrei dokumentiert.
3. Die Schlüssellängen sind hinreichend lang, damit der Signier-Algorithmus einem Brute-Force-Angriff widerstehen kann.

Jede digitale Unterschrift ist individuell und nur in Verbindung mit dem damit unterschriebenen Dokument gültig. Sie verliert jeden Sinngehalt ohne das Dokument und kann somit nicht als Unterschrift für andere Dokumente verwendet werden.

Wird der Algorithmus, der der digitalen Unterschrift zugrunde liegt, gebrochen, sind alle damit geleisteten Unterschriften ungültig.

6.6 Digitale Zertifikate

Im täglichen Leben kann man sich gegenüber jeder anderen Person ausweisen. Dies geschieht mittels eines Dokuments, das von jedem als Original anerkannt wird und auf dem die Zuordnung zwischen den Daten des Doku-

ments und der Person, die es verwendet, zweifelsfrei anerkannt wird. Wie geschieht das im Einzelnen?

Derjenige, der ein solches Dokument vorlegt, zeigt an, dass er der Besitzer dieser Urkunde ist. Die Zuordnung zwischen den darauf enthaltenen Daten und der Person geschieht im Allgemeinen über das Lichtbild, das die Person zeigt, deren Daten dokumentiert sind. Würde ein anderer diesen Ausweis verwenden, wäre die Zuordnung zwischen dem Dokument und der Person, die ihn verwendet, nicht gegeben. Ein Ausweis kann damit nur von einer einzigen Person verwendet werden.

Wie aber kann man nun eine solche Ausweisprozedur im Internet einführen, in dem eine solche Verifikation nicht möglich ist? Wie kann man feststellen, wer am anderen Ende der Leitung sitzt? Ist es tatsächlich die Hausbank, mit der man gerade die Kontotransaktionen ausführt?

Ein Ausweis ist fälschungssicher, das heißt, es ist nur sehr schwer, einen Ausweis mit falschen Daten zu erstellen. Im Allgemeinen zweifelt man selten an der Echtheit eines solchen Dokuments.

Ein Internetausweis würde sich aus Datenbits zusammensetzen, die in dieser Form sehr einfach zu kopieren und zu fälschen wären. Wie aber stellt man die Eindeutigkeit der Zuordnung zu einer Person und die Integrität eines solchen Dokuments sicher? Eine Lösung bietet die Kryptografie.

Wir betrachten Architektur der digitalen Zertifizierung wieder anhand eines Beispiels. Joe User möchte über das Internet eine Musik-CD kaufen und wendet sich an einen Online-Shop namens »Internet-Music Online GmbH«. Diese Firma betreibt eine Webseite, die als Publikationsmedium dient, um die CDs anzupreisen und Bestellungen anzunehmen. Die URL dieser Seite lautet http://www.internet-music-online.de. Die Firma hat sich diesen Webserver im Haus installiert und betreibt ihn auch selbst.

Joe ist sich sicher, dass eine Firma »Internet-Music Online GmbH« existiert, da diese schon sehr erfolgreich CDs an viele Kunden verkauft hat und in der Öffentlichkeit ein anerkanntes Unternehmen ist. Er ruft mit seinem Browser diese Webseite auf, stöbert im Warenbestand der Firma, sucht sich eine CD aus und möchte diese über seine Kreditkarte kaufen. Spätestens hier wird er aber skeptisch, da er nicht weiß, wem er nun eigentlich die Kreditkarteninformationen anvertraut. Idealerweise ist es tatsächlich diese Firma, bei der er die Ware kauft, es könnte sich aber auch um eine Webseite der Firma »Hacker und Co.« handeln, die auf Kreditkartenbetrug spezialisiert ist und die Webseite einfach kopiert hat, um sich als »Internet-Music Online GmbH« auszugeben.

Ein solcher Angriff ist nicht nur theoretisch möglich, er wurde schon sehr erfolgreich durchgeführt. Hier das Vorgehen: Eine Webseite lässt sich mit einfachen Mitteln kopieren. Es sind im Internet Tools erhältlich, mit denen man eine URL konfiguriert und die dann selbstständig nahezu die gesamte Seite downloaden. Die Daten kann man anschließend auf einen eigenen Webserver installieren. Damit hat man den Online-Shop kopiert.

Das nächste Problem ist die Namensauflösung: Wenn Kunden die Seite http://www.internet-music-online.de aufrufen, kontaktiert der Browser seinen nächsten DNS-Server, der die IP-Adresse des Rechners www in der Domain internet-music-online.de auflöst. Normalerweise würde sich das DNS-System mit der betreffenden Domäne verbinden und diese Informationen direkt abfragen. Kommen aber diese Art von Anfragen öfter vor, hält der lokale DNS diese Daten in einem Cache, um diese Anfrage beim nächsten Mal direkt zu beantworten. Und der Cache stellt den Angriffspunkt beim Umleiten von Internetdomänen dar. Hackerangriffe zielen darauf, diese Cache-Informationen zu fälschen und die IP-Adressauflösung auf die eigene Domain zu lenken. Man spricht hier von einem »vergifteten« DNS-Cache.

Sind diese Vorbereitungen seitens »Hacker und Co.« eingeleitet, wird jeder Zugriff auf http://www.internet-music-online.de auf die gefälschte Seite der Betrüger gelenkt, ohne dass es der Anwender bemerkt. In dieser Form besteht keine Möglichkeit für Joe User festzustellen, ob er sich wirklich beim Online-Shop befindet oder nicht.

Gäbe es hier keine Lösung, wäre der Internethandel nicht möglich. Wie aber kann man sich nun sicher sein, dass man wirklich mit dem Online-Shop verbunden ist? Hierzu muss der Anbieter einige Vorbereitungen treffen, bevor er die Webseite in Betrieb nimmt.

Ein Ausweis ist ein Dokument, das von einer Instanz ausgestellt wurde, die anerkannt und akzeptiert ist. Diese Funktion nimmt im realen Leben eine Behörde wahr, die über das Einwohnermeldeamt Daten der Bevölkerung aufnimmt, überprüft und ein Dokument erstellt, das diese zertifiziert. Die Anerkennung des Ausweises resultiert aus der Glaubwürdigkeit der Behörde.

Genau genommen steht zwischen zwei Personen, die sich gegenseitig ihre Authentizität belegen möchten, eine dritte Instanz, die dies in Form der von ihr ausgestellten Ausweise ermöglicht. Den gleichen Ansatz führt man im Internet durch: Eine dritte, vertrauenswürdige und anerkannte Instanz, nennen wir sie »CA« für »Certificate Authority«, stellt fälschungssichere Dokumente aus, mit denen sich der Eigentümer ausweisen kann. Der Eigentümer dieses Ausweises und nur dieser kann dieses Dokument veröffentlichen.

Die CA ist ein in der Öffentlichkeit anerkanntes Unternehmen mit einem eigenen Schlüsselpaar, das sie zur digitalen Unterschrift der von ihr ausgestellten Dokumente verwendet. Der Public Key der CA ist im Internet verbreitet, idealerweise ist er Bestandteil jeder Standardinstallation eines Internetbrowsers wie des Netscape Communicator, des Opera Browser oder des Internet Explorer.

Bevor sich also die Beispielfirma »Internet-Music Online GmbH« mit dem Online-Geschäft befasst, besorgt sie sich einen Ausweis dieser Zertifizierungsbehörde. Dazu begibt sich ein Mitarbeiter der Firma zur CA, um dort seine Identität und die seiner Firma nachzuweisen. Er übergibt der CA seine Daten, darunter Name der Firma, Ort, Telefonnummer und Mailadresse.

Außerdem übergibt er den Public Key der Firma »Internet-Music Online GmbH«, die sich in der Vorbereitung ein Schlüsselpaar mit Public und Private Key erzeugt hat. Der private Schlüssel wird streng vertraulich behandelt und verlässt niemals das Unternehmen.

Die CA überprüft die Angaben zur Identität der Firma und stellt nach zweifelsfreier Glaubhaftigkeit der Daten einen digitalen Ausweis aus. Dies geschieht über ein digitales Dokument, auf dem die Daten der Firma im Klartext stehen, inklusive des Public Key des Unternehmens. Dieses Dokument wird von der CA mit deren digitaler Unterschrift komplettiert.

Fassen wir noch einmal zusammen: »Internet-Music Online GmbH« besitzt ein Schlüsselpaar mit Public und Private Key. Die Unternehmensdaten werden zusammen mit dem Public Key von einer CA verifiziert und mit deren Unterschrift signiert. Das Ergebnis ist ein digitales Zertifikat der Firma, ausgestellt von der CA. Es hat eine Gültigkeitsdauer von einem oder mehreren Jahren und muss nach Ablauf dieser Zeit erneuert werden. Für das Ausstellen eines solchen Dokuments bezahlt die Firma, die auf ein solches angewiesen ist, Geld an die CA. Deren Kapital ist das Vertrauen, das man in ihren Namen setzt. Eine CA kann es sich nicht leisten, ein einziges Zertifikat mit falschen Daten auszustellen, da ansonsten ihr Ruf verloren geht und ihre Ausweise keine Glaubwürdigkeit mehr haben.

Betrachten wir nun die Anwendung eines digitalen Zertifikats. Das Unternehmen »Internet-Music Online GmbH« installiert eine Webseite mit Online-Katalog und Bestellformularen. Das Zertifikat der CA wird installiert und schließlich nimmt der Shop seinen Betrieb auf.

Joe User startet seinen Browser erneut und öffnet die URL http://www.internet-music-online.de und wird automatisch auf https://www.internet-music-online.de umgeleitet. Bei dem Protokoll »https« handelt es sich um eine verschlüsselte Verbindung, bei der das digitale Zertifikat zum Einsatz kommt. Hier die Vorgänge im Einzelnen:

1. Joe User öffnet die Seite http://www.internet-music-online.de und wird auf https://www.internet-music-online.de umgeleitet.
2. Beim Verbindungsaufbau übersendet der Webserver des Online-Shops das digitale Zertifikat, das von der CA ausgestellt ist.
3. Joe User überprüft das Zertifikat, indem er die digitale Unterschrift der CA verifiziert. Dies kann er tun, weil er auf seinem Browser bereits den Public Key der CA vorinstalliert hat. (Jeder Browser hat eine Liste der Public Keys aller Standard-Ass.) Die Verifikation der Unterschrift des Zertifikats entspricht genau dem Vorgehen aus Abschnitt 6.5. Joe User kann demnach sicher sein, dass die Daten, die auf dem Zertifikat dokumentiert sind, von der CA überprüft wurden. Fassen wir noch einmal zusammen, was Joe User zu diesem Zeitpunkt weiß:

- Er weiß, dass er im Besitz des Public Key der CA ist, weil er über eine Standardimplementierung des Browsers verfügt. Der Hersteller des Browsers hat eine Liste aller Standard-CAs vorgegeben.
- Er weiß, dass das digitale Zertifikat von der CA ausgestellt wurde. (Nur diese hat den Private Key zum Leisten dieser Unterschrift.) Dies konnte er überprüfen, weil er den Public Key der CA hat.
- Er weiß, dass die CA die Daten von »Internet-Music Online GmbH« überprüft hat. Somit kennt er neben dem Namen und der Telefonnummer des Online-Shops auch dessen verifizierten Public Key.

4. Joe User generiert eine Zufallszahl, die als geheimer Schlüssel verwendet wird, verschlüsselt diesen mit dem Public Key des Online-Shops und übersendet diesen an den Webserver von »Internet-Music Online GmbH«.
5. Der Online-Shop dechiffriert den geheimen Schlüssel mit Hilfe seines Private Key. Beide Seiten vereinbaren einen symmetrischen Verschlüsselungsalgorithmus, den sie zur weiteren Kommunikation verwenden.
6. Joe User füllt das Online-Bestellformular aus, trägt seine Kreditkartennummer ein und kauft eine CD. Die Daten werden dabei verschlüsselt zwischen dem Shop und dem Kunden ausgetauscht und sind für Dritte nicht lesbar.

Warum nun ist diese Art der Datenübertragung sicher? Die Antwort hierzu liefert Punkt 3 des vorangegangenen Protokolls: Joe User bekommt ein digitales Zertifikat vom Server übertragen,

- das echt ist, weil die CA es unterschrieben hat,
- auf dem der Public Key des Online-Shops eingetragen ist, der von der CA überprüft wurde.

Dieses Zertifikat kann von niemand anderem als vom Online-Shop selbst verwendet werden, weil der darin enthaltene Public Key eindeutig mit dem entsprechenden Private Key des Online-Shops verbunden ist. Wird der Public Key gegen einen anderen Public Key (beispielsweise von »Hacker und Co.«) ausgetauscht, wäre die Unterschrift der CA ungültig, was bei der Überprüfung des Zertifikats durch Joe User aufgefallen wäre.

Wäre der Anwender auf eine gefälschte Seite umgelenkt worden, könnte diese ihm kein gültiges Zertifikat vorweisen.

Somit hat man auf diese Weise ein Verfahren entwickelt, mit dem man eine eindeutige Authentifizierung im Internet umsetzen und weiterhin eine verschlüsselte Verbindung realisieren kann, ohne dass man als Kommunikationspartner zusammenkommen und Schlüssel austauschen muss.

Bei dem vorgestellten Protokoll wurde kein spezielles Verfahren genannt. Man kann für den Hash-Algorithmus MD5 (Message Digest Version 5) oder SHA (Secure Hash Algorithm) einsetzen, für das asymmetrische Verschlüs-

selungsverfahren bieten sich die Verfahren Diffie-Hellman, RSA oder andere Architekturen an.

Der Schwachpunkt des Protokolls liegt in der Geheimhaltung des Private Key des Online-Shops.

- Wird er publik, kann jemand anderer im Namen des Besitzers unterschreiben oder sich fälschlicherweise authentifizieren.
- Geht er verloren, kann er nicht wieder ersetzt werden. Damit kann der entsprechende Public Key nicht mehr eingesetzt werden und das digitale Zertifikat ist nicht mehr anwendbar.

Für beide Probleme gibt es Lösungen, die aber nicht optimal sind.

Wird der private Schlüssel publik, kann das Zertifikat auf die so genannte »Revocation List« der CA gesetzt werden. Hier veröffentlicht die Certificate Authority alle Zertifikate, die auf Wunsch der Besitzer oder aber auch der CA als ungültig erklärt wurden. Es ist die Pflicht des Anwenders, sich zu informieren, ob diese Liste das präsentierte Zertifikat enthält.

Bevor der Private Key verloren geht, kann man ein Backup des Schlüssels machen, das ebenso sicher aufbewahrt werden muss wie das Original.

6.7 Authentifizierung

6.7.1 Passwörter

Die Anforderungen einer minimalen Sicherheitspolitik verlangen vom User, dass er sich für den Zugriff auf seine Daten und Anwendungen ein Passwort wählt, das hinreichend lang und ausreichend »kompliziert« ist. Das macht es unwahrscheinlich, dass dieser Schlüssel von anderen erraten werden kann. Kontraproduktiv wirkt dabei die Bequemlichkeit der Anwender, die sich keine komplizierten Zugangscodes ausdenken und merken wollen.

Was bei manchen Anwendern oft als »kleinlich« und »lästig« empfunden wird, birgt bei Vernachlässigung unter Umständen große Sicherheitsrisiken. So wurden bis vor einiger Zeit die Passwörter der Unix-Betriebssysteme mit dem Algorithmus »crypt« gehashed und in der Datei /etc/passwd (oder ähnlich) hinterlegt. Selbst wenn die Hashes nicht direkt entschlüsselt werden konnten, waren viele Passwörter von den Anwendern aus dem täglichen Leben gewählt. Diesen Schwachpunkt machen sich »Wörterbuch-Attacken« zunutze, die wie folgt ablaufen:

1. Organisiere eine Wörterliste. Nach Möglichkeit nimmt man eine Aufzählung aller Wörter aus dem eigenen oder einem sonst verwendeten Sprachgebrauch. Diese Listen liegen im Internet auf vielen Seiten zum Download bereit.

2. Organisiere eine Liste von gehashed-ten Passwörtern, entweder über die Datei »passwd« oder über den Export einer Directory-Datenbank in ein Textfile.
3. Nimm das erste Wort aus der Wortliste und erzeuge dessen Hash-Wert.
4. Vergleiche diesen Wert mit jedem Wert aus der Passwortliste. Bei einer Übereinstimmung ist das Passwort aufgedeckt.
5. Streiche das erste Wort aus der Wortliste.

Die Schritte 3 bis 5 werden auf alle Wörter der Liste angewandt. Es sind auch Variationen möglich, wie zum Beispiel die Kombination von Wörtern mit Zahlen (»Sonne1«), Vertauschungen (»Snoen«), Variationen (»sOnne«) oder Invertierungen (»Ennos«). Mit den heutigen durchschnittlichen Prozessorleistungen stellen solche Aufgaben pro Passwort einen Aufwand von einigen Minuten dar, die Programme hierzu kann man im Internet downloaden. Es ist noch einmal zu betonen, dass der Hash-Algorithmus hierbei nicht gebrochen wurde, sondern nur dessen Anwendung.

Für die Wahl eines sicheren Passworts sollten folgende Kriterien gelten:

- Es muss hinreichend lang sein.
- Es darf kein Wort aus dem Wörterbuch sein, vielmehr sollte es möglichst sinnfrei sein.

Gerade diese beiden Punkte machen es einem Anwender schwer, sich solche Wörter auszudenken und vor allem zu merken.

Man kann versuchen, über administrative Mittel die User zu zwingen, bestimmte Grundregeln der Passwortsicherheit einzuhalten. Eine so genannte Passwort-Policy enthält alle administrativen Mittel, um die Qualität der Zugangsdaten zu verbessern:

- Möglichkeit für die User, ihre Passwörter zu ändern: Kann er dies nicht tun, ist ein schwaches Passwort für eine lange Zeit gültig und stellt ein potenzielles Sicherheitsloch dar.
- Minimale Länge: Mit der Angabe einer Mindestlänge von Zeichen für ein Passwort wird verhindert, dass zu kurze Schlüssel verwendet werden.
- Verfallsdatum: Mit dem Altern eines Passworts verfällt es nach einer bestimmten Zeitspanne und kann nur noch unter Zuhilfenahme eines Administrators geändert werden. Ein ungültiges Passwort gestattet keine weiteren Zugang mehr zu den damit verbundenen Applikationen.
- Syntax-Check: Durch eine Verifikation wird überprüft, ob das Passwort syntaktischen Kriterien genügt. Ob es eine Mindestanzahl von Ziffern oder eventuelle Sonderzeichen enthält oder ob es nicht in dieser oder ähnlicher Weise in Wortlisten auftaucht usw.

- »History«: Mit dem Einführen einer History verhindert man, dass ein User ein altes Passwort nach einer einmaligen oder mehrmaligen Änderung wiederverwendet.
- Account-Lock: Dadurch wird verhindert, dass man durch »Ausprobieren« an Applikationen das Passwort herausfindet. Im Allgemeinen werden zwei erfolglose Authentifizierungsversuche gestattet, bevor der Account gesperrt wird.

Anwender werden immer versuchen, Passwörter so zu wählen, dass sie sich diese möglichst gut merken können. Die von der Passwort-Policy geforderte »kryptische« Wahl eines Zugangscodes eröffnet unter Umständen Angreifern einen anderen Weg, um zu Passwort-Informationen zu kommen: Der Anwender schreibt sich den Code auf und bewahrt ihn an einer vermeintlich sicheren Stelle auf. Einen Blick unter die Tastatur eines Arbeitsplatzrechners oder in die Schreibtischschublade kann manchmal bereits genügen, damit Zugangsdaten in unbefugte Hände gelangen.

Eine Möglichkeit für eine Lösung wäre die Verwendung von Zitaten.

Beispiel: »Ein Ring, sie zu knechten, sie alle zu finden ...« wird für die Erzeugung eines Passworts verwendet: Das Zitat kann man sich leicht merken und »ähnlich« aussehende Zeichen werden substituiert, Zahlwörter in Ziffern ausgedrückt, Groß- und Kleinschreibung alterniert. So oder so ähnlich kann man sich sein eigenes Regelwerk von Substitutionen zusammenstellen. In diesem Beispiel wird aus den ersten vier Wörtern des Zitats »1rin9$1ezU«, ein Ausdruck, der zwar nicht wirklich sicher ist, da die Zeichen immer noch durch das Zitat und die Substitutionsvorschriften miteinander korreliert sind, das aber um ein Vielfaches besser ist als so manche Passwörter, die heute wichtige Applikationen schützen.

6.7.2 Digitale Zertifikate

Wir haben im vorherigen Abschnitt die Anwendung des digitalen Zertifikats als Authentifizierung verwendet. Der Online-Shop weist sich mit diesem Dokument gegenüber dem Kunden aus und dieser glaubt ihm, weil man mit Hilfe der Kryptografie zweifelsfrei nachweisen kann, dass der Gegenüber niemand anderes ist als der, für den er sich ausgibt. In manchen Geschäftsbeziehungen möchte aber auch die Serverseite wissen, mit wem sie hier Handel betreibt. Dies kann man dann mit einem clientseitigen Zertifikat einrichten. Dieses wird genauso eingesetzt wie das besprochene serverseitige Zertifikat. Der Kunde muss sich hierfür glaubhaft bei der CA vorgestellt haben und neben seinen Daten einen Public Key hinterlegen, den er vorher generiert hat.

Manche Unternehmen setzen diese Art der Authentifizierung direkt in ihrem internen Netzwerk ein. Sie installieren sich einen CA-Server und sind damit in der Lage, für alle Mitarbeiter digitale Zertifikate auszustellen. Für den Fall, dass nun ein Angestellter von zuhause aus eine Telefonverbindung

zum unternehmensinternen RAS-Dienst (remote access service) aufbauen möchte, muss sich das Unternehmen sicher sein, dass nicht ein Unternehmensfremder unberechtigterweise Zutritt zum internen Netzwerk erhält. Dazu verlangt das RAS-System, bevor es die telefonische Datenverbindung erlaubt, ein clientseitiges Zertifikat, das von der unternehmenseigenen CA ausgestellt wurde. Wenn der Anrufer dies vorlegen kann, bekommt er den Zugriff auf das interne Netz. Ein solches Zertifikat hat natürlich beim Bücherkauf keine Relevanz, weil die CA von einem beliebigen Unternehmen kein Vertrauenspotenzial gegenüber dem Rest der Welt hat. Dies ist aber auch nicht Intention dieser Architektur: Es soll lediglich eine Absicherung des Unternehmens gegenüber Fremden beim Zugriff auf interne Ressourcen gegeben sein und dazu ist diese Authentifizierung vollkommen ausreichend. Der Kostenfaktor ist hierbei ebenfalls zu berücksichtigen: Ein digitales Zertifikat bei einer anerkannten CA kostet einen Beitrag, der bei großen Mitarbeiterzahlen nicht unerheblich ist. Bei einer eigenen CA fällt dieser Faktor weg (wenn man mal von den Lizenzkosten der Software absieht).

Es lassen sich fast alle Dienste in diese Authentifizierungsarchitektur einbinden: vom Remote-Access-Service der Dial-In-Verbindung bis zum Web-, Proxy- und Maildienst ist fast bei jedem Hersteller eine Zertifikat-basierte Authentifizierung möglich. Dies hat einen weiteren nicht unerheblichen Vorteil: Der Benutzer benötigt fast keine Passwörter. Mit dem Vorweisen seines Zertifikats wird er automatisch erkannt und einer Benutzergruppe zugeordnet. Ganz ohne Passwort kommt diese Architektur aber nicht aus: Der Private-Key ist üblicherweise auf dem lokalen System verschlüsselt gespeichert und lässt sich nur mit einem Passwort aktivieren. Ansonsten wäre es einem potenziellen Dieb eines Computers möglich, auf alle Ressourcen zuzugreifen, weil er einen ungeschützten Zugriff auf den Private Key des Benutzers hätte.

Der User hat die Verantwortung, seinen eigenen Private Key geheim zu halten. Er hat dafür Sorge zu tragen, dass dieser nicht in fremde Hände gelangt. Dazu darf er den Schlüssel niemals preisgeben. Leider wird es immer Angriffe geben, die darauf zielen, in den Besitz fremder Private Keys zu gelangen. Der Schaden auf Seiten des Benutzers kann dann vom unberechtigten Zugriff Dritter (im Namen des Eigentümers des privaten Schlüssels) auf wichtige Daten bis hin zur Ungültigkeit aller bislang geleisteten Unterschriften reichen. Um diesem Problem entgegenzuwirken, werden diese Schlüssel mehr und mehr auf Smartcards hinterlegt. Dabei handelt es sich um Datenträger im Scheckkartenformat, auf denen sich entweder ein Datenspeicher oder ein kompletter Rechner im Miniaturformat befindet. Im letzten Fall generiert die Smartcard das Schlüsselpaar selbst und sorgt dafür, dass der Private Key niemals die Karte verlässt. Das heißt, alle Ver- und Entschlüsselungen werden auf dieser Karte durchgeführt und nur die Ergebnisse werden von der Karte geliefert. Die Verwendung einer solchen Karte ist mit einem PIN geschützt und damit kann der Besitzer sich an jedem dafür vorgesehenen Smartcard Reader authentifizieren oder sogar digitale Unterschriften leisten.

6.8 Public Key Infrastructure (PKI)

Viele Internetseiten preisen ihre Inhalte über SSL-Verschlüsselung an. In den Medien wird diese Methode als das Mittel der geheimen und sicheren Datenübertragung gepriesen, ohne dass sich jemand wirklich damit auseinander setzt. Bei einem Zugriff auf eine sichere Seite erscheinen plötzlich lästige Fenster auf den Bildschirm, die man wegklicken muss, bevor man sich zum Beispiel dem Homebanking widmen kann. Es scheint, als ob sich niemand wirklich die Mühe macht und nachliest, was in diesen Informationen tatsächlich steht. Dabei ist dies gerade der wichtigste Punkt in der gesamten Architektur der PKI: die Bestätigung des Anwenders, dass er dem vorgelegten Zertifikat und insbesondere der CA, die dieses ausgestellt hat, vertraut. Ziel dieses Abschnitts ist es, die Vorgänge in dieser Architektur vorzustellen und ein gewisses Verständnis für Sicherheitsaspekte in der Datenübertragung aufzubauen.

Bisher wurden die Architekturen der symmetrischen und der asymmetrischen Verschlüsselung beschrieben. Der vorliegende Abschnitt macht Sie nun mit Anwendungen dieser Technologie vertraut.

6.8.1 Secure Socket Layer (SSL)

Sie haben bisher einen Überblick über die Aktionen beim Aufbau einer verschlüsselten Verbindung bekommen. Es ist natürlich klar, dass all diese Aktionen nicht vom Benutzer durchgeführt werden können. Einerseits hat er sicherlich nicht immer das erforderliche Verständnis für die Notwendigkeit und die Konsequenzen der einzelnen Schritte, andererseits ist die Fülle aller Maßnahmen in der Anwendung nicht praktikabel. Betrachten wir beispielsweise den Verbindungsaufbau mit einer sicheren Webseite: Der Server sendet ein Zertifikat, der Client nimmt dieses entgegen, extrahiert den Hash der Unterschrift der CA, vergleicht diesen mit den Daten im Zertifikat, generiert im Erfolgsfall einen geheimen Schlüssel, den er mit dem Private Key der Webseiten-Betreiber chiffriert und an den Server zurücksendet. All diese Aktionen würden einen unbedarften Anwender vollkommen überfordern. Daher sind sie in ein Protokoll eingebettet, das das komplette Vorgehen über die verwendete Software automatisiert. Damit dies herstellerübergreifend geschieht, hat man sich auf einen Standard geeinigt: SSL.

Der Secure Socket Layer ist eine Verschlüsselungsschicht im Kommunikationsprotokoll. Diese führt alle Aktionen aus, die – wie bereits beschrieben – eine sichere Verbindung über ein unsicheres Kommunikationsmedium ausmachen. Betrachten wir die Aktionen anhand einer verschlüsselten *http*-Verbindung im Einzelnen (die Abläufe sind zusammengefasst und in der Protokollimplementierung komplexer):

- Ein Benutzer startet seinen Browser und öffnet die URL https://www.internet-music-online.de.

- Die Browsersoftware öffnet eine Socketverbindung zum Server auf Port 443 und initiiert eine Verbindung.
- Der Browser spezifiziert, welchen Algorithmus er für die symmetrische bzw. die asymmetrische Verschlüsselung akzeptiert. (Diese Einträge sind von der Art und der Konfiguration des Browsers abhängig.)
- Der Server übersendet sein digitales Zertifikat an den Browser und spezifiziert, welche Art der symmetrischen Verschlüsselung verwendet werden soll. Gleichzeitig wird ausgehandelt, welche Verschlüsselung beide Seiten übereinstimmend akzeptieren. (Hat der Client beispielsweise nur eine Schlüssellänge von 40 Bit und der Server akzeptiert alles zwischen 40 und 1024 Bit, werden 40 Bit als Basis für die Verschlüsselung genommen.)
- Der Browser überprüft die Gültigkeit des angebotenen Zertifikats: Welche CA hat das Zertifikat ausgestellt? Für den Fall, dass die CA nicht bekannt ist, wird der Benutzer informiert und aufgefordert, zu entscheiden, ob er dem Zertifikat vertraut.
- Ist eine Übereinstimmung des Hash-Werts der CA mit dem des Dokuments vorhanden? Ist dies nicht der Fall, wird das Zertifikat als ungültig erkannt und der Benutzer wird aufgefordert, zu entscheiden, ob er dem Zertifikat vertraut.
- Ist das vorgehaltene Zertifikat noch gültig oder ist es bereits abgelaufen? Die Gültigkeitsdauer beträgt bis zu einigen Jahren. Ist dieses Intervall abgelaufen, wird der Benutzer informiert und aufgefordert, zu entscheiden, ob er dem Zertifikat vertraut.
- Ist der Hostname des Webservers der gleiche, den die CA bei der Ausstellung des Zertifikats in das Dokument eingetragen hat? Gibt es Unterschiede, wird das Zertifikat als ungültig erkannt und der Benutzer wird aufgefordert, zu entscheiden, ob er dem Zertifikat vertraut.
- Der Browser generiert einen geheimen Schlüssel für die symmetrische Verbindung.
- Der Browser verwendet den Public Key des Webservers, den er aus dem Zertifikat extrahiert, und verschlüsselt den geheimen Schlüssel mit diesem.
- Der Browser übersendet den chiffrierten geheimen Schlüssel an den Webserver.
- Der Webserver entschlüsselt den chiffrierten geheimen Schlüssel mit seinem Private Key.
- Beide Seiten verwenden in Folge den symmetrischen Verschlüsselungsalgorithmus zur weiteren Kommunikation.

Wir haben hier die clientseitige Authentifizierung über ein digitales Zertifikat unterschlagen, das sich im Ablauf nach Punkt zwei im vorgestellten Protokoll einfügt.

Die Software des Browsers alarmiert den Benutzer nur in Ausnahmefällen und vereinbart die Aktivitäten zum Aufbau einer verschlüsselten Verbindung mit dem Server selbstständig.

Dieser Ablauf ist nicht nur für den Einsatz über *http* vorgesehen, es werden mittlerweile auch weitere Applikationsprotokolle wie *ldap* und *nntp* über diese Architektur gesichert. Vom Schichtenmodell der Protokoll-Layer schiebt sich die SSL-Ebene zwischen die Applikation (beim Webserver *http*) und dem TCP/IP-Stack des Betriebssystems.

6.9 Sicherheit und Mail

Wie bereits in Kapitel 4 vorgestellt, erfolgt die übliche Verwendung von E-Mail zur Datenübertragung unverschlüsselt. Eine solche offene Datenübertragung kann an Konzentrationspunkten wie firmeneigenen MTAs relativ einfach »abgefangen« werden, indem man mit einem einfachen »Snoop« auf dem Netzwerk die ausgetauschten Nachrichten mitprotokolliert. Die belauschten Personen bemerken nicht einmal den Lauscher, da der eigentliche Mailverkehr nicht unterbunden wird. Aber nicht nur bei der firmeninternen Nachrichtenübermittlung sind Mails das Ziel von Lauschangriffen, sondern auch im weltweiten Datenverkehr versuchen Organisationen, politische oder wirtschaftliche Vorteile durch Mitlesen von Nachrichten zu bekommen. Als Gegenmaßnahmen wurden bereits die diversen Verschlüsselungsarchitekturen besprochen. Wir werden uns nun mit den wichtigsten Standards beschäftigen, mit Hilfe derer man sich gegen unbefugtes Mitlesen Dritter absichern kann: PGP und S/MIME.

6.9.1 Pretty Good Privacy (PGP)

Einer der Vorreiter der gratis Verschlüsselung ist »Pretty Good Privacy« (PGP). Dieses Programm wurde von Phil Zimmermann entwickelt und kann für den privaten Gebrauch kostenlos verwendet werden.

Bei dem Programm handelt es sich um ein komplettes Paket, das von der Schlüsselgenerierung bis hin zur digitalen Unterschrift alle Verfahren beinhaltet, die notwendig sind, um eine private und fälschungssichere Kommunikation über das Internet durchzuführen.

Geschichte von PGP

Mit PGP rückte die Notwendigkeit der Verschlüsselung von Daten im Internet erstmals in den Blickpunkt der Öffentlichkeit. Man wurde sich bewusst, dass die Datenübertragungen im Internet mehr oder weniger ungeschützt sind und dass es für andere Personen oder Organisationen recht einfach ist,

die Kommunikation zwischen zwei Partnern zu belauschen. Mit der Einführung von PGP erhoben auf der anderen Seite staatliche Organisationen wie die NSA den Anspruch, jede Art von Kommunikation belauschen zu können. Ansonsten wäre es ihren Aussagen zufolge möglich, dass terroristische Organisationen ungehindert abhörsichere Kommunikationskanäle über das Internet aufbauen. (Eine Vision, die sich leider bestätigte: Mit PGP wurden Daten ausgetauscht, die bei den Terroranschlägen auf das World Trade Center am 11. September 2001 in New York eine Rolle spielten.)

Mit der Implementierung des Public-Key-Verfahrens wurde den Vorbehalten der amerikanischen Regierung bezüglich einer starken Verschlüsselung in einem Senatsbeschluss im Jahre 1991 entsprochen: Beschluss 266 verlangte eine Hintertür in jedem Verschlüsselungsverfahren, mit deren Hilfe sich jede chiffrierte Kommunikation entschlüsseln lässt. Ferner durfte keine Software aus den USA ausgeführt werden, die eine starke Verschlüsselung beinhaltete. Diese Verordnung brachte Phil Zimmermann zu dem Entschluss, vor Inkrafttreten des Beschlusses sein Verschlüsselungsverfahren zu verbreiten, das keine Hintertür enthält. Da die Software frei verfügbar war und Zimmermann nicht für die Verbreitung von PGP durch Dritte zur Verantwortung gezogen werden konnte, hatte eine anschließende Ermittlung keine weiteren Folgen für ihn. Nichtsdestotrotz werden Verstöße gegen die Exportbestimmungen durch die »Ausfuhr« von PGP in den USA von der ITAR (International Traffic in Arms Regulation) geahndet, das heißt, PGP wird als »Waffe« eingestuft.

Funktionsumfang

Verschlüsselung: Mit PGP lässt sich jede Datei chiffrieren und dechiffrieren. Dabei werden symmetrische und asymmetrische Verfahren verwendet.

Signatur: Es ist möglich, eine Datei mit PGP zu signieren. Dadurch kann man die Willensbekundung des Unterzeichners dokumentieren und die Authentizität der Daten sicherstellen.

»Web Of Trust«: Durch eine gegenseitige Zertifizierung kann man die öffentlichen Schlüssel anderer Personen signieren und damit ein »Netz des Vertrauens« schaffen.

Architektur von PGP

PGP nutzt folgende Mechanismen:

Kompression: Um die Verschlüsselung der Daten möglichst effizient zu gestalten, werden die Dateien vor dem Chiffrieren komprimiert. Die ersten Versionen von PGP nutzten pkzip von PKWare, seit Version 7.0.3 wird LZS oder Deflate verwendet.

Message Digest: Verwendet werden MD5 (Message Digest Version 5) und SHA (Secure Hash Algorithm). In jedem Fall ist SHA dem als unsicher eingeschätzten MD5 vorzuziehen.

Symmetrische Verschlüsselungsverfahren: Zum Einsatz kommen CAST (Carlisle Adams, Stafford Tavares), IDEA (International Data Encryption Algorithm), Triple-DES (Data Encryption Standard).

Asymmetrische Verschlüsselungsverfahren: PGP nutzte in den ersten Versionen ausschließlich RSA und Diffie-Hellman (DH), später kam ElGamal, eine Modifikation des ersten asymmetrischen Verfahrens von DH hinzu.

Bei der Installation von PGP wird ein Schlüsselpaar erzeugt. Diese beiden Schlüssel werden mit einer Passphrase gesichert auf der Festplatte hinterlegt. Nur derjenige, der dieses Passwort (»Wort« ist unzureichend, normalerweise verwendet man einen Satz) kennt, ist in der Lage, den Private Key freizuschalten und PGP mit diesem Schlüsselset zu verwenden.

Nach der Erzeugung des Schlüsselpaars kann der Anwender mit jeder anderen Person chiffriert kommunizieren. Dabei wird der Public Key eines jeden »Gesprächspartners« ausgetauscht. Auf diese Weise sammelt man im Laufe der Zeit eine Reihe verschiedener öffentlicher Schlüssel unterschiedlicher Personen an, die an einem so genannten (digitalen) Schlüsselbund gesammelt werden.

Soll eine Nachricht an eine andere Person verschlüsselt übermittelt werden, so muss sich der Absender erst den Public Key des Empfängers verschaffen (falls dieser nicht bereits an seinem digitalen Schlüsselbund vorhanden ist). Mancher versendet seinen Public Key automatisch per SigFile in jeder Mail.

Mit PGP kann man auch eine Nachricht signieren. Dazu verwendet man seinen eigenen Private Key, mit dem der Hash des Dokuments verschlüsselt als Attachment an die Mail angehängt wird.

Eine Kombination von Unterschrift und Verschlüsselung ist ebenfalls möglich.

Wie bei anderen Hybridverfahren auch wird ein symmetrischer Schlüssel erzeugt, mit dem die eigentliche Nachricht chiffriert wird. Mit dem Public Key des Empfängers wird der symmetrische Schlüssel chiffriert und an die Nachricht angehängt. Diese Kombination wird als Nachricht an den Empfänger versendet.

Ausführliche Informationen zu PGP finden sich in [34].

6.9.2 S/MIME

Mit S/MIME sollen die drei wichtigen Herausforderungen des privaten Mailverkehrs gelöst werden:

- Authentizität des Absenders: Der Empfänger kann überprüfen, von wem er die Mail bekommen hat. Er vertraut auf die Sicherheit eines digitalen Zertifikats einer bekannten CA und den Umstand, dass nur der Absender im Besitz des privaten Schlüssels ist, der zu dem Public Key gehört, der in dem digitalen Zertifikat angegeben ist.

- Vertraulichkeit des Inhalts: Aufgrund der Tatsache, dass die Datenübertragung verschlüsselt erfolgt, können Dritte zwar die Chiffrate abhören. Die Stärke der Chiffieralgorithmen verhindert aber, dass die Klartexte ohne Kenntnis des entsprechenden Schlüssels sichtbar werden.
- Sicherheit vor Modifikationen: Da die Daten verschlüsselt werden, können sie nicht gelesen und noch weniger modifiziert werden. Eine Veränderung der Daten ergibt kein gültiges Chiffrat mehr, so dass eine Modifikation schnell bemerkt wird.

Architektur

S/MIME arbeitet mit drei unterschiedlichen Betriebsmodi: Signiert (Signed), Verschlüsselt (Encrypted) und Signiert+Verschlüsselt (Signed+Encrypted).

- Signiert: Wenn ein Absender seine Mail vor dem Versenden signiert, so nimmt er einen Hash-Wert der Mail, verschlüsselt ihn mit seinem Private Key und hängt dieses Ergebnis an die Mail. Der S/MIME-Standard sieht vor, dass der Absender sein digitales Zertifikat der Mail zufügen kann. Mit diesem Modus wird die Nachricht immer noch öffentlich lesbar über das Netz ausgetauscht, sie ist aber vor Verfälschung sicher, da mit einer Änderung des Inhalts der Nachricht automatisch die Unterschrift verfällt. Zur Erinnerung: Diese Unterschrift kann nur derjenige leisten, der im Besitz des Private Key ist.

  Die Verifikation der Signatur auf Empfängerseite wird vom S/MIME-Standard ebenfalls festgelegt: Nach Erhalt der Mail wird über den Klartext ein Hash-Wert genommen, mit Hilfe des Public Key des Absenders wird der ursprüngliche Hash-Wert, den der Absender ermittelt hat, sichtbar gemacht, beide Werte werden verglichen und müssen für eine gültige Signatur übereinstimmen.
- *Verschlüsselt*: Der Absender schreibt die Nachricht im Klartext. Der S/MIME-Standard erzeugt eine Zufallszahl Z1, die als symmetrischer Schlüssel verwendet wird. Mit diesem Schlüssel wird die Nachricht chiffriert. Für die Übertragung des symmetrischen Schlüssels wird vom Empfänger der Public Key P2 angefordert. Mit diesem Public Key wird der Schlüssel, mit dem die Nachricht chiffriert wurde, verschlüsselt und an die Nachricht angehängt. Eine solche Mail besteht damit aus dem Z1-chiffrierten Inhalt und dem P2-chiffrierten Z1.

  Diese Nachricht wird in dieser Form über das Netzwerk versendet und kann nicht mitgelesen werden. Nach Erhalt der Nachricht dechiffriert der Empfänger den P2-chiffrierten Schlüssel Z1 mit Hilfe seines Private Keys P1. Mit Z1 dechiffriert er anschließend den Text der Nachricht.
- *Signiert und verschlüsselt*: Der Absender durchläuft das Protokoll »Signed« gefolgt von »Encrypted« und kann optional sein digitales Zertifikat an die Mail anhängen.

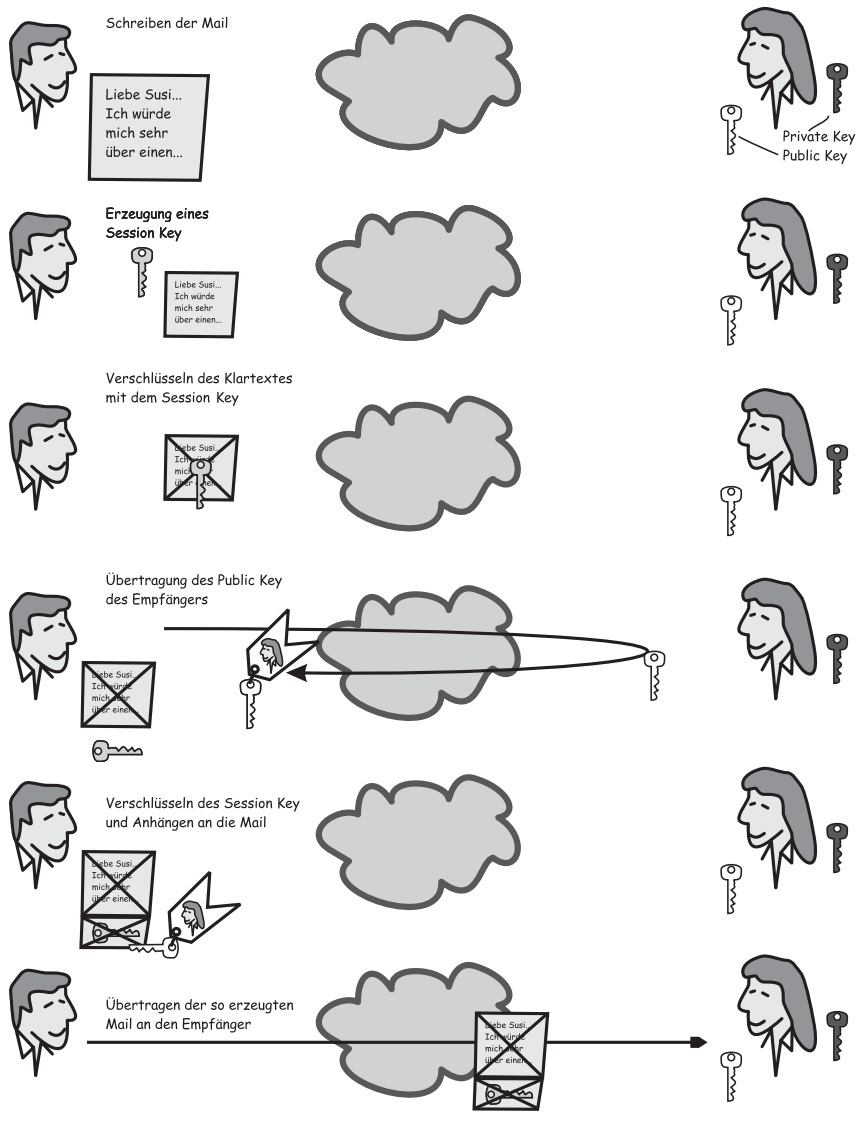


Abbildung 6.13: Verschlüsseln einer Mail mit S/MIME

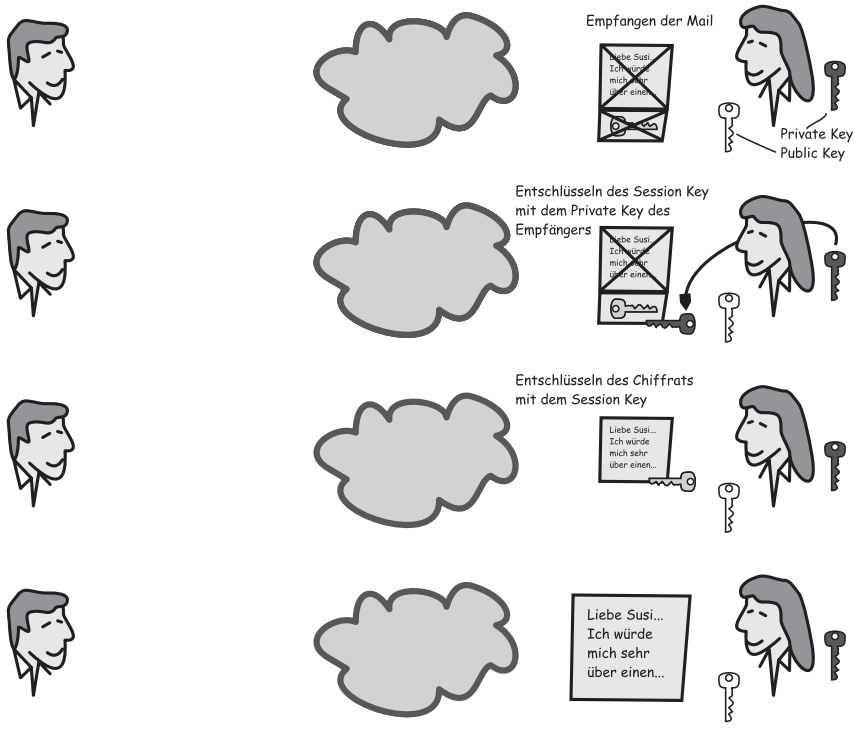


Abbildung 6.14: Übertragung, Empfangen und Entschlüsseln der Mail mit S/MIME

6.9.3 Einschränkungen

Ein standardisiertes Regelwerk für die Verschlüsselung von Daten ist sicher ein gutes Mittel, um Dritte daran zu hindern, private Kommunikationen zu belauschen. Eine Reihe von Einschränkungen sind aber zu beachten, will man sich ausschließlich auf diese Architektur verlassen.

Anwender-»Trägheit«

Leider ist das Sicherheitsbewusstsein der Anwender in den wenigsten Fällen ausreichend ausgebildet, um die notwendigen Vorkehrungen zum privaten Datenaustausch zu treffen. Ist vielleicht innerhalb eines Unternehmens ein bestimmter Standard noch umsetzbar, so trifft der Anwender bei gelockerten Vorgaben und insbesondere im privaten Austausch auf eine Situation, in der der Gegenüber entweder einen anderen Standard verwendet oder schlichtweg kein Sicherheitsempfinden hat.

Architektureinschränkungen

- *Passive Daten*: So stark die verwendete Verschlüsselung auch sein mag, verhindert sie dennoch nicht, dass bestimmte Daten verraten werden. Dazu gehören der Zeitstempel des Nachrichtenaustauschs, der Umfang und die Kommunikationspartner. Ein Dritter, der im Netzwerk auf dem

Nachrichtenein- und -ausgang lauscht, kann protokollieren, wer wann in welchem Umfang mit wem kommuniziert hat. Selbst wenn der Inhalt nicht bekannt ist, so verraten diese Sekundärdaten viel über den möglichen Inhalt.

- *Passive Falsifizierung von Daten durch Rewriting der Mails*: Dabei ist nicht einmal die bewusst durchgeführte Veränderung Dritter am Inhalt einer Mail verantwortlich dafür, dass der Empfänger eine falsche Signatur des Dokuments erhält oder die Nachricht nicht mehr entschlüsseln kann. Sobald ein MTA ein einziges Bit im Inhalt des Nachrichtenblocks aufgrund besonderer Umstände verändert, ist die Vertraulichkeit des Nachrichtenaustauschs vermeintlich beeinträchtigt. Nur eine korrekte Signatur einer Nachricht in Zusammenhang mit einer fehlerfreien Entschlüsselung und der Gewissheit, dass die Schlüssel nur in autorisiertem Besitz sind, sichert die Privatsphäre.

6.10 Kryptografie im kommerziellen Einsatz

6.10.1 Vertrauensverhältnisse

Eines der größten Probleme im Netz ist der Aufbau einer Vertrauensstellung zu einer Person oder einer Institution, die man unter Umständen nicht persönlich kennt. Technisch gesehen ist man zwar in der Lage, kryptografisch abgesichert zu kommunizieren, das Vertrauen auf die Identität des Gegenübers insbesondere bei Verbindungen zwischen zwei Parteien, die sich nie vorher kennen gelernt haben, ist nur über Umwege zu bekommen. Umgekehrt vertraut der Einzelne einer bekannten oder ihm nahestehenden Person mehr als einem Fremden. Innerhalb dieser Spanne bewegt sich der Einsatz von Public-Key-Verfahren, die eine netzbasierte Kommunikation unterstützen.

Man kann drei verschiedene Vertrauensstellungen unterscheiden:

- *Direktes Vertrauen*. Eine persönlich bekannte Person genießt eine besondere Vertrauensstellung. Ein digitales Zertifikat, das von dieser Person stammt, wird ohne langes Hinterfragen als vertrauenswürdig akzeptiert. Wer dieses Zertifikat ausgestellt hat, spielt keine Rolle, wichtig ist nur, dass es sich um ein X.509-konformes Dokument handelt. Beispiel: Alice und Bob kennen sich sehr gut und vertrauen einander. Alice generiert ein Dokument, das sie selbst mit ihrem eigenen Private Key unterschreibt und das sie in ein X.509-Format bringt. Sind beide räumlich voneinander getrennt, kann Alice das Zertifikat an Bob senden, anschließend können sie sich per Telefon miteinander verständigen, ob beispielsweise eine Prüfsumme über das Zertifikat auf beiden Seiten identische Werte ergibt. (Hier ist die vertraute Stimme am Telefon ein Beleg für die Authentizität des jeweiligen Gegenübers.)

Dieses Verfahren ist sehr aufwändig, wenn viele Personen miteinander kommunizieren. Dies erfolgt über eine Verifikation des Gegenübers per Telefon und eine umfangreiche lokale Sammlung von Zertifikaten von Personen, denen man bereits vertraut. Personen, die sich nicht kennen, können keine Vertrauensbasis aufbauen. Gerade dies ist aber für Verbindungen im Internet notwendig, wenn es um Online-Shops, Internetbörsenhandel oder Homebanking-Applikationen geht.

- *Vertrauensnetz*. Hier werden Abhängigkeiten des Vertrauens unter den verschiedenen Personen definiert. So wie im ersten Beispiel genießen Alice und Bob wechselseitig ein hohes Vertrauen. Dieses Potenzial wird genutzt, indem der jeweils andere eine dritte Person in die Vertrauensstellung einbezieht. So vertraut Alice auch einem digitalen Zertifikat von Carol, das von Bob signiert wurde. (Bob kennt oder überprüft die Identität von Carol und signiert ihr Zertifikat. Nur er ist in der Lage, dies zu tun, da nur er seinen Private Key kennt.) Auf diese Weise kann Alice nun Carol vertrauen, obwohl sie sie noch nie zuvor getroffen hat.

  Ein solches Vertrauensnetz kann man nun hierarchisch ordnen: Alice vertraut Bob, der Carol vertraut, die wiederum Malcolm ein digitales Zertifikat ausstellt. Aus Sicht von Alice ist Malcolm zwar immer noch vertrauenswürdig, allerdings »nicht mehr so sehr«, da die dargestellte Kette bereits relativ lang wurde. Hier kann jeder nach eigenem Gusto definieren, wann keine Vertrauensbasis mehr vorhanden ist.

  Der Nachteil einer solchen Struktur: Wenn sich jemand als nicht vertrauenswürdig herausstellt, aber eine Reihe von Zertifikaten signiert hat, so werden diese Daten ebenfalls ungültig. Es ist nun sehr schwer, die Abhängigkeiten neu zu ordnen und verfallene Zertifikate aus dem Netz zu nehmen und gegen neue auszutauschen.

- *Hierarchisches Vertrauen*: Hier entstehen Vertrauensinstitutionen, deren größtes Kapital die Integrität ist. Ein solches Unternehmen hat sich aufgrund der eigenen Reputation eine Vertrauensbasis mit vielen Anwendern erworben. Es überprüft jedes Dokument, welches es signiert, und steht für dessen Richtigkeit ein. Kommt nur ein einziges Dokument in Umlauf, das keine vertrauenswürdige Daten beinhaltet und das von dieser Institution signiert wurde, ist diese Vertrauensstellung verspielt. So kann Alice ein Zertifikat einer solchen Certificate Authority (CA) erhalten und jedem Bekannten oder Fremden die eigene Authentizität belegen. Insbesondere in der internetbasierten Ökonomie ist eine solche Topologie wichtig, da nur eine zentrale Instanz Zertifikate ausgibt, validiert und ungültig erklären kann. Insbesondere kann eine Unterschrift, die von einer Person über ein solches Zertifikat ausgestellt worden ist, im Nachhinein nicht vom Signierenden bestritten werden, ein Umstand, der für Verträge von entscheidender Bedeutung ist. Eine Signatur über ein Dokument mittels eines Private Key, dessen Public Key über ein digitales Zertifikat, ausgestellt von einer CA, hinreichend beglaubigt wurde, bekundet eine vertragsgültige Willenserklärung.

Theoretisch würde eine einzige weltweite CA genügen, um ein strenges hierarchisches CA-Modell umzusetzen. In der Realität werden aber eine Reihe dieser Certificate Authorities betrieben, die mit unterschiedlichen Preismodellen und Serviceklassen den verschiedenen Anforderungen der Kunden nachkommen. So werden Zertifikate für Server ausgestellt, um deren Authentizität gegenüber ihren Kunden zu belegen, andere authentifizieren den Anwender gegenüber einem Dienst oder einem Shop sowie Benutzer untereinander. Weiterhin wird der absolute Anspruch der unbedingten Vertrauenswürdigkeit entsprechend dem Inhalt eines digitalen Zertifikats von den CAs diversifiziert. Es existieren verschiedene Klassen unterschiedlicher Vertrauensstufen:

1. *Geringe Sicherheit (E-Mail-Zertifikat)*: Zertifikate können über eine Webseite erworben werden. Die einzige Verifikation seitens der Certificate Authority ist eine gültige Mailadresse, die bei der Anmeldung angegeben werden muss und an die das Zertifikat übersendet werden kann. Ein solches Class-1-Zertifikat dient üblicherweise nur als Testzertifikat oder wird verwendet, wenn zwei Personen »mal eben« über einen sicheren Kanal über das Internet Nachrichten austauschen. Diese Zertifikate sind meist kostenlos und verfallen bereits nach kurzer Zeit.

2. *Mittlere Sicherheit*: Um ein solches Zertifikat zu bekommen, muss man der CA die Kopie eines amtlichen Ausweises (z.B. Personalausweis oder Führerschein) vorlegen. Je nach Betreiber wird man zusätzlich telefonisch kontaktiert, um die Daten zu bestätigen. Mit dieser (Class 2) Sicherheitsstufe hat man bei einer Authentisierung die Gewissheit, dass der Betreiber der CA einen gewissen Aufwand betrieben hat, um die Daten der Person, die sie zertifiziert, zu überprüfen. Einem konsequenten Fälschungsangriff hält eine solche Überprüfung der Personaldaten jedoch nicht stand.

3. *Hohe Sicherheit*: Die Person, die sich um ein Zertifikat dieser Sicherheitsstufe bemüht, muss sich persönlich bei einer Registrierungsstelle der Certificate Authority vorstellen. Die Angaben werden geprüft, bevor das Zertifikat ausgestellt wird. Ein Class-3-Zertifikat bietet die Sicherheit, dass es den Zertifikatsinhaber tatsächlich gibt, wie im Zertifikat angegeben. Um eine solche Zertifizierung zu falsifizieren, muss ein Aufwand betrieben werden, der der Fälschung eines Personalausweises entspricht.

4. *Höchste Sicherheit*: Die Certificate Authorities unterliegen einer staatlichen Kontrolle. Die Authentisierung basiert auf der Angabe von Bürgen und der Vorlage von mehreren staatlich ausgegebenen Dokumenten.

Einstufige Zertifikationshierarchie

Die einfachste Art einer hierarchischen Zertifikationsarchitektur ist diejenige, bei der eine einzige CA die Aufgabe hat, alle Interessenten mit digitalen Zertifikaten zu versehen. Die anerkannten Certificate Authorities im Internet fingen einmal mit einer einstufigen Hierarchie an, in den heutigen Ausbaustufen sind sie mittlerweile mehrstufig.

Diese einstufige Hierarchie findet man zuweilen in unternehmensinternen Netzwerken, bei denen eine firmeneigene CA Zertifikate an Mitarbeiter und Kunden ausstellt und damit eine Authentifizierung bzw. eine Verschlüsselung der Daten beim Austausch über das Internet gestattet. Eine CA, die von unserer Beispielfirma »meinefirma.com« betrieben wird, versieht die Kunden und die Mitarbeiter mit entsprechenden Zertifikaten. In diesem Fall sind die Anwender von der Richtigkeit der Daten überzeugt, da diese eine Vertrauensstellung gegenüber »meinefirma.com« haben. Natürlich kann man mit einem solchen Zertifikat keine Einkäufe im Internet tätigen, da über unseren Kunden- und Mitarbeiterstamm hinaus die CA der Firma »meinefirma.com« keine Vertrauensstellung im Internet genießt.

Mehrstufige Hierarchiemodelle

Bei einer mehrstufigen Hierarchie einer CA-Architektur signiert ein übergeordneter Dienst einen oder mehrere untergeordnete CAs. Diese wiederum stehen der Zertifizierung von Anwendern zur Verfügung. Damit ist eine Vertrauensstellung zwischen mehreren CAs auf einer Stufe geschaffen. Aber auch zwei oder mehrere CAs auf einer Stufe können sich gegenseitig vertrauen, indem sie sich über das Kreuz zertifizieren (Cross Certification).

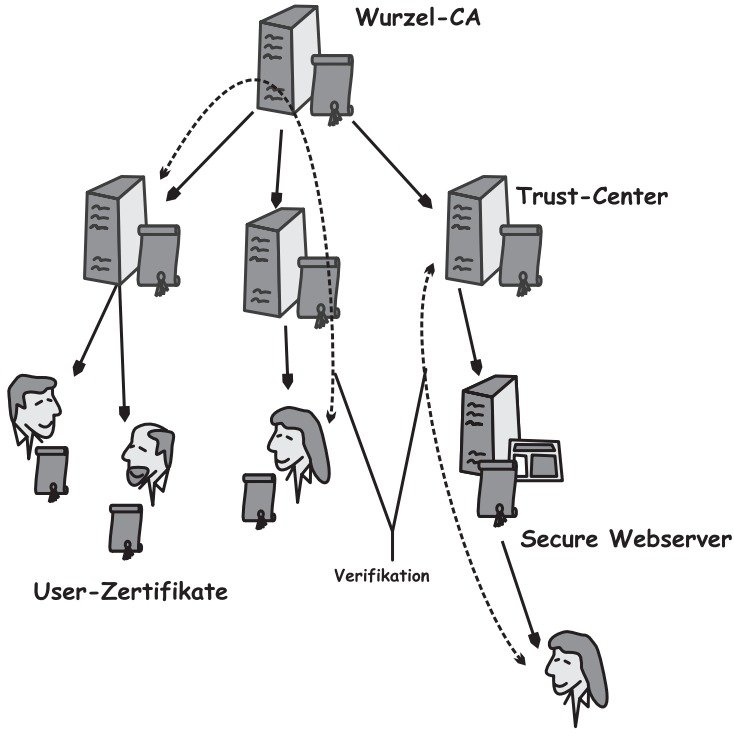


Abbildung 6.15: Mehrstufige CA-Architektur

6.10.2 Deutsches Signaturgesetz

Das Europäische Parlament verabschiedete am 19.1.2000 eine Richtlinie zur Vereinheitlichung der digitalen Signatur für den europäischen Wirtschaftsraum. In dieser Anweisung sind alle EU-Staaten verpflichtet worden, ihre nationalen rechtlichen Bestimmungen innerhalb von 18 Monaten an diese Richtlinie anzupassen. Für die deutsche Gesetzgebung bedeutet dies, dass insbesondere die strengen Zulassungsvorschriften für die so genannten Trust Center oder Zertifizierungsstellen aufgrund dieser Richtlinie deutlich erleichtert werden müssen.

Im deutschen Bundestag erkannte man die Notwendigkeit einer rechtlichen Absicherung eines Internetbasierten Handels. Mitte Februar 2001 wurde ein Gesetz beschlossen, das die Einführung digitaler Unterschriften als Dokumentation einer Willensbildung für digitale Dokumente vorsieht. Dieses Gesetz wurde schließlich am 09.03.2001 vom Bundesrat gebilligt und bildet damit die Grundlage für die Rechtsgültigkeit einer digitalen Unterschrift.

Der Maßstab für die Vertrauensanforderungen an ein staatlich anerkanntes Trust-Center ist der Passus der Haftungsregelung des Signaturgesetzes: »Verletzt ein Zertifizierungsdiensteanbieter die Anforderungen dieses Gesetzes [...] oder versagen seine Produkte [...], so hat er einem Dritten den Schaden zu ersetzen, den dieser dadurch erleidet, dass er auf die Angaben in einem [...] Zertifikat [...] vertraut.«

Grundlage der Zertifizierungsarchitektur ist der X.509-Standard, wie es bereits in einem vorangegangenen Kapitel beschrieben wurde. Dieser Standard ist in Teilbereichen so variabel gestaltet, dass er eine Reihe von verschiedenen Interpretationsmöglichkeiten zulässt. Die entsprechenden proprietären Erweiterungen in Produkten verschiedener Hersteller können dazu führen, dass verschiedene X.509-Zertifikate im Einsatz Inkompatibilitäten verursachen. Um eine einheitliche Regelung zu erwirken, hat das Bundesamt für Sicherheit in der Informationstechnik (BSI) ein eigenes X.509v3-Profil für das deutsche Signaturgesetz entwickelt, das in der SigI-Spezifikation verankert ist.

Das deutsche Signaturgesetz sieht eine zweistufige CA-Hierarchie vor, deren oberste Vertrauensinstanz von der Deutschen Regulierungsbehörde für Telekommunikation und Post (RegTP) betrieben wird und das drei weitere Trust-Center zertifiziert, von denen beispielsweise der einzelne Benutzer digitale Zertifikate erwerben kann: das Trust Center der Deutschen Telekom (Telesec), das Trust Center der Deutschen Post (Signtrust) und das Trust Center der Bundesnotarkammer.

6.10.3 Identrus

Eine weitere Form der standardisierten Zertifizierung hat sich im Bankenumfeld entwickelt. Hier werden internationale Geldinstitute mit einer Zertifizierungsarchitektur versehen, die in einer strengen Hierarchie alle beteiligten Trust-Center miteinander verbindet: Identrus.

Dieses Unternehmen wurde 1999 gegründet und geht aus einem ursprünglichen Joint Venture zwischen ABN-AMRO, Bankers Trust, Bank of America, Barclays Bank, Bayerische Hypo- und Vereinsbank, Chase Manhattan Bank, Citibank und Deutsche Bank hervor. Es sieht die Bereitstellung einer kryptografischen Vertrauensstellung basierend auf einem PKI-System für den Business-to-Business-basierten Zahlungs- und Datenverkehr vor.

Eine von Identrus betriebene CA übernimmt die Rolle der Wurzelzertifizierung. Diese Self Signed Certificate Authority gilt als oberste Vertrauensinstanz, die weiteren Trust-Center eine Zertifizierung bereitstellt. Diese wiederum können ihren Kunden digitale Zertifikate ausstellen, die unternehmensübergreifend Gültigkeit haben, da die jeweiligen 1st Level CAs die Zertifizierung durch die RootCA Identrus anerkennen.

6.10.4 Kinokarten, Geld und Internet

Bestimmte Kinos bieten bereits die Möglichkeit, über das Internet eine Sitzplatzreservierung vorzunehmen und das Ticket für den Film direkt am heimischen Computer auszudrucken. Damit erspart man sich das lästige Anstehen an der Kinokasse und kann in aller Ruhe direkt zum Film gehen. Wie aber schützt sich ein Kino vor dem Missbrauch einer solchen Einrichtung?

Betrachten wir hierfür zunächst einmal das Protokoll, mit dem man dieses Angebot umsetzen kann.

1. Eine Anwenderin möchte Karten für einen Film reservieren und besucht aus diesem Grund die Internetseite des Kinos. Sie sucht sich einen Film und den gewünschten Sitzplatz aus.
2. Auf dieser Seite übergibt sie (in verschlüsselter Form) ihre Kreditkartennummer. Ein Dritt-Unternehmen kann deren Gültigkeit überprüfen. Der Bezahlvorgang wird abgeschlossen.
3. Die Webseite des Kinos erzeugt eine hinreichend große Zufallszahl. Diese Ticketnummer wird in einer Datenbank des Kinos hinterlegt.
4. Die Webseite erzeugt ein Dokument, auf dem sich neben den Reservierungsdaten die Ticketnummer befindet.
5. Dieses Dokument wird vom Kino digital signiert. Die Kundin druckt sich dieses Ticket aus. Die Nummer des Tickets sowie die digitale Unterschrift des Kinos wird nicht nur im Klartext, sondern auch als Barcode ausgewiesen.
6. Die Kundin geht ins Kino und zeigt ihr Ticket vor. Am Eingang zum Kinosaal kann ein Kontrolleur über einen Barcode-Scanner die Ticketnummer und die digitale Unterschrift einlesen.

7. Die Ticketnummer wird in der kinoeigenen Datenbank verglichen, als gültig erkannt und mit den Film- und Reservierungsdaten abgeglichen. Sind alle Daten in korrekter Übereinstimmung, erhält die Kundin den Zutritt zum Film.

Abbildung 6.16: Internet-Kinoticket

Welche Sicherheiten haben beide Parteien? Für eine solche Analyse wechselt man am besten die Seiten, um sich zu überlegen, wie sich das System missbrauchen lässt. Will ein Betrüger das System überlisten, müsste er zunächst ein Ticket mit einer gültigen Kontrollnummer erstellen. Die Wahrscheinlichkeit für ein Gelingen dieses Unterfangens ist umgekehrt proportional zur Größe der Zufallszahl. Die Größenordnungen für eine hinreichende Sicherheit wurden in Abschnitt 6.4.2 beschrieben: Eine »Zufallszahl« aus 28 Zeichen böte eine Vielfalt von Möglichkeiten, die der Anzahl der Atome in unserer Galaxie entspräche. Das Erraten einer gültigen Ticketnummer aus einer solchen Menge ist so gut wie unmöglich. Die Ausgabe einer gültigen Ticketnummer ist einfach zu bewerkstelligen, aber nicht zu reproduzieren.

Was aber hält einen User davon ab, das ausgedruckte Ticket zu kopieren und mit der gesamten Familie ins Kino zu gehen? Alle gültigen Tickets werden mit ihrer Nummer in der Datenbank des Unternehmens vorgehalten. Das »Einlösen« des Tickets entwertet die gültige Nummer, ein Zeitstempel markiert den Zeitpunkt, wann es eingelöst wurde. Eine andere Person, die mit einer Kopie des Tickets Einlass begehrt, wird abgewiesen.

Die digitale Unterschrift entspricht der Bestätigung des Unternehmens, dass dieses Ticket von ihm ausgestellt wurde. Dadurch hat einerseits der Kunde die Beweismöglichkeit, dass das Kinounternehmen den Bezahlvorgang akzeptiert hat, andererseits hat das Unternehmen eine einfache Möglichkeit festzustellen, ob das Ticket von ihm ausgestellt wurde oder nicht.

Beide Seiten, Kunde wie Unternehmen, verlassen sich auf die Sicherheit der großen Zahlen und der schwindenden Wahrscheinlichkeit, dass weder eine gültige Ticketnummer erzeugt werden kann noch dass jemand die digitale Unterschrift des Unternehmens fälschen kann.

Die annullierten Kontrollnummern müssen nicht über einen langen Zeitraum archiviert werden. Das Ticket ist für einen bestimmten Film zu einem bestimmten Termin gültig. Einmal eingelöst kann die Ticketnummer zu einem späteren Zeitpunkt prinzipiell wieder ohne Probleme verwendet werden (die Mannigfaltigkeit aller möglichen Nummern verhindert dies aber bereits).

Abbildung 6.17: Digitaler Geldschein

Digitales Geld ist die logische Weiterentwicklung dieses Ansatzes. Ein Anwender hat bei einer Bank ein Konto mit digitalem Geld. Die Intention ist, dass man im Internet mit diesem Geld direkt und ohne Kreditkarte einkaufen kann. Dabei möchte man alle Vorteile dieses Bezahlwesens nutzen:

1. Die Bezahlung ist anerkannt. Der Kunde kann in jedem beliebigen Laden im Internet mit diesem Geld einkaufen.
2. Die Bezahlung ist sicher. Der Verkäufer ist sich sicher, dass er einen gültigen monetären Wert für seine Ware erhält.
3. Der Bezahlvorgang ist anonym: Der Einkäufer muss seine Identität beim Einkauf nicht preisgeben.

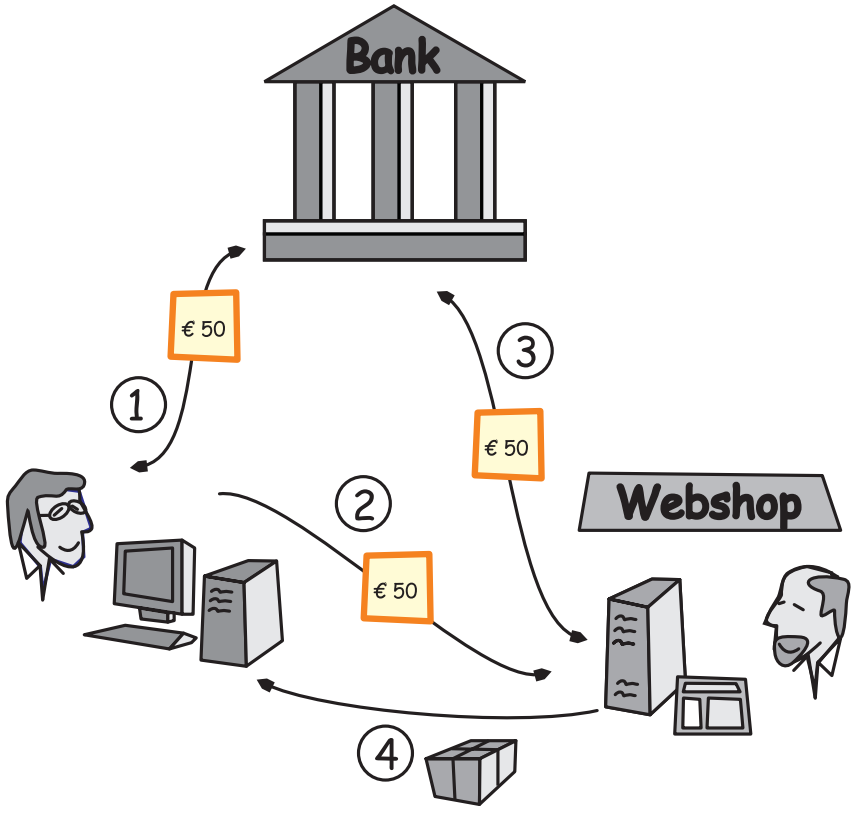


Abbildung 6.18: Ablauf des Bezahlvorgangs: 1. Beschaffung des Geldscheins, 2. Bezahlen mit dem Schein, 3. Überprüfung der Gültigkeit, 4. Aushändigen der Ware.

Zunächst muss das Geld von einer vertrauenswürdigen Instanz ausgegeben werden. Im folgenden Beispiel hat ein Kunde bei einer Bank ein Konto und er benötigt nun einen digitalen Geldschein. Sein Konto wird um einen Betrag von beispielsweise €100 belastet, für dessen Gegenwert er ein digitales Dokument erhält. Dieser digitale Geldschein kann jederzeit von der Bank über eine sichere Webverbindung abgehoben werden und liegt auf einem Speichermedium auf Seiten des Kunden.

Bei einem Bezahlvorgang wird dieses Dokument an den Händler übertragen. Dieser lässt den digitalen Geldschein bei seiner Bank prüfen. Ist die Echtheit bestätigt, wird die Ware ausgegeben, der Händler lässt sich den Geldwert auf seinem Konto gutschreiben und der Handel ist abgeschlossen.

Auch hier stellt sich die Frage, ob dieser digitale Geldschein nicht kopiert und für einen weiteren Einkauf wiederverwendet werden kann. Dies ist deshalb nicht möglich, weil die ausgebende Bank dem digitalen Geld eine Seriennummer mitgegeben hat, die zufällig und eindeutig ist.

Wird mit dem ausgegebenen »Schein« bezahlt, übergibt der Händler das Dokument an die Bank und diese überprüft die Seriennummer auf deren Gültigkeit. Ist das digitale Geld gültig, wird die Seriennummer vernichtet und der Geldbetrag dem Händler auf dessen Konto gutgeschrieben.

Dieses Verfahren ist nicht zu hundertprozentig anonym. Die Bank hat die Möglichkeit festzuhalten, an welchen ihrer Kunden sie einen digitalen Geldschein mit einer bestimmten Seriennummer ausgegeben hat. Im weiteren Verlauf kann sie den Händler identifizieren, der diesen bestimmten Schein einlöst. Auf diese Weise kann das Geldinstitut das Kaufverhalten ihrer Kunden verfolgen.

Es gibt in der Tat kryptografische Möglichkeiten, um diesem Problem zu begegnen, so dass das digitale Geld ein sicheres und anonymes Mittel des Bezahlens im Internet darstellt. Eine detaillierte Darstellung dieser Technik würde aber weit über die Zielsetzung dieses Buchs hinausgehen.

Leider ist die Akzeptanz bei den Anwendern wie bei den Händlern noch weit entfernt von einem täglichen Einsatz. Die elektronische Geldbörse wird erst dann zu einem Erfolgsmodell, wenn sich die Geldinstitute mit den virtuellen Shops auf einen Standard geeinigt haben, der den reibungslosen Ablauf des Geldverkehrs regelt.

6.11 Zusammenfassung

Ohne die moderne Verschlüsselungstechnologie hätte das Web wohl eine weit langsamere Entwicklung genommen. Erst ein sicheres Verfahren zur Datenübermittlung ermöglichte Applikationen wie Homebanking. Der gesamte Datenverkehr im heutigen elektronischen Handel wäre nicht möglich gewesen.

Der permanente Wettstreit zwischen Kryptologen und Analytikern produziert immer weitere und sicherere Verfahren zum Schutz der Daten. Die weite Verbreitung der Standardverfahren bringt ebenfalls einen sehr umfangreichen Test der Datensicherheit mit sich. Auf diese Weise ist dieser Bereich der Internetentwicklung einer stetigen Veränderung unterworfen.

7 World Wide Web

7.1 Einführung

So wie kein anderer Dienst hat das World Wide Web zum großen Erfolg des Internets beigetragen. Die mit einem solchen Dienst verbundenen Möglichkeiten beflügeln die Phantasien all jener, die sich damit vertraut machen. Daten vom anderen Ende der Welt standen plötzlich mit einem Mausklick zur Verfügung. Aber auch die Inhalte im Internet haben sich geändert. Was bis dahin aus dem Netz kam, sei es Btx oder Videotext, war textbasiert, bestenfalls aufgelockert durch ein paar einfache ASCII-Grafiken. Demgegenüber sind Webseiten fast beliebig formatierbar und gestatten das Einbinden von Grafiken und Fotos.

In den Anfängen des Webs hatte man als Anwender noch keine Suchmaschinen, über die man gezielt recherchieren konnte. Man tastete sich langsam in dieses Medium hinein, versehen mit einer immer länger werdenden Liste an Bookmarks. Erst später erleichterten Dienste wie Yahoo, Lycos oder Altavista die Orientierung.

Viele Anwender erkannten aber auch schnell, wie einfach das Schreiben und Veröffentlichen einer Webseite war. Gerade an Universitäten konnten Studenten ihre eigenen Seiten auf den Webserver der Institute hinterlegen.

Als noch keine Tools zur Erzeugung von Webseiten, beispielsweise FrontPage oder Homesite, verfügbar waren, war man entweder gezwungen, selbst eigenen *html*-Code zu schreiben (*html*: Hypertext Markup Language; hierzu später mehr) oder man kopierte andere Webseiten und passte sie den eigenen Bedürfnissen an. Auf diese Weise hatte jeder eine Möglichkeit, Teilnehmer und Anbieter in diesem globalen Netzwerk zu sein. Neue Ideen machten das Design der eigenen Webseite immer attraktiver, Counter wurden installiert, um sich an der Zugriffsfrequenz zu erfreuen, Gästebücher wurden hinterlegt, in denen man sich eintragen konnte.

Für die Wirtschaft waren die neuen Möglichkeiten noch vielversprechender. Gerade für kleinere Unternehmen oder solche, die sich auf das Web spezialisierten, stellte ein Internetauftritt das Sprungbrett in eine andere Klasse des wirtschaftlichen Erfolgs dar. Jedes Unternehmen hat die gleichen Möglichkeiten bei der Aufstellung und dem Design einer Webseite, die Erreichbarkeit für andere Anwender liegt nur einen Mausklick entfernt. So wurden neue Kunden von den alteingesessenen Unternehmen abgeworben, neue Geschäftsbereiche gegründet und neue Technologien entwickelt.

Vor diesem Hintergrund ist es interessant, sich die Grundlagen des Dienstes zu verdeutlichen, mit denen das Internet einen solch großen Erfolg innerhalb einer so kurzen Zeit begründet hat.

7.2 Architektur des Webs

Grundlage der Webtechnologie ist der Austausch von Daten und Datenbeschreibungen, die jedes beliebige Format unterstützen. So werden über Webseiten Texte, Bilder, Videos, Grafiken und vieles mehr ausgetauscht. Dabei basiert die grundlegende Struktur auf einer Dokumenten-Beschreibungssprache, der so genannten »Hypertext Markup-Language« (*html*). Daten, die in dieser Sprache geschrieben werden, haben eine intrinsische Selbstbeschreibung. So ist eine *html*-Datei zugleich Informationsträger und Darstellungsanweisung.

Daten, die in dieser *html*-Darstellung geschrieben werden, werden über das Hypertext-Transfer-Protokoll (*http*) übertragen. Sie werden von einem Browser angefordert und von einem Server geliefert.

7.3 Browser und Server

Um im World Wide Web (WWW) Informationen zu sammeln, verwendet der Anwender einen Browser. Es handelt sich dabei um eine Software, die eng mit dem betriebssystemeigenen Netzwerkprotokoll verknüpft ist. So beherrscht das Betriebssystem die Möglichkeiten, mit anderen Rechnern im Internet eine Verbindung aufzubauen und Daten auszutauschen. Die Anforderung der Daten, der Austausch von *html*-Dokumenten, das Verhandeln von Zugriffsrechten und vieles mehr, wird vom Hypertext-Transfer-Protokoll durchgeführt. Sowohl der Browser auf der einen als auch der Webserver auf der anderen Seite »sprechen« diese Sprache.

Der Anwender fordert über den Browser ein Internetdokument an, in dem er entweder einen Link auf einem bereits dargestellten Dokument selektiert, eine Bookmark-Adresse heraussucht oder im Adressbereich ein Ziel im Internet einträgt. Im Web hat die Internetadresse einen eigenen Namen, die so genannte URL (Uniform Resource Locator).

Sobald der Browser die Anweisung bekommt, eine Internetadresse zu kontaktieren, nimmt er die betreffende URL auf und zerlegt sie in die einzelnen Bestandteile. So gibt eine URL an, in welchen Bereichen der angesprochene Server zu finden ist, was zu tun ist, wenn dieser kontaktiert ist, und welches Dokument anschließend angefordert wird. (Die Struktur einer URL wird in Kapitel 13 eingehend beschrieben.)

Der Server bearbeitet die eingegangene Anforderung und stellt die Dokumente zur Verfügung, die der Client angefordert hat. Dazu gehören in den meisten Fällen ein *html*-Dokument und eine Sammlung darin enthaltener Gra-

fiken. Die im *html*-Dokument enthaltenen Formatierungsanweisungen teilen dem Browser mit, in welcher Form die Daten dem Anwender präsentiert werden sollen. Durch so genannte Tags kann der Browser zwischen Darstellungskommandos und Inhalt unterscheiden. Der Browser ist fehlertolerant realisiert: Syntaktisch nicht korrekte Anweisungen werden ignoriert.

Webseiten verlassen sich immer weniger auf Texte und Grafiken, Webprogrammierer verwenden alle Arten von Animationen wie Videos und Audiodateien. Ein Standard-Browser kann bereits mit vielen Multimediadateien umgehen, sein Funktionsumfang lässt sich aber noch zusätzlich erweitern. So hat er eine Schnittstelle, über die Fremdhersteller ihre eigenen Applikationen programmieren können, diese »Plug-Ins« werden nachträglich installiert. Darüber hinaus besitzt er eine »JavaVirtual Machine« zur Ausführung von Java-Bytecode. Dadurch können Webseiten Applikationen bereitstellen, die auf den Client übertragen und dort ausgeführt werden.

7.3.1 Formulierung von Browser-Requests

Damit ein Browser ein Dokument anfordern kann, wird ihm eine URL vermittelt. Kapitel 13 beschreibt die Struktur von solchen Verweisen, im Folgenden beschränken wir uns auf einen *http*-basierten Request.

- Der Browser erhält vom Benutzer eine Anforderung der Form

  ```
  http://<server>/<daten-selektion>
  ```

  Diese Syntax hat folgende Bedeutung:

  »http« teilt dem Browser mit, in welcher Sprache er den Server ansprechen soll.

  »<server>« definiert den Rechnernamen im Netz, der kontaktiert werden soll. Hier muss zumindest der Rechnername angegeben werden. Ist dieser nicht in der eigenen Domain erreichbar, muss der Name der Domäne angegeben werden. Der Browser geht davon aus, dass der Zielserver über den Standard-Port 80 erreichbar ist, andernfalls benötigt er explizit die Portnummer. Werden vom Server Authentifizierungsdaten für den Zugriff auf das entsprechende Dokument gefordert, so kann man diese in diesem Feld direkt übermitteln. Kapitel 13 beschreibt die entsprechenden Einzelheiten.

  »<daten-selektion>« teilt dem Server mit, welches Dokument vom Anwender angefordert wird. Dieses Feld enthält vornehmlich Pfad und Dokumentname. Eine Vielzahl weiterer Optionen dienen der Selektion der Daten und werden ebenfalls in Kapitel 13 beschrieben.

- Der Browser sendet den Request an den Server.

  Eine solche URL wird vom Browser in die Bestandteile zerlegt und umgesetzt. So wird zunächst der Servername identifiziert und dessen IP-

Adresse vom lokalen DNS aufgelöst. Der Webserver wird anschließend über *http* kontaktiert und die Anforderung auf das betreffende Dokument übermittelt.

Eine Anforderung über *http* hat folgende Syntax:

```
GET /<URI> HTTP/<VERSION>
```

Dabei teilt der Browser mit, welches Dokument gewünscht wird (<URI>) und mit welcher Version des Kommunikationsprotokolls er angesprochen werden möchte (<VERSION>). Das *http*-Protokoll lässt noch eine Reihe weiterer Felder für die Formulierung eines Request zu, die aber nicht immer verwendet werden. Allgemein lautet deren Syntax:

```
<Anforderungsart><Daten-Selektion> HTTP/<Version>
<Attributname1>:<Attribut-Wert1>
<Attributname2>:<Attribut-Wert2>
<Attributname3>:<Attribut-Wert3>...
...
<Request-Body>
```

Fordert der Anwender über eine URL ein Webdokument an, so beinhaltet dieses in der Regel weitere Dateien, die hinzugeladen werden müssen.

Beispiel:

- Ein Anwender fordert die Webseite von »Meinefirma.com« an:
  »http://www.meinefirma.com«
- Der Browser verbindet sich über *http* mit dem Webserver und fordert diese URL an.
- Der Webserver erkennt anhand der URL, dass kein spezielles Dokument angefordert wurde, und übermittelt das für diesen Fall vorgesehene Indexfile »index.html«:
- Der Browser erhält über *http* das *html*-Dokument (index.html) und durchsucht es nach weiteren Dateien, die notwendig sind, um die Webseite komplett darzustellen. (Grafiken werden in einer *html*-Datei beispielsweise über »<img src=/images/bild.gif>« eingebunden.) Die Daten des index.html-File werden vom Browser dargestellt, während die noch nicht vorhandenen Dateien erneut über separate GET-Befehle angefordert werden:

  ```
  Client: Kontaktiere <server>
  Server: Bestätige Verbindung
  Client: Fordere "/" an
  Server: Liefere index.html
  Client: Baue Verbindung ab

  Client: Parse index.html

  Client: Kontaktiere <server>
  Server: Bestätige Verbindung
  Client: Fordere "/pictures/bild.gif" an
  ```

```
Server: Liefere bild1.gif
Client: Baue Verbindung ab

Client: Kontaktiere <server>
Server: Bestätige Verbindung
Client: Fordere "/pictures/foto1.jpg" an
Server: Liefere foto1.jpg
Client: Baue Verbindung ab
```

- Jede einzelne Datei stellt einen gesonderten *http*-Zugriff dar. Der Browser sammelt die Daten und stellt sie dar.

Spezifikation der Anforderungsart

Die Anforderungsart teilt dem Server mit, was der Browser von ihm erwartet. Es kann sich hierbei um »GET« handeln, in *http 1.0* werden außerdem »POST« und »HEAD« unterstützt. *http 1.1* bietet darüber hinaus noch einige weitere Anforderungsformen an (PUT, DELETE, TRACE, CONNECT, OPTIONS), die wir in Folge vorstellen werden:

- **GET**

  Der Browser fordert vom Server ein bestimmtes Dokument an.

  Beispiel: In einer Beispielfirma sei ein Webserver installiert. Ein Client-Request ist über die URL

  »http://www.meinefirma.com/«

  erreichbar. Wird – wie hier – kein dediziertes Dokument verlangt (wie es beim ersten Kontakt eines Users mit einer Webseite üblich ist), fordert der Browser ein Index-File an. Dies wird mit dem einfachen Slash (/) angezeigt.

  Über *http* wird folgendes Kommando übertragen:

  ```
  GET /HTTP/1.0
  ```

  Gleiches gilt, wenn in einem Unterbereich der Webseite kein explizites Dokument angefordert wird:

  »http://www.meinefirma.com/marketing/«:

  ```
  GET /marketing/  HTTP/1.0
  ```

  Auch hier hat der Webserver im Verzeichnis »/marketing/« ein Index-Dokument bereitzustellen. Wird dies nicht unterstützt oder ist kein Dokument dieser Art vorhanden, listet der Webserver die im Verzeichnis vorhandenen Dokumente auf oder er reagiert mit einer Fehlermeldung.

  Unter expliziter Angabe eines Dokuments in der URL wird diese Anforderung im *http*-Protokoll übertragen.

  »http://www.meinefirma.com/marketing/statistik.html«:

  ```
  GET /marketing/statistik.html HTTP/1.0
  ```

Mit der Einführung von *http* 1.1 wurde der GET-Befehl wie alle anderen unter *http* 1.1 verwendeten Methoden um die Angabe des »host«-Attributs erweitert, die unbedingt erforderlich ist. Dies hat folgenden Grund:

Der Client löst einen Servernamen in eine IP-Adresse auf, bevor er den Webserver kontaktiert. Dies ist bei der Verwendung eines einzigen Rechnernamens für den Server unkritisch. Reagiert aber ein Webserver auf zwei oder mehrere verschiedene Namen, so geht diese Information bei der IP-Adressauflösung verloren. Dieses Problem wird mit der Angabe des »host«-Attributs im *http* 1.1-Header gelöst. Dadurch werden so genannte virtuelle Server unterstützt, deren Domainname unterschiedlich lauten kann, die aber auf eine einzige IP-Adresse zielen:

```
>telnet www.joe-user.com 80
Der Client kontaktiert 123.45.67.8

GET/HTTP/1.1
Host: www.joe-user.com

…

>telnet www.meinefirma.com 80
Der Client kontaktiert den gleichen Server 123.45.67.8

GET/HTTP/1.1
Host: www.meinefirma.com

…
```

Durch das Host-Feld erkennt der Webserver den Unterschied der Requests und liefert im ersten Fall die Homepage von Joe User und im zweiten die Geschäftsseite von meinefirma.com.

Für die Anforderung eines Dokuments sind weitere Informationen vom Client nicht notwendig. Oft werden aber automatisch noch zusätzliche Daten übermittelt, die dem Server über die Felder <AttributnameN> weitere Details über den Client-Zugriff mitteilen. Hier ein kleiner und unvollständiger Überblick, eine komplette Übersicht ist in [13] dokumentiert:

- *»User-Agent«*: Dieses Feld verrät eine Menge über den verwendeten Browser und das Betriebssystem des Clients.

  Beispiel: Bei einem Zugriff auf eine Webseite werden dem Webserver folgende Daten vom Browser über das *http*-Protokoll übermittelt:

  ```
  HTTP User Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US;
  rv:0.9.2) Gecko/20010726 Netscape6/6.1 WebWasher 3.0
  ```

  Die User-Agent-Erkennung kann die Darstellung einer Webseite unterstützen (indem man z.B. für die verwendete Browserversion angepasste Webseiten anbietet). Eine solch bereitwillige Auskunft, die von vielen Client-Programmen standardmäßig übertragen wird, ist aber den Bedürfnissen der Anwender in Sachen Privatsphäre nicht dienlich. Vielfach sind sich die User dessen nicht bewusst, wie viele ihrer Daten im Netz verteilt werden.

- »*Referer*«: Wird ein Verweis in einem Dokument verwendet, um auf eine andere Seite zu kommen, so kann das »Referer«-Feld für die Identifikation des ersten Dokuments herangezogen werden. (Der Feldname schreibt sich im *http*-Standard tatsächlich in dieser vermeintlich falschen Form.)
- »*Max-Forwards*«: Der Client gibt an, wie viele Weiterleitungen er akzeptiert, bis der Verweis als nicht erreichbar angesehen wird. Damit wird verhindert, dass eine URL1 auf URL2 zeigt, die auf URL3 weiterleitet usw. Mit einem Wert von »Max-Forwards« von beispielsweise »2« werden maximal zwei Weiterleitungen vom Client akzeptiert.
- »*If-Modified-Since*«: Dieses Feld wird vom Client verwendet, um dem Server mitzuteilen, dass er das angeforderte Dokument eigentlich bereits im lokalen Cache hat und nur dann ein Update erwartet, wenn es seit dem letzten Laden verändert wurde. Mit dieser Einstellung kann der Datenverkehr über das Netz und damit die Ladezeiten der Dokumente drastisch reduziert werden. Dieses Flag wird insbesondere von Proxyservern verwendet, die einen eigenen Cache führen und häufig angeforderte Daten lokal für die anfragenden Clients vorhalten.
- »*Authorization*«: Um dem User für jedes Dokument, das in einer geschützten Umgebung existiert, das erneute Authentifizieren zu ersparen, kann der Client das »Authorization«-Flag verwenden, das der Browser automatisch dem Request beifügt und das die User-ID/Passwort-Daten beinhaltet:

  <Authorization: Basic base64 (username:password)>

  Beispiel: Ein User loggt sich auf einer geschützten Seite über seine User-Kennung und das dazugehörige Passwort ein. Weitere Dokumente in diesem Bereich, seien es nun die Grafiken zu dem eigentlichen *html*-Dokument oder weitere Webseiten, werden in Folge mit dem »Authorization«-Flag versehen, das dem Webserver jedes Mal die Kennung übermittelt. Zur Erinnerung: Dies ist insbesondere deshalb notwendig, da der *http*-Transfer nicht persistent ist und die Verbindung nach der Übertragung eines Dokuments abgebaut wird.

Der GET-Befehl wird neben der Anforderung von Daten auch zur Datenübermittlung an den Server verwendet. Insbesondere bei *html*-Forms werden in den Eingabefeldern vom User eingetragene Daten extrahiert und an den Server zur weiteren Verarbeitung weitergereicht.

Beispiel: Ein Webserver übermittelt dem Client eine Webseite mit einem Formular, das eine Reihe von Eingabefeldern in der Maske hat. Der »Submit«-Button stößt die Übermittlung der Daten über die URL an, die eine Aktion (zum Beispiel ein CGI-Script) auf Seiten des Servers initiiert. In diesem Fall werden die Attribut/Value-Paare der Textfeldnamen mit den vom User eingegebenen Werten über die URL per

```
"http(s)://<server>c/<Verzeichnis>/
<aktion>?<Textfeld1>=<Wert1>&<Textfeld2>=<Wert2>...
```

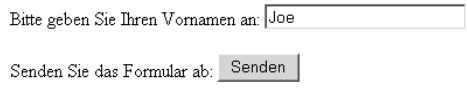
übertragen.

Hier der Beispiel-html-Code für die vom Webserver übermittelte Seite:

```
<form method="get"
action="http://www.meinefirma.com/cgi-bin/Form.pl">
<p>Bitte geben Sie Ihren Vornamen an:
<input type=text name=Vorname size=25>
<p>Senden Sie das Formular ab:
<input type=submit value="Senden">
</form>
```

Diese *html*-Sequenz stellt der Browser dem Client in etwa folgendermaßen dar:

Abbildung 7.1: html-Formular



Der User stößt mit dem »Senden«-Button die Übertragung an den Webserver an, indem der Browser einen GET-Request der folgenden Form übermittelt:

```
http://www.meinefirma.com/cgi-bin/Form.pl?name=Joe
```

Diese URL wird im URL-Feld des Browsers in dieser Form angezeigt.

Auf *http*-Protokoll-Ebene übersendet der Client an den Server folgende Daten:

```
GET /cgi-bin/Form.pl?name=Joe&submit=Submit HTTP/1.1

host: www.meinefirma.com

Referer: file:/Beispielformular1.html

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:0.9.2)
Gecko/20010726 Netscape6/6.1 WebWasher 3.0

Accept: *.*
```

Man sollte beachten, dass die Gesamtlänge der Variablen und Werte, die im Teil der URL ab »?« aufgeführt sind, in manchen Umgebungen nicht mehr als 256 Zeichen betragen kann. In diesen Fällen kann es bei der Datenübertragung zu Problemen kommen.

Abschließend ein komplettes Beispiel einer *http*-Verbindung über telnet auf Port 80:

```
>telnet www.meinefirma.com 80

GET / HTTP/1.1
host: www.meinefirma.com
```

```
Accept: text/plain
Accept: text/html
Accept: */*
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:0.9.2)
Gecko/20010726 Netscape6/6.1 WebWasher 3.0

HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
Date: Thu, 26 Jul 2001 19:11:57 GMT
Set-cookie: sessionid=4ZOABTQAAD1FBAMTA1LU45Q;path=/
…
Content-type: text/html
Content-length: 14006

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">

<HTML>
<HEAD>
<TITLE>Willkommen auf der Internetseite von Meinefirma.com</TITLE>

…
```

- **POST**

Wie beim GET-Befehl werden über POST Daten an den Server übermittelt, die weitere Aktionen initiieren. Auch hier werden wir uns in den Beispielen zunächst nur auf CGI-Scripte beschränken, der Server kann aber auch mit anderen Programmen auf diesen POST-Request reagieren. Grundlage ist wieder eine *html*-basierte Formular-Sequenz, die im Browserfenster das gleiche Aussehen wie in Abbildung 7.1 hat. Die Implementierung auf *html*-Basis sieht dagegen etwas anders aus:

```
<form method="post"
action="http://www.meinefirma.com/cgi-bin/Form.pl">
<p>Bitte geben Sie Ihren Vornamen an:
<input type=text name=Vorname size=25>
<p>Senden Sie das Formular ab:
<input type=submit value="Senden">
</form>
```

Auch hier initiiert der User mit dem »Senden«-Button die Übertragung an den Webserver, allerdings wird die Datenübertragung nicht in der URL kodiert:

```
http://www.meinefirma.com/cgi-bin/Form.pl
```

Auf *http*-Protokoll-Ebene übersendet der Client an den Server folgende Daten:

```
>telnet www.meinefirma.com 80
POST /cgi-bin/Form.pl HTTP/1.1
host: www.meinefirma.com
```

```
Referer: file:/Beispielformular2.html
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:0.9.2)
Gecko/20010726 Netscape6/6.1 WebWasher 3.0
Accept: *.*
Content-type: application/x-www-form-urlencoded
Content-length: 22
name=Joe&submit=Submit

HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
Date: Thu, 19 Nov 2001 22:10:17 GMT
Content-type: text/html
Content-length: 1321
…
```

Die Antwort des Servers hängt von der Applikation des Webservers und dessen Resultat ab. Im Unterschied zum GET-Verfahren werden die Attributnamen und die im Formular verwendeten Werte direkt über das *http*-Protokoll an den Server gesendet.

- **HEAD**

  Nicht immer wird das gesamte Dokument gewünscht, wenn man einen Server kontaktiert. Programme wie Linkchecker, die die Gültigkeit von Verweisen innerhalb eines Dokuments überprüfen, legen keinen Wert auf die Vollständigkeit des Dokuments, sondern testen lediglich die Erreichbarkeit. Der HEAD-Befehl liefert die Header-Informationen aus dem *http*-Protokoll bei Anforderung eines Dokuments, ohne das Dokument zu übersenden.

- **PUT (http 1.1)**

  Mit Hilfe der PUT-Methode kann ein Dokument auf einem Webserver ohne direkten Zugriff erzeugt oder überschrieben werden. Ein Client, der mit dieser Methode Daten überträgt, erzeugt auf dem Server eine Datei unter der referenzierten URI (Unified Resource Identifier). Eine bereits existierende Datei, die unter der gleichen URI erreichbar ist, wird überschrieben. Aus Sicherheitsgründen unterstützen die wenigsten öffentlichen Webserver diese Methode.

- **DELETE (http 1.1)**

  Diese Methode gestattet es, mit entsprechenden Zugriffsrechten den Inhalt von URI-Pointern zu löschen. Bei schlecht konfigurierten Webservern kann dies dazu führen, dass Unberechtigte Dokumente über diesen Befehl löschen können. Aus Sicherheitsgründen unterstützen die wenigsten öffentlichen Webserver diese Methode.

- **TRACE (http 1.1)**

  Mit Hilfe der TRACE-Methode kann ein Client den Durchgriff des Requests auf den Zielserver über eventuelle Proxys oder Redirections verfolgen. Der eigentliche Zielserver generiert eine Antwort. Die Trace-

Methode dient in erster Linie der Analyse und der Überprüfung der Verbindung zwischen Client und Server.

- **CONNECT (http 1.1)**

  Die CONNECT-Methode wird in erster Linie für SSL-basierte Tunnelverbindungen über Proxyserver verwendet. Ein Proxy kann den Inhalt einer verschlüsselten Sitzung zwischen Client und Server nicht auswerten. Damit er dennoch als Internetknoten verwendet werden kann, nimmt er über die CONNECT-Methode die Daten an und reicht sie durch. Ein solches Verhalten wird als »Tunnel« bezeichnet.

- **OPTIONS (http 1.1)**

  Der Client kann über einen OPTIONS-Request die verfügbaren Möglichkeiten abfragen, die ihm ein Webserver zur Kommunikation und zum Datenaustausch gestattet. Dies geschieht insbesondere, ohne dass ein Dokumententransfer oder eine Aktion auf dem Webserver initiiert wird.

  Beispiel:

```
>telnet www.meinefirma.com 80
OPTIONS * HTTP/1.1
host: www.meinefirma.com
HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
Date: Thu, 12 Dec 2001 02:13:32 GMT
Allow: HEAD, GET, PUT, POST, DELETE, TRACE, OPTIONS, MOVE, INDEX, MKDIR,
RMDIR
Content-length: 0
```

7.3.2 Server-Antwort auf Browser-Requests

Wenn der Webserver, der auf einen Client-Request angesprochen wird, aktiv ist und sich in keinem Fehlerstatus befindet, reagiert er auf den Request entweder mit dem gewünschten Dokument oder er verweigert die Herausgabe aufgrund einer serverinternen Bestimmung.

Dokumentübermittlung über http

Ein Server erhält nach einem Client-Verbindungsaufbau einen URL-basierten Request. Folgende Schritte werden durchlaufen:

Überprüfung der Autorisierung. Normalerweise werden Webdaten für den anonymen Zugriff freigegeben. In vielen kommerziellen Anwendungen sind aber Zugangs- oder Account-Angaben notwendig, um ein gewünschtes Dokument zu erhalten. Wenn in einer URL oder im *http*-Header Account-Informationen mitgeliefert werden, so werden die User-Daten auf Richtigkeit überprüft.

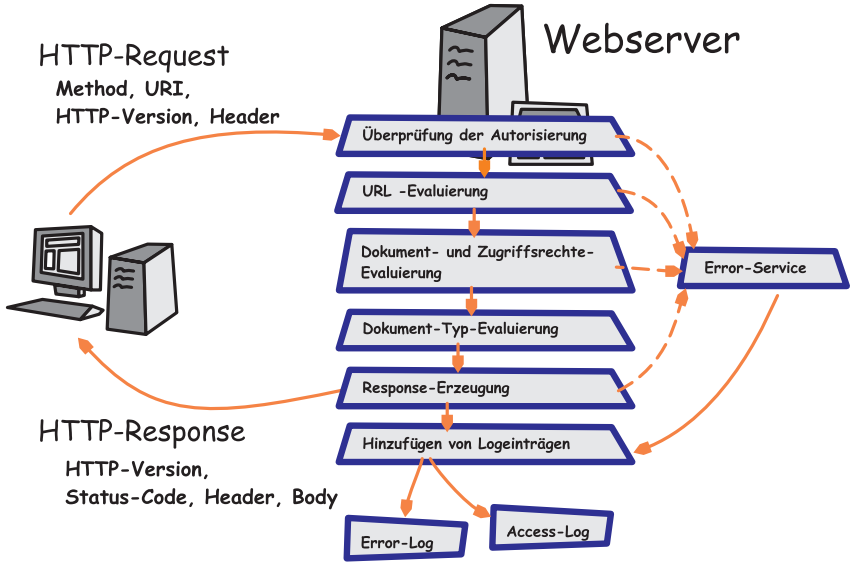


Abbildung 7.2: HTTP-Transaktionen auf dem Webserver

URL-Evaluierung. In der Phase der URL-Evaluierung identifiziert der Webserver das in der URL referenzierte Dokument in seinem Dokumentenstamm auf dem File-System.

Beispiel: Eine URL der Art »www.meinefirma.com/marketing/statistiken/2001/Produktevaluierung.html« deutet darauf hin, dass auf dem Filesystem des Webservers ein Verzeichnis der Art <webserver-documentroot>/marketing/statistiken/2001/ existiert. Oft spiegelt sich die Verzeichnisstruktur in der URL wider. Es kann aber auch sein, dass sich das Dokument in einem vollkommen anderen Bereich des Filesystems befindet, der nichts mit dem in der URL suggerierten Pfad zu tun hat (beispielsweise unter »/opt/Webdokumente/marketing/«).

Evaluierung der Dokument- und Zugriffsrechte. Die Zugriffsrechte auf ein Dokument werden zusammen mit den Daten aus der Überprüfung der Autorisierung ermittelt. Die in der ersten Phase einer *http*-Transaktion erhaltenen und verifizierten Account-Informationen werden nun mit den Zugriffsrechten auf das Dokument abgeglichen.

In den meisten Fällen ist der Zugriff auf ein Webdokument ohne Authentifizierung möglich. Bei einer geschützten Seite greift der Client zunächst ebenfalls ohne Account-Daten zu. Die Phase »Überprüfung der Autorisierung« liefert keine Werte, so dass User-ID und Passwort dem Server zunächst nicht übermittelt werden. Die Phase »Evaluierung der Dokument- und Zugriffsrechte« verwehrt hier nun den Zugriff auf die geforderte Datei. Wenn der Client diese Ablehnung erhält, fordert er über ein Popup-Fenster den Anwender auf, User-Namen und Passwort anzugeben, deren Daten er dann in einem neuen *http*-Request an den Server übermittelt.

Dokumenttyp-Evaluierung. In dieser Phase wird der MIME-Typ des angeforderten Dokuments ermittelt. Das Ergebnis wird im Attribut »content-type« im *http*-Protokoll dem Client mitgeteilt (text/html, image/gif, ...).

Response-Erzeugung. Mit der Identifizierung des gewünschten Dokuments und der Validierung der Zugriffsrechte wird zusammen mit dem Content-Typ eine *http*-basierte Datenübertragung formuliert und an den Client übermittelt.

Hinzufügen von Logeinträgen. Je nach Konfiguration des Servers werden Logdateien geschrieben. Diese können mehr oder weniger umfangreich gestaltet sein. Es wird normalerweise jeder Zugriff zusammen mit dem Status der Übertragung und den Clientdaten (IP-Adresse, gegebenenfalls User-ID, Zeitstempel, Browsertyp, ...) protokolliert.

Error-Service. Diese Phase wird nur dann durchlaufen, wenn in einem der vorangegangenen Schritte ein Fehler aufgetreten ist. Es wird ein Eintrag in eine Logdatei geschrieben und der Server reagiert statt mit dem gewünschten Dokument mit einer Fehlermeldung.

Ein Webserver arbeitet einen Request auf ein Dokument nach einem bestimmten Schema ab:

- Überprüfung der Autorisierung
- Evaluierung der Dokument- und Zugriffsrechte
- Evaluierung des Dokumenttyps
- Erzeugung des Response
- Hinzufügung von Logeinträgen
- Error-Service bei Fehlern

Übersicht über die Statuscodes eines Webservers

Wie die meisten anderen Server auch, so übermittelt ein Webserver mit der Beantwortung eines Request einen Statuscode. Diese sind hierarchisch organisiert und informieren den Client über den Status der Transaktion. Hier eine Übersicht über die geläufigsten Codes und deren Verwendung:

1xx. Die Gruppe der Statuscodes, die mit einer Eins beginnen, sind informell und werden normalerweise nicht verwendet.

2xx. Die Gruppe der Statuscodes, die mit einer Zwei beginnen, quittieren einen erfolgreichen Datenübertrag. Normalerweise wird nur der Code 200 verwendet:

- 200 OK

  Der Server bescheinigt eine geglückte Datenübertragung.

3xx. Die Gruppe der Statuscodes, die mit einer Drei beginnen, informieren den Client darüber, dass sich das angeforderte Dokument auf einem anderen Server befindet:

- 301 Moved permanently

  Das Dokument ist dauerhaft auf eine andere URL verlegt worden. Der Client möge sich dieses Dokument von dieser neuen URL laden. Alle Verweise auf diese Resource sollten aktualisiert werden.

- 302 Moved temporarily

  Das Dokument ist temporär auf eine andere URL verlegt worden. Der Client möge sich dieses Dokument von dieser neuen URL laden. Die Verweise auf die neue URL sollten aber nicht geändert werden, da abzusehen ist, dass sich dieser Verweis erneut ändern wird.

- 304 Not modified

  Dieser Statuscode wird in Zusammenhang mit der Verwendung des »If-Modified-Since« im Request durch den Client eingesetzt. Wenn der Server keine Änderung des Dokuments seit dem vom Client übermittelten Datum feststellt, beantwortet er diesen Request mit dieser Statusmeldung. Gerade in Zusammenhang mit Cache-Vergleichen durch Browser oder durch Proxyserver erspart man sich mit einer solchen Meldung einen zeitaufwändigen Download.

4xx. Die Gruppe der Statuscodes, die mit einer Vier beginnen, signalisieren dem Client einen Fehler, der aufgrund eines nicht zulässigen Request erzeugt wird.

- 400 Bad request

  Der vom Client gesendete Request entspricht nicht einem Format, das der Server versteht.

- 401 Unauthorized

  Der Client ist nicht berechtigt, die angeforderte Ressource zu bekommen. Dies kann eventuell über eine gültige Authentisierung behoben werden.

- 403 Forbidden

  Der Client ist nicht berechtigt, die angeforderte Ressource zu bekommen. Dies kann auch nicht über eine Authentisierung behoben werden.

- 404 Not found

  Der Server kann die vom Client angeforderte Ressource nicht finden. Dies kann entweder durch eine syntaktisch richtige, aber semantisch falsch formulierte URL hervorgerufen werden, oder aber ein Dokument verweist auf ein anderes, das nicht mehr existiert.

5xx. Die Gruppe der Statuscodes, die mit einer Fünf beginnen, signalisieren dem Client, dass der Server ein technisches Problem hat und auf den Request nicht wie gewünscht reagieren kann.

- 500 Internal error

  Der Server selbst hat ein technisches Problem, das ihn daran hindert, den Request des Client wie erwartet zu beantworten.

- 501 Not implemented

  Die vom Client verwendete Methode im Header des Requests wird vom Server nicht unterstützt. Wird beispielsweise die DELETE-Methode aus dem Umfang der Services des Servers entfernt, kann ein Delete-Request von einem Client nicht ausgeführt werden.

- 503 Service unavailable

  Diese Statusmeldung ist für den Fall vorgesehen, dass der Server überlastet ist und den Request derzeit nicht bearbeiten kann. Der Client kann die Anforderung des Dokuments zu einem späteren Zeitpunkt wiederholen.

Statuscodes informieren den Client über den Zustand des Servers. Sie sind in verschiedene Klassen unterteilt, die hierarchisch geordnet sind. Über die Nummer des Statuscodes kann ein Client die Fehlerklasse identifizieren und entsprechend weitere Aktivitäten einleiten.

7.4 Webserver-Administration

Die auf dem Markt angebotenen Webserver bieten unterschiedliche Formen der Administration an. Die einfachste Art ist die Konfiguration über eine Konfigurationsdatei, die alle für den Start und den Betrieb des Servers notwendigen Parameter enthält. Andere Server bieten grafische Oberflächen an, mit denen die Administration vereinfacht werden soll.

Für die Installation und den Betrieb des Servers müssen verschiedene grundsätzliche Fragestellungen geklärt werden, die in der Basiskonfiguration umgesetzt werden.

7.4.1 Basiskonfigurationen

Ein Webserver wird mit einigen wenigen Installationsschritten zu einer Publikationsplattform für Webdokumente. Damit er von »außen« erreichbar wird, sollten unterschiedliche Konfigurationen den Internetstandards entsprechen.

- Port: Von einem Webserver wird angenommen, dass er über einen Standard-Port erreichbar ist. Gerade weil alle Browser von der Annahme aus-

gehen, dass eine Webseite über den Port 80 (oder bei einer sicheren Verbindung über Port 443) erreichbar ist, wird bei der Anforderung über eine URL auf dessen Angabe verzichtet. Natürlich kann ein Webserver auf jedem beliebigen freien Port installiert werden, allerdings steht der Anwender dann von der Aufgabe, den Server und die darauf hinterlegten Seiten zu finden. Eine Webseite auf einem Server, der nicht auf den Standard-Ports läuft, ist eigentlich nur noch über einen direkten Link zu erreichen.

- Im Internet wird für die meisten Webseiten angenommen, dass der betreffende Server auf den Hostnamen »www« reagiert. Eine Firmenpräsentation wird nur dann wirklich wahrgenommen, wenn sie über die URL »www.<firmenname>.com« bzw. eine entsprechende Länderkennung für die Toplevel-Domains erreichbar ist. Natürlich kann ein Webserver auch unter jedem anderen Hostnamen eingerichtet werden, allerdings muss der Anwender dann diesen Namen erraten.
- Von einem Webserver wird erwartet, dass er bei der Anforderung über eine URL wie www.<firmenname>.com ohne Angabe eines bestimmten Dokuments eine Seite aufbaut, die so genannte Index-Seite. Somit sind die beiden folgenden URLs gleichbedeutend:

  ```
  http://www.<firmenname>.com
  http://www.<firmenname>.com/index.html
  ```

- Ein Webserver kann Verzeichnisse direkt über ein Indexfile bedienen oder ein Directory-Listing präsentieren. (Details werden im nächsten Abschnitt besprochen, ein Directory-Listing ist in Abbildung 7.13 dargestellt.) Im Sinne einer sicheren Webpräsentation ist ein Index-File einem Listing vorzuziehen.
- Services, die auf einem Port kleiner 1024 gestartet werden, bedürfen einer Root-Berechtigung. Ein Webserver hat damit alle Berechtigungen, die der Root-Account besitzt. Dies ist insbesondere dann kritisch, wenn unsichere Applikationen per CGI (Common Gateway Interface; siehe Abschnitt 7.5.3) über den Server ausgeführt werden können. Um dies zu verhindern, sollte ein Webserver unter einer User-Berechtigung laufen, die weit weniger Rechte auf dem Betriebssystem besitzt. Das Starten des Webdienstes geschieht dann über den Root-Account, anschließend aber wird der laufende Service dem gesonderten User-Account übergeben.
- Für einen Server, der über SSL erreichbar sein soll, muss ein digitales Zertifikat von einer CA angefordert werden, das nach Erhalt beim Server eingespielt wird.
- Ist ein Webdokument auf von einer ursprünglichen Seite zu einem anderen Webserver übertragen worden, so kann eine URL-Weiterleitung konfiguriert werden, die beim Zugriff auf die ursprüngliche Seite automatisch aufgerufen wird.

- Tritt beim Abwickeln eines Client-Request an irgendeiner Stelle des Ablaufs ein Fehler auf, erscheint statt des angeforderten Dokuments eine Fehlermeldung in Form von *html*-Seiten. Solche »Error-Pages« sind auf dem Webserver konfigurierbar und können modifiziert werden. Eine angeforderte Webseite, die nicht auf dem Server hinterlegt ist, kann mit einer angepassten Webseite beantwortet werden, die über die simple »FILE NOT FOUND«-Anzeige hinausgeht (siehe Abbildung 7.3).

7.4.2 Virtuelles Hosting

Der Webserver einer großen Firma wird häufig besucht und ist ausreichend damit beschäftigt, die eigene Domain zu betreiben. Dies wird bei kleineren Firmen anders sein, deren Seiten nicht so häufig besucht werden und bei denen der Serverprozess öfter im Leerlauf ist. Hier bietet es sich an, dass eine weitere Domain über den gleichen Dienst betrieben wird. Als Beispiel soll wieder die Firma »Meinefirma.com« dienen. Einer der Mitarbeiter, Joe User, möchte seine eigene Domain betreiben, indem er den Firmenserver mitbenutzt.

Hier bieten sich verschiedene Möglichkeiten an, die im Folgenden beschrieben werden sollen.

Virtuelle Server auf einer IP-Adresse

Ein Serverprozess ist an eine bestimmte IP-Adresse gebunden. In diesem Fall hat der Host des Webservers eine einzige IP-Adresse und der Webserver-Prozess (*http*-Daemon) lauscht auf eingehende Client-Requests, die an diese Adresse auf den Webserver-Port gerichtet sind. Im Beispiel hat der Webserver die IP-Adresse 123.45.67.8, die Domain-Name-Auflösung der Domäne »www.meinefirma.com« zeigt auf diese IP-Adresse und der *http*-Daemon ist auf Port 80 konfiguriert. Die DNS-Konfiguration sieht nun vor, dass die DNS-Auflösung von »www.joe-user.com« auf die gleiche IP-Adresse zielt. Dazu meldet Joe User seine Domain beim DENIC (Deutsches Network Information Center) an und gibt diese IP-Adresse als die für seine Domain zuständige Adresse an.

Damit werden weltweit alle Anfragen an »www.joe-user.com« auf den dafür bestimmten Webserver gelenkt.

Eine solche Form des Server-Hosting ließ sich unter der Vorgängerversion von *http* 1.1 nicht einrichten. Wie wir bereits gesehen haben, wird die URL-Namensauflösung vom Client aus betrieben. Im weiteren Verlauf der *http*-Kommunikation zwischen Client und Server tritt der Zielserver-Part der URL nicht mehr auf. Dies ist der Grund, warum man in *http* 1.1 im Header des Protokolls zusätzlich das Host-Attribut angibt: Hier wird noch einmal explizit die Domain benannt, die man erreichen möchte.

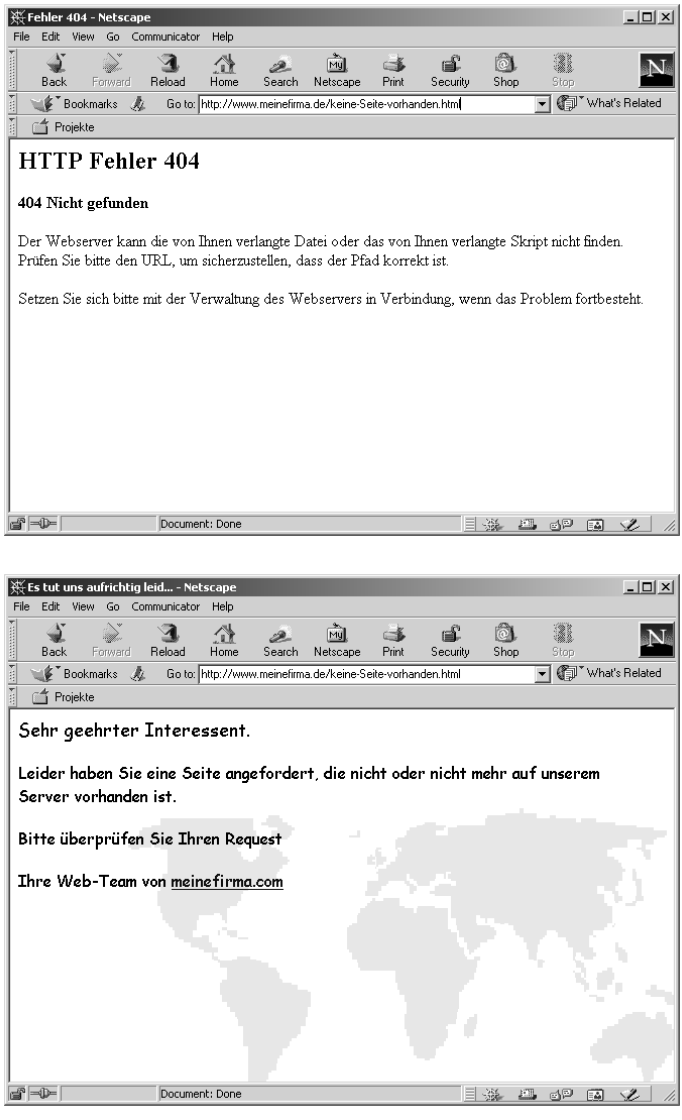


Abbildung 7.3: Standard-Fehlermeldung (oben) und eine angepasste Error-Page (unten) eines Webservers

Beispiel:

Der Anwender besucht die Webseite der Firma und fordert diese über die URL »www.meinefirma.com« an. Die Namensauflösung des DNS liefert für den Webserver die IP-Adresse 123.45.67.8, entsprechend kontaktiert der Browser diesen Host. Anschließend wäre die Domänen-Information verloren, wäre im *http*-Header nicht noch die gewünschte Domain explizit angegeben:

```
GET / HTTP/1.1
Host: www.meinefirma.com
```

Ein anderer Anwender möchte die Webseite von Joe User kontaktieren, das DNS-System liefert wieder die IP-Adresse 123.45.67.8, im Header steht aber explizit die verlangte Domain:

```
GET / HTTP/1.1
Host: www.joe-user.com
```

Ein *http* 1.1-fähiger Webserver wertet diese Informationen aus, entsprechend der angeforderten Domain »weiß« er, dass das Dokument-Root-Verzeichnis der Domain »www.meinefirma.com« im Beispiel unter »/opt/server/docs/meinefirmacom/« auf dem Filesystem zu finden ist, das der Domain von Joe User dagegen unter »/opt/server/docs/joeuser/«.

Über diese Konfiguration werden damit zwei Domänen mit einem einzigen Webserver auf dem Standardport 80 betrieben.

Diese Form der Serverkonfiguration wird manchmal auch als »Software Virtual Server« bezeichnet.

Zusammenfassung des Beispiels von mehreren virtuellen Servern auf einer IP-Adresse

Ressourcen:

Host:	woodstock.meinefirma.com
IP-Adresse:	123.45.67.8

Konfiguration:

www.meinefirma.com:	/meinefirmacom/index.html
DNS:	123.45.67.8
Port:	80
www.joe-user.com:	/joeuser/index.html
DNS:	123.45.67.8
Port:	80

Ergebnis:

Ein Serverprozess:	123.45.67.8:80
1. Document-Root:	...docs/meinefirmacom/index.html
Startseite:	www.meinefirma.com/index.html
2. Document-Root	...docs/joeuser/index.html
Startseite:	www.joe-user.com/index.html

Mehrere Webserverprozesse auf einer IP-Adresse

Hier werden verschiedene Serverprozesse auf einer Maschine betrieben. Diese Konfiguration hat aber das Problem, dass ein Rechner, der nur mit einer einzigen IP-Adresse versehen ist und dessen Webserver-Prozess auf dem Standard-Port 80 betrieben wird, keinen weiteren Serverprozess zulässt, der auf dem gleichen Port lauschen soll. Damit wäre Joe User gezwungen, seinen Dienst auf einem anderen Serverport einzurichten, (beispielsweise Port 81), was zur Folge hätte, dass seine Domain immer die Angabe des Ports erzwingt: http://www.joe-user.com:81. Da ein Standardzugriff auf eine Webseite normalerweise nicht mit einer Portangabe versehen ist, ist nicht zu erwarten, dass die Webseite von Joe User häufig gefunden wird.

Zusammenfassung des Beispiels von mehreren virtuellen Servern auf einer IP-Adresse

Ressourcen:

Host:	woodstock.meinefirma.com
IP-Adresse:	123.45.67.8

Konfiguration:

www.meinefirma.com:	/meinefirmacom/index.html
DNS:	123.45.67.8
Port:	80
www.joe-user.com:	/joeuser/index.html
DNS	123.45.67.8
Port:	81

Ergebnis:

1. Serverprozess:	123.45.67.8:80
2. Serverprozess:	123.45.67.8:81
1. Document-Root:	...docs1/meinefirmacom/index.html
1. Startseite:	www.meinefirma.com/index.html
2. Document-Root:	...docs2/joeuser/index.html
2. Startseite:	www.joe-user.com:81/index.html

Virtuelle Server auf mehreren IP-Adressen

Entgegen dem Beispiel aus dem vorangegangenen Abschnitt kann ein einzelner Rechner durchaus zwei Serverprozesse auf dem Standard-Port betreiben, allerdings nur unter der Voraussetzung, dass für jeden Dienst eine eigene IP-Adresse konfiguriert ist. Der einzelne Serverprozess wird an die jeweilige IP-Adresse gebunden, die verschiedenen Prozesse existieren nebeneinander jeweils auf ihrem eigenen Standardport 80. Für die DNS-

Konfiguration bedeutet dies, dass die Namensauflösung für www.meinefirma.com auf eine andere IP-Adresse zielt als für www.joeuser.com. Dies ist bei der Anmeldung von beiden Domains bei DENIC zu berücksichtigen. Die Client-Zugriffe werden auf die jeweils dafür vorgesehenen IP-Adressen auf den gleichen Rechner geleitet und der jeweils dafür vorgesehene Serverprozess nimmt die Bearbeitung des jeweiligen Request entgegen.

Diese Form von verschiedenen virtuellen Servern bezeichnet man auch als »Hardware Virtual Server«.

Zusammenfassung des Beispiels von mehreren virtuellen Servern auf unterschiedlichen IP-Adressen

Ressourcen:

Host:	woodstock.meinefirma.com
1. IP-Adresse:	123.45.67.8
2. IP-Adresse:	123.45.67.9

Konfiguration:

www.meinefirma.com:	/meinefirmacom/index.html
DNS:	123.45.67.8
Port:	80
www.joe-user.com:	/joeuser/index.html
DNS	123.45.67.9
Port:	80

Ergebnis:

1. Serverprozess:	123.45.67.8
2. Serverprozess:	123.45.67.9
1. Document-Root:	...docs1/meinefirmacom/index.html
1. Startseite:	www.meinefirma.com/index.html
2. Document-Root	...docs2/joeuser/index.html
2. Startseite:	www.joe-user.com/index.html

Wie man sieht, können die Ressourcen eines Webservers geteilt werden und unterschiedliche Domänen müssen nicht unbedingt mit verschiedenen Servern bedient werden. Die Aufteilung eines Servers auf unterschiedliche Domänen ist insbesondere dann sinnvoll, wenn abzusehen ist, dass eine Domain allein den Server nicht auslastet.

7.5 Content-Management

7.5.1 Webseiten-Erstellung

Einführung

Webseiten können auf vielfältige Weise erzeugt werden. Ziel ist es, einen Quelltext in *html* zu generieren, der auf dem Webserver hinterlegt wird. Auf welche Weise dies geschieht, ist unerheblich. So kann man mit einem einfachen Texteditor jede Webseite programmieren, wenn man sich einmal mit der *html*-Syntax vertraut gemacht hat. Gerade dies ist von vielen Programmierern die einzig sinnvolle Art, *html*-Dokumente zu erzeugen.

Aber nicht jeder Anwender ist ein Programmierer. Gerade die User, die ihre eigene Homepage erstellen möchten, sind nicht unbedingt bereit, Programmieraufwand zu leisten oder sich diese Fähigkeiten anzueignen. Für diesen Markt sind Webeditoren entwickelt worden, die mit einem grafischen Interface und einem Toolset ausgerüstet sind. Damit lassen sich mit einigen wenigen Handgriffen durchaus brauchbare Ergebnisse erzeugen. Diese Editoren liefern nach Fertigstellung der Seite den Quelltext, der anschließend auf dem Webserver publiziert wird.

Welche der beiden Methoden für die Entwicklung angewandt wird, ist eine Frage des persönlichen Geschmacks. Der Vorteil der Programmierung über einen Texteditor ist die Kontrolle über den Source-Code. Der Programmierer kennt den Quelltext und kann ihn seinen Bedürfnissen entsprechend optimieren. Dagegen spricht allerdings die längere Entwicklung der Seite: Während bei Entwicklungstools vorgefertigte *html*-Bausteine vorhanden sind, die man direkt verwenden kann, muss bei der Entwicklung über den Texteditor jeder einzelne Befehl direkt eingegeben werden.

Die grafischen Editoren werden auch als WYSIWYG-Tools bezeichnet, ein Akronym, das für »What You See Is What You Get« steht. So wie der Entwickler eine Seite mit den grafischen Tools aufbaut, so (oder so ähnlich) wird die Seite auf dem Webserver hinterlegt und anschließend im Browser angezeigt. Der Nachteil eines solchen Vorgehens ist, dass der Entwickler kaum mehr Einfluss auf den eigentlichen Quelltext hat. Ob diese Seite nun sinnvoll »programmiert« ist, liegt in der Fähigkeit des Editors. Und diese sind nicht immer in der Lage, ausschließlich sinnvollen und performanten *html*-Code zu erzeugen.

Eine interessante Alternative bieten Editoren, die von beiden Philosophien gleichermaßen profitieren, das heißt, dass ein WYSIWYG-Editor gleichzeitig erlaubt, den Quelltext zu bearbeiten, und umgekehrt mit Tools für Quelltextbausteine aufwartet.

Statische Seiten

Die weitaus meisten Webseiten sind in statischem *html* geschrieben. Wir werden uns hier auf eine sehr kurze Einführung in diese Programmiersprache beschränken. In der Literatur gibt es eine Fülle von Büchern mit sehr detaillierten Beschreibungen zu diesem Thema.

html, die »Hypertext Markup Language«, ist eine Scriptsprache, mit der eine Webseite aufgebaut wird. Ein solches Dokument enthält den Inhalt zusammen mit der Anweisung, wie dieser darzustellen ist.

Beispiel:

```
<b>Hier ist eine Text, fett dargestellt.</b>

<font face="Helvetica, Arial, sans-serif">Dieser Text ist in einem
speziellen Schriftsatz verfasst</font>
```

Zur Visualisierung ist ein so genannter *html*-Browser notwendig, der die Darstellungsanweisungen interpretieren und umsetzen kann.

Statische Seiten sind Dokumente, die unabhängig davon, welcher Client zu welcher Zeit zugreift, immer den gleichen Inhalt liefern. Seitens des Browsers bedeutet dies, dass keine weiteren Applikationen wie Applets oder ActiveX-Komponenten gestartet werden. Der Inhalt der Seite kommt immer von einem bestimmten unveränderten Dokument des Webservers. Statische Webseiten sind daher meist nur informell, das heißt, es werden Informationen dargeboten, der Client interagiert aber nicht mit dem Server.

Ein *html*-Dokument besteht aus Informationen und Darstellungsanweisungen. Diese beiden Datentypen unterscheiden sich durch die Darstellung: Anweisungen über die Präsentation werden über so genannte »Tags« markiert, während die reine Information unmarkiert ist.

Ein Browser interpretiert die Tags und stellt die Informationen entsprechend dar. Sind fehlerhafte oder für den Browser unbekannte Tags vorhanden, so werden diese fehlertolerant entweder ignoriert oder als Information dargestellt.

Eine *html*-Seite hat einen bestimmten Aufbau, der sich in einen Kopfteil (»Header«) und den Inhalt (»Body«) untergliedert. Im Header werden Zusatzinformationen wie Titel und Metadaten zu dem Dokument untergebracht, im Body-Teil ist der eigentliche Inhalt hinterlegt.

Beispiel:

```
<html>
 <head>
  <title>Beispiel</title>
 </head>
 <body>
  <h1>Beispiel einer Webseite</h1>
```

```
  <br>
  Hier ein Beispiel einer Webseite
  <br>
 </body>
</html>
```

Jedes Buch über *html* bietet einen Einblick in die weitere Programmierung von Webseiten, über die man sich tiefer in die Programmierung mit dieser Scriptsprache einarbeiten kann.

Dynamische Seiten

Die Interaktion des Clients mit dem Server zeichnet dynamische Seiten aus. Dabei übergibt der Client Daten an den Server, der darauf individuell reagiert. Wird ein Webformular von einem Anwender ausgefüllt und abgeschickt, übernimmt der Server die Eingaben und erzeugt darauf basierend einen individuellen Response. Damit gehen dynamische Seiten weit über den reinen informellen Charakter von statischen *html*-Seiten hinaus und gestatten eine individuelle Interaktion zwischen Anwendern und Diensten. So werden Homebanking-Applikationen möglich oder man kann über Webshops Einkäufe tätigen. Für die Umsetzung von dynamischen Seiten existiert eine Palette von verschiedenen Architekturen, (vgl. Abschnitt 7.5.3; eine eingehende Betrachtung würde allerdings weit über den Fokus dieses Buchs hinausgehen).

7.5.2 Webseiten und Business-Logik

Mit der Integration von Applikationen in Webseiten strebt man eine Wechselwirkung zwischen Interessenten und Webseitenanbietern an, die über den rein informellen Charakter einer statischen *html*-Seite hinausgeht. So werden Applikationen entwickelt, die den Zugriff auf Datenbanken oder Funktionen gestatten, die auf Seiten des Servers liegen und die eine Dienstleistung gegenüber dem Client darstellen. Im Laufe der Zeit entstanden Homebanking-Applikationen, Aktiendepots, Börsen-Charts, Online-Shops und vieles mehr. Natürlich durchlaufen diese Seiten eine gewisse Entwicklung. Beginnend mit der ersten Präsentation wechselt die Optik sowie die Applikationsfunktionalität und passt sich stetig den Wünschen der Kunden, Interessenten, aber auch der Firmenphilosophie an. Neue Entwürfe des Designs wie des Service-Umfangs erfordern eine stetige Anpassung an die Vorgaben von Marketing wie an Kundenwünsche und erzwingen eine permanente Weiterentwicklung der Webpräsentation.

Um diesen Anforderungen gerecht zu werden, entwickelte man im Laufe der Zeit mehrere Strategien zur Einbettung von Applikationen in den Kontext von statischen *html*-Seiten. Mit Architekturen wie Serverside JavaScript oder Active Server Pages konnten Webseitenentwickler diese Services implementieren und innerhalb kurzer Zeit Funktionalitäten anbieten, die weit über die Möglichkeiten einer reinen *html*-Implementierung hinausgingen.

Ein klassisches Beispiel sind Datenbankanbindungen, die innerhalb einer *html*-Umgebung initiiert wurden, Abfragen formulierten und Ergebnismengen ausgaben. Dafür wurden Script-Inkludierungen innerhalb der *html*-Seiten Funktionalitäten programmiert, die in ihrem eigenen Kontext definiert sind. Durch spezielle Tags sowie die Anmeldung der Seite als eine »aktive« Seite wird dem Webserver mitgeteilt, dass nach dem Request der Seite nicht einfach nur der Inhalt »geliefert« wird, sondern dass die Seite vorher vom Server untersucht (»geparst«) wird und die eingebetteten Funktionen ausgeführt werden. Deren Ergebnis wird zusammen mit dem *html*-Gerüst als Ergebnis an den Client übermittelt, dabei handelt es sich nur noch um Code, den der Browser am anderen Ende interpretieren und darstellen kann, also *html*-Direktiven, JavaScript und Grafiken.

Diese Form der Service-Anbindung schaffte einen großen Freiraum für webbasierte Dienste, hatte man doch jetzt ganz andere Voraussetzungen für die Einbindung von Diensten, die man über das Web publizieren konnte.

Viele Entwickler banden nun Applikationen in den *html*-Kontext ein, indem sie die Services direkt in die *html*-Dateien integrierten. Dieses Verfahren liefert sicherlich schnelle Ergebnisse, birgt aber einen entscheidenden Nachteil. Werden die Applikationen umfangreicher oder die Webseiten umstrukturiert, müssen nicht selten die Applikationen umprogrammiert, im schlechtesten Fall neu geschrieben werden.

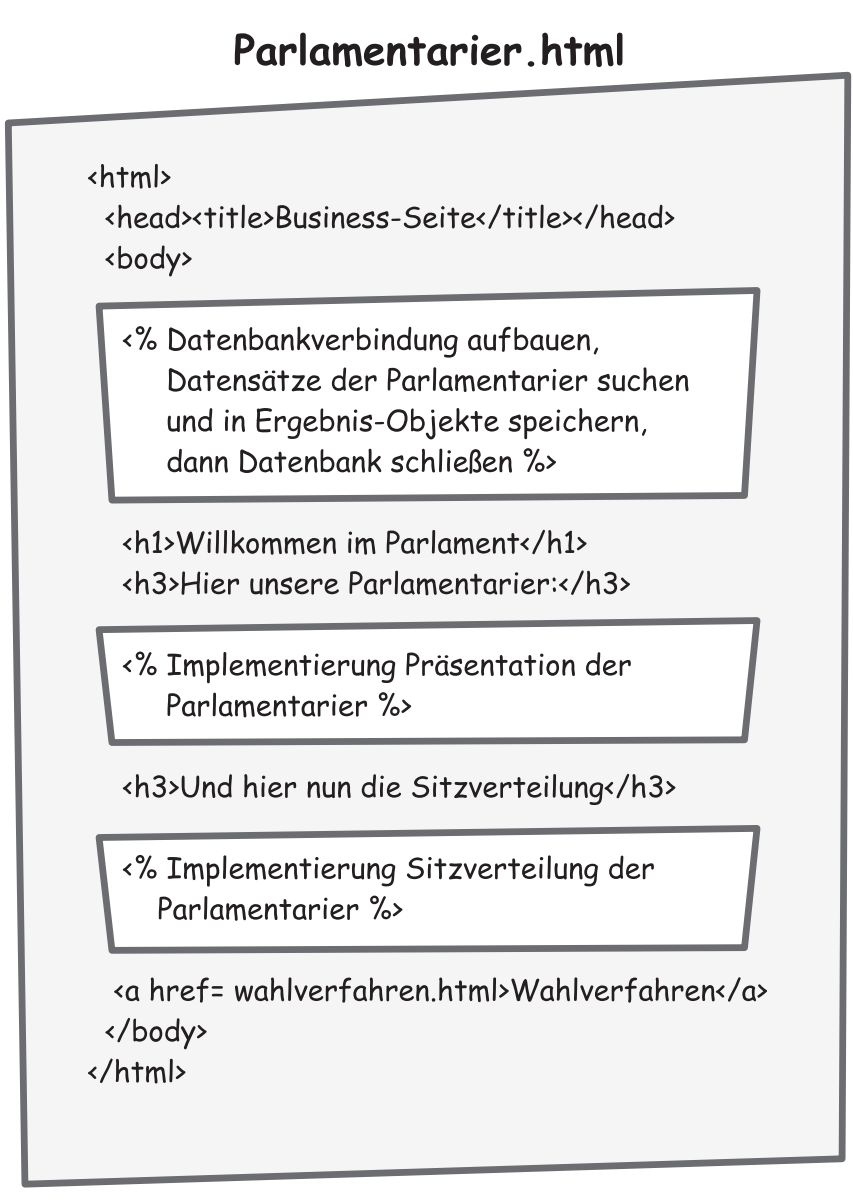
Schlimmer noch, sind die eigentlichen Entwickler der Seite nicht mehr im Unternehmen und müssen andere deren Aufgabe wahrnehmen, so ist es sehr schwierig, die Applikations- von der Präsentationslogik zu trennen und eine Weiterentwicklung vorzunehmen. Mit anderen Worten: Eine solche Lösung wächst nicht mit den Anforderungen.

Eine Implementierung nach dem Modell der Vermischung von Präsentationsschicht und Applikationen wird als »Page Centric Model« bezeichnet.

Beispiel:

Ein Parlament möchte eine Webseite publizieren, auf der alle Abgeordnete einschließlich Wahlkreis, Mandat, Lebenslauf und Wahlergebnis präsent sind. Die dafür vorgesehenen Funktionen zum Abrufen der Daten aus der Datenbank werden im ersten Ansatz in die Webseite hineinprogrammiert. Nach relativ kurzer Zeit kann der Webseitenentwickler schon recht brauchbare Ergebnisse vorweisen (Abbildung 7.4). Sobald aber das Entwicklerteam nicht mehr verfügbar ist und die Webseite einer Überarbeitung bedarf, wird der Aufwand sehr hoch. Webdesigner müssen sich mit der Implementierung der Applikationen vertraut machen, eine Erweiterung der Funktionalität kann unter Umständen so komplex werden, dass eine Neuentwicklung nicht zu vermeiden ist. Eine solche Architektur ist demzufolge nicht (oder nur unter Einsatz von Zeit und Geld) skalierbar.

Abbildung 7.4: Page Centric Model



Eine Trennung der Präsentationsschicht von der funktionalen Logik würde die Skalierbarkeit von Webseiten erheblich vereinfachen. So könnte mit sehr viel geringerem Aufwand die zugrunde liegende Business-Logik geändert werden, ohne dass sich die Präsentationslogik ändern würde. Eine solche Implementierung erfordert allerdings eine genauere Planung der Applikationen und deren Komponenten. Die Investition von Zeit und Mittel ist zu Anfang vielleicht höher, macht sich aber im Laufe der Zeit bezahlt, wenn die Webseite mit relativ wenig Aufwand geändert werden kann.

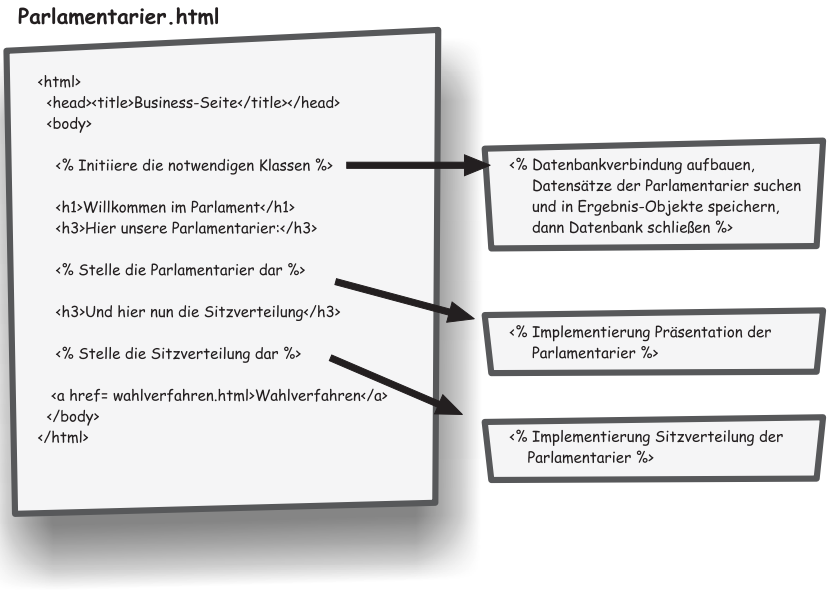


Abbildung 7.5: Component Centric Model

Dabei muss eine Architektur der Implementierung gewählt werden, die diesen Ansatz unterstützt. Schnittstellen zur Trennung von Präsentations- und Applikationslogik müssen definiert werden.

Eine solche Architektur wird als »Component Centric Model« beschrieben. Neben der Trennung von Business-Logik und Präsentationsebene sollte man auch Argumente wie die Wiederverwertbarkeit bereits bestehender Komponenten berücksichtigen. Ein Beispiel ist die Verwendung von Enterprise JavaBeans, ein Satz von Applikationsklassen, die für Geschäftsabläufe in Unternehmen bereits implementiert sind und aus denen man eigene Applikationen ableiten kann.

Page Centric Model: Die Entwicklung einer Webseite ist applikationsgetrieben. Die Realisierung der Applikation ist innerhalb des *html*-Kontext umgesetzt. Dabei wird nicht oder nur kaum zwischen Applikations- und Darstellungsebene unterschieden. Die Umsetzung führt auf diese Weise zu schnellen Ergebnissen, die allerdings langfristig schwer zu warten und schlecht skalierbar sind.

Component Centric Model: Die Trennung der Applikations- oder Business-Logik wird konsequent von der Präsentationslogik umgesetzt. Diese Umsetzung erfordert eine längere Planungsphase und ein sorgfältigeres Vorgehen, dafür lassen sich Weiterentwicklungen einfacher und effizienter vornehmen.

7.5.3 Einige Techniken und Architekturen von dynamischen Webseiten

Zugriffe auf einen Webserver über *http* sind nicht persistent, sondern werden nach jedem erfolgten Datentransfer abgebaut. (Es ist mit *http* 1.1 möglich, über »keep-Alive« eine *http*-Session aufrechtzuerhalten, beim Betrieb eines Webservers versucht man allerdings, diese Verbindungen zu vermeiden, um möglichst viele parallele Anfragen bearbeiten zu können.) Aufgrund dieser Zustandslosigkeit ist es schwierig, dynamische und für den Client individuell gestaltete Webseiten zu generieren, wenn man sich ausschließlich auf die Techniken einer statischen *html*-Seite beschränkt. Für eine Dienstleistung ist es aber gerade notwendig, dass sie auf Eingaben eines Anwenders reagiert und mit dem Client über angepasste Webseiten wechselwirkt. So ist ein *html*-Formular nur dann sinnvoll, wenn auf der Serverseite die Eingaben entgegengenommen und verarbeitet werden, was mit statischen *html*-Seiten nicht oder nur sehr bedingt realisiert werden kann.

Die folgenden Abschnitte befassen sich mit den Grundzügen der verschiedenen Techniken von dynamischen Webseiten.

Active Server Pages

Die von Microsoft entwickelte Technologie bietet eine Möglichkeit, *html*-Daten dynamisch zusammenzusetzen. Während bei einem statischen *html*-Dokument der Inhalt vorgegeben ist und in der fertigen Form von den Clients über den Webserver geladen wird, bietet die Technik von ASPs ein Verfahren, *html*-Seiten über Funktionen dynamisch zusammensetzen zu lassen. Dies wird umgesetzt, indem die Webseiten neben dem reinen *html*-Content Anweisungen in anderen Scriptsprachen beinhalten.

Beispiel:

Eine der wichtigsten dynamischen Anwendungen ist die Darstellung von Datenbankinhalten auf einer Webseite. Diese Webseiten werden über *html* aufgebaut und die Felder werden über ASP dargestellt.

Angenommen, in der Firma »Meinefirma.com« wird eine Datenbank betrieben, die sich über SQL ansprechen lässt. Über eine Webseite bieten wir die Möglichkeit, einen Kunden zu suchen. Eine Anfrage der Form

```
SQL = "Select * from [KundenData] where [Kundennummer] =
"&Request.QueryString("Kundennummer") & "
```

die in einer Initiierungsseite implementiert ist, liefert dem Webserver den gewünschten Datensatz.

Um die Feldinhalte in den *html*-Text zu schreiben, wird die Ergebnismenge auf eine Liste von Attributen gemappt, die in der *html*-Seite dargestellt wird:

```
<HR>
<B>Kundenstammdaten:</B><BR>
<B>Nummer<B>      <% = rs.Fields("Kundennummer")    %>
```

```
<B>Name</B>       <% = rs.Fields("KundenName")    %>
<B>Vorname</B>    <% = rs.Fields("KundenVorname")     %>
```

Der Webserver interpretiert die Script-Befehle und führt sie aus, bevor er anschließend die Daten über *html* an den Client übersendet. Auf Seiten des Browsers ergibt sich die folgende Ausgabe:

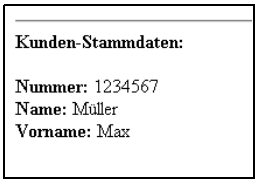


Abbildung 7.6: Beispiel für die Bereitstellung von Datenbankdaten über ASP

Eine ASP-Datei ist eine Textdatei, die beliebige Kombinationen von Elementen in *html*-Format oder Script-Befehlen enthalten kann. ASP-Befehle werden innerhalb einer Seite durch gesonderte Tags (<% ... %>) gekennzeichnet. Alles, was innerhalb dieser Tags steht, wird nicht als Text dargestellt, sondern vom Webserver ausgeführt. Die verwendete Scriptsprache ist üblicherweise Visual Basic.

Da bei dieser Technik keine Standardisierung vorliegt, unterstützen nur wenige Webserver dieses Verfahren. Im Unterschied zu einer reinen *html*-Seite wird ein ASP-Dokument über die Dateiextension ».asp« geführt.

CGI

CGI steht für »Common Gateway Interface« und bezeichnet eine Schnittstelle eines Webservers zu weiteren Anwendungen auf Seiten des Servers. Über diese Schnittstelle greift der Webserver auf Applikationen zu, die in der Umgebung des Betriebssystems als eigene Prozesse laufen. Diese Applikationen können in jeder beliebigen Sprache implementiert werden. Wichtig ist, dass ihre Ausführung durch eine korrekte Umgebung auf Seiten des Betriebssystems realisiert wird. So werden Perl-Scripte nur dann erfolgreich ausgeführt, wenn auf Seiten des Servers ein Perl-Interpreter installiert ist, der automatisch gestartet wird, wenn auf ein Perl-Script ausführend zugegriffen wird, sei es durch einen System-User oder durch den Webserver.

Für die Übergabe von Daten an eine Applikation gibt es zwei Möglichkeiten: »GET« und »POST«. Abschnitt 7.3.1 beschreibt, wie Daten übergeben werden, wenn sie mit einer dieser Methoden übertragen werden. Im Folgenden betrachten wir die Weiterverarbeitung der Daten genauer, wenn sie per *http* über diese Methoden zum Webserver gelangen, der per CGI auf die dafür vorgesehene Applikation zugreift.

Zunächst definieren wir ein Arbeitsbeispiel. Auf einer Webseite ist ein Formular implementiert, das Eingaben des Anwenders übernimmt, die am Webserver eine bestimmte Aktion auslösen. Der Einfachheit halber nehmen wir unser vorhergehendes Beispiel wieder auf und ergänzen es um ein weiteres Formelement, eine Checkbox:

```
<form method="get" action="http://www.meinefirma.com/cgi-bin/Form.pl">
<p>Bitte geben Sie Ihren Vornamen an:
   <input type=text name=Vorname size=25>
<br>
<p>Firmenmitglied:
   <input type=checkbox name="Firma"><br>
<!-- Hier übertragen wir noch eine versteckte Variable: -->
   <input type=hidden name="Testwert" value="42">
<p>Senden Sie das Formular ab:
   <input type=submit value="Senden">
</form>
```

Die Browsersicht eines solchen *html*-Snipplets (inklusive eines exemplarischen Eintrags eines Anwenders) sieht etwa wie folgt aus:

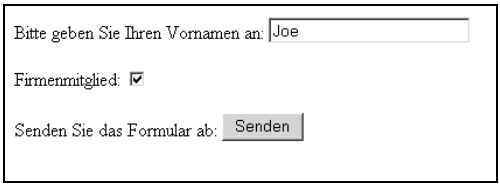


Abbildung 7.7: Beispiel einer html-Form zur Verarbeitung von Anwenderdaten über CGI

Mit dem Betätigen des »Senden«-Buttons wird eine Aktion am Webserver ausgelöst. Die Datenübertragung geschieht über *http*, dabei werden die Daten in eine URL verpackt und an den Webserver übermittelt:

```
http://www.meinefirma.com/cgi-bin/Form.pl?Vorname= Joe&
Firma=on&Testwert=42
```

Die Konfiguration am Webserver sieht ein Verzeichnis vor, in dem ausführbare Programme abgelegt werden. Der Aufruf einer URL, die in dieses Verzeichnis zeigt und auf eine ausführbare Datei zielt, wird vom Server nicht als zu lieferndes Dokument interpretiert. Vielmehr führt er dieses Programm aus, sammelt dessen Ergebnis und reicht es an den Client weiter.

Das ausführbare Programm ist in unserem Beispiel »Form.pl«. Die Extension ».pl« bezeichnet ein Perl-Script, das von einem Perl-Interpreter verarbeitet wird. Die Ausführung des Programms wird vom Betriebssystem des Hosts übernommen, auf dem es hinterlegt ist. Der Webserver triggert lediglich den Aufruf und sammelt den Output des Programms zur Lieferung an den Client. Die an den Webserver übermittelten Daten (hier über den GET-Request) werden als Startparameter im Programmaufruf übergeben.

In diesem Beispiel wurde »GET« als Übergabemethode und ein Perl-Script als ausführbares Programm gewählt. Genauso gut kann die »POST«-Methode für die Datenübergabe sowie ein ausführbares Programm aufgerufen werden, das in einer anderen Programmiersprache geschrieben wurde. Wichtig für den Webserver ist nur, dass der Programmaufruf zu keinem Fehler führt. Der Output des Programms selbst wird an den Client weitergereicht und muss von dessen Browser interpretiert werden können.

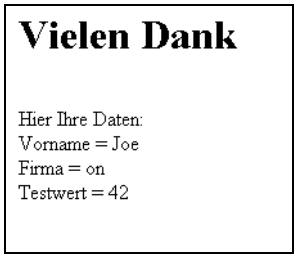


Abbildung 7.8: Browser-Sicht des Response vom Webserver auf die Eingabe der Beispieldaten

Hier ein Beispiel für die Implementierung unserer Applikation:

```
#!/bin/perl
# Beispielprogramm Form.pl
read(STDIN, $eingaben, $ENV{'CONTENT_LENGTH'});
# Hier wird dem Client mitgeteilt, dass nun eine
# html-Seite kommt. Wichtig ist, dass ein "Return"
# nach diesem String übermittelt wird:
print "Content-type: text/html\n\n";
# Die Webseite wird nun erzeugt, indem zunächst
# die Headerinformationen generiert werden:
print "<html><head><title>Vielen Dank</title></head>";
print "<body>";
print "<h1>Vielen Dank</h1><br>";
print "Hier Ihre Daten:<br>";
# Nun beginnen wir mit dem Auswerten der Daten,
# die der Client übermittelt hat:
@felder = split(/&/, &eingaben);
foreach $feld(@felder){
   ($name, $wert) = split(/=/,$feld);
   print "$name = $wert", "<br>\n";
   }
print "<br>";
# Terminieren der Webseite:
print "</body></html>\n";
```

Die *html*-Sequenzen sind fett markiert.

Abbildung 7.8 zeigt die Browserdarstellung dieses Perl-Scripts mit den Eingabewerten aus Abbildung 7.7. Mit Hilfe dieser einfachen Anwendung kann nun ein Datensatz von einer Webseite über ein Formular gelesen und verarbeitet werden. Wir haben uns hier nur auf die einfache Darstellung der Daten beschränkt, natürlich kann man weitaus komplexere Anwendungen programmieren. Auch haben wir hier Perl verwendet, das aufgrund seiner Einfachheit eine beliebte Programmiersprache für die CGI-Programmierung ist. Jede andere Programmiersprache, die die Wechselwirkung zwischen Clients und dem Webserver über *html*-Seiten erlaubt, ist geeignet, Applikationen zu schreiben.

Die Einfachheit der Aktivierung von CGI-basierten Programmen in Verbindung mit der Mächtigkeit von Applikationen, die man nun in den Webkontext einbinden kann, machte CGI schnell zu einer beliebten Architektur dynamischer Webseiten. Ein entscheidender Nachteil von CGI ist die geringe Performance in Verbindung mit Schwächen in der Betriebssicherheit des Serversystems.

Performance: Der Webserver startet für jeden Client-Request einen eigenen Applikationsprozess auf dem Betriebssystem. Jeder diese Prozesse nimmt einen eigenen Speicherplatz in Anspruch, der Webserver muss jede Instanz der Applikation verwalten, obwohl sie außerhalb seiner Betriebsumgebung läuft. Daher kann eine solche Architektur in der Regel nicht sehr viele gleichzeitige Anwender (Concurrent User) bearbeiten.

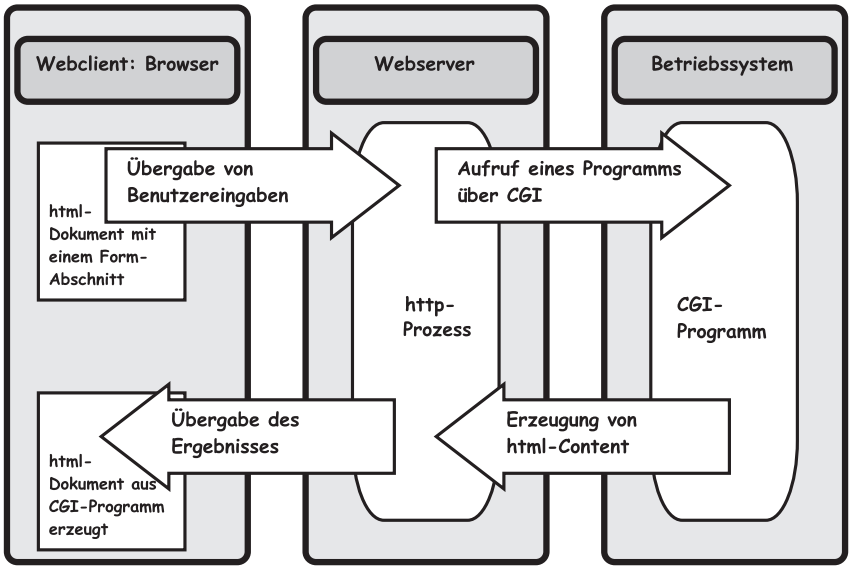


Abbildung 7.9: Prozessablauf beim Aufruf einer Applikation über CGI

Sicherheit: Ein Prozess, der von einem Webserver initiiert wird und auf dem Betriebssystem läuft, kann alles tun, was ihm erlaubt ist. Dies kann zu gefährlichen Nebeneffekten führen, die ein Programmierer eines solchen Scripts vielleicht nicht bedacht hat. So kann es passieren, dass potenzielle Angreifer durch Lücken in Applikationen, die über CGI angesprochen werden, über eine Webseite Zugang zum Betriebssystem erhalten. Insbesondere wenn der Webserver unter Root-Berechtigung gestartet ist, kann der über ihn gestartete Sekundärprozess alles das tun, was die Root-Berechtigung erlaubt.

Viele Webserver umgehen diese Problematik wenigstens teilweise, indem der Webserverprozess unter Root-Berechtigung gestartet wird (dies ist notwendig, da alle Prozesse, die unterhalb Port 1024 laufen, Root-Rechte für den Start erfordern), die Prozesszugehörigkeit anschließend aber an einen ande-

ren System-User mit weit weniger Rechten auf dem Betriebssystem übergeben wird.

FastCGI

Bei CGI ergibt sich das Problem, dass mit jedem Aufruf einer Webseite mit CGI-Applikation ein eigener Prozess auf dem Server gestartet wird, was zu deutlichen Performance-Einbußen führen kann. Um dem zu begegnen, ohne die CGI-Architektur aufgeben zu müssen, arbeitet man bei FastCGI auf Basis eines einzigen Applikationsprozesses, der permanent auf Anfragen durch Clients über den Webserver wartet. Bei einer Implementierung einer Applikation, die in Perl geschrieben wurde, wird der Standard-Perl-Interpreter durch eine auf den Webserver angepasste Version ersetzt, der zur Laufzeit Daten mit dem Server austauscht. Dadurch gelangen die Client-Anfragen direkt in einen einzigen laufenden Prozess. Mit dieser Technik kann man zwar Performance-Gewinne verbuchen, sieht sich aber immer noch mit den Nachteilen der geringeren Performance im Vergleich zu anderen Architekturen konfrontiert. Hinzu kommen die Sicherheitsrisiken dieser Technik.

Java Applets

Java ist eine Programmiersprache, deren wichtigstes Merkmal die Plattformunabhängigkeit ist. Ein Programm, dessen Quelltext in Java vorliegt, wird von einem Java-Compiler in einen Bytecode übersetzt, der von jedem Java-Interpreter, der so genannten »Java Virtual Machine« ausgeführt werden kann, unabhängig davon, ob dieser sich auf einem Unix-, einem Macintosh- oder einem Windows-Betriebssystem befindet. Bei der Programmiersprache handelt es sich um eine Architektur, die darauf zielt, Programme objektorientiert umzusetzen. Ein »Garbage Collector« sorgt dafür, dass Speicherplatz für nicht benötigte Objekte automatisch wieder freigegeben wird. Eine Referenzarithmetik, die ohne Zeiger den Zugriff auf Daten steuert, sorgt für Programmiersicherheit.

Für die in Java implementierten Programme ist nur noch relevant, wo sie ausgeführt werden. So existieren Standalone-Programme auf Java-Basis, deren Bytecode lediglich von der »Java Virtual Machine« (JVM) ausgeführt wird. Applets sind »Prográmmchen«, die von einem Browser im Rahmen einer *html*-Seite vom Webserver heruntergeladen und auf Seiten des Clients ausgeführt werden. So werden beispielsweise Applets auf Webseiten verwendet, die die Systemzeit des Clients auslesen und auf der vom Webserver gelieferten *html*-Seite darstellen. Sinnvollere Anwendungen sind beispielsweise Bankapplikationen, über die eine sichere Verbindung zum Geldinstitut aufgebaut wird und die Überweisungs- oder Kontoverwaltungsfunktionen enthalten.

Ein Applet wird in den Kontext einer Webseite eingebunden, der Programmaufruf kann an jeder beliebigen Stelle im Body einer *html*-Seite erfolgen. Die Startparameter werden in einem Parametersatz der Webseite mitgegeben. Zusammen mit der Webseite und dem parametrisierten Applet führt der

Browser die Applikation clientseitig aus. Hier ein Beispiel für eine Webseite mit einem integrierten Applet-Aufruf:

```
<html>
  <head><title>Beispiel Java-Applet</title></head>
  <body>
    <h2Hier eine Webuhr</h2>
    <applet code=Webclock width=300 height=300>
      <param name=text value="Web-Clock">
      <param name=type value="classic">
      <param name=bgcolor value=255,10,30>
    </applet>
  </body>
</html>
```

In diesem Beispiel wird vom Webserver ein einfaches *html*-Dokument zusammen mit einer Applikation geliefert. Diese hat einen Startparametersatz, der das Design der Applikation, eine Uhr, die die Systemzeit auf dem Client-Rechner ausliest und auf eine bestimmte Art darstellt, bestimmt.

Das Applet muss nicht einmal vom Webprogrammierer selbst geschrieben werden. Mittlerweile gibt es umfangreicher Bibliotheken mit vorgefertigten Applets, auf die man zurückgreifen kann, wenn man eine Internetseite zusammenstellt.

Die Wiederverwertbarkeit ist ein wichtiges Charakteristikum von Java-Applikationen. Einzelne Objekte bis hin zu kompletten Anwendungen werden in Klassenbibliotheken zusammengefasst und über eine hierarchische Struktur verwaltet. Ausgehend von diesen Klassen werden weitere Unterklassen entsprechend den Anforderungen des Programmierers und seiner Aufgabe abgeleitet. Damit ist eines der wichtigsten Argumente des Software-Engineering umgesetzt: die Wiederverwendbarkeit und Erweiterbarkeit bestehender Lösungen. Insbesondere ist das einzelne, individuelle Applet eine Ableitung der verallgemeinerten Applet-Implementierung des Java-Modells.

Der Quelltext des Programms wird über den Java-Compiler in einen Bytecode übersetzt und zusammen mit dem *html*-Dokument vom Webserver geliefert. Der Aufruf eines Applets im *html*-Kontext kann über eine Menge verschiedener Methoden gesteuert werden, die im Applet-Tag definiert sind:

```
<applet
codebase   = [URI auf das Verzeichnis, in dem das Applet
             auf dem Server hinterlegt ist],
code       = [Name der Applet-Klasse],
alt        = [Platzhaltername für das Applet],
name       = [Name der Instanz],
width      = [Breite der Appletdarstellung in Pixel],
height     = [Höhe der Appletdarstellung in Pixel],
align      = [Ausrichtung des Applets auf der Webseite],
```

```
vspace     = [vertikaler Abstand vom html-Kontext in Pixel],
hspace     = [horizontaler Abstand vom html-Kontext in Pixel]>
```

Die Java-Applet-Klasse ist eine vorgefertigte Implementierung einer Klasse, aus der man seine eigenen Applets ableiten kann. Die Basisklasse hat bereits eine Reihe von Methoden implementiert, die direkt verwendet werden können. Hier eine Zusammenfassung der wichtigsten Funktionen:

- public void init()

  Diese Methode initialisiert das Applet, es wird unmittelbar nach dem Laden gestartet. Damit werden alle Objekte und Werte geladen, die für die Ausführung des Applets im weiteren Verlauf notwendig sind.

- public void start()

  Diese Methode wird nach der Initialisierung gestartet. Sie wird auch aufgerufen, wenn man die Webseite, die das Applet enthält, verlassen hat und zu ihr zurückkehrt.

- public void stop()

  Diese Methode wird immer dann aufgerufen, wenn der User die Webseite verlässt, wenn er also auf eine andere Webseite geht oder den Browser schließt.

- public void destroy()

  Diese Methode dient zur Freigabe von Ressourcen und wird nach der stop-Methode aufgerufen, wenn der Browser geschlossen wird oder der User auf eine andere Internetseite wechselt.

Es existieren noch eine Vielzahl weiterer fertiger Methoden, die direkt zur Verfügung stehen, wenn man ein Applet aus der Basisklasse »java.applet. Applet« ableitet.

Java ist eine sehr umfangreiche und mächtige Programmiersprache. Gerade dies kann aber bei Internetapplikationen zu Problemen führen, insbesondere dann, wenn ein solches Programm auf den Client-Rechner heruntergeladen und ausgeführt wird. Ohne Schutzmaßnahmen könnte ein Internetsurfer auf eine Webseite gelangen, über die eine schädliche Applikation übertragen werden kann. Der User würde diese Applikation auf seinen Rechner laden, die ausgeführt würde und die, unbemerkt oder nicht, Daten ausspähen oder zerstören könnte.

Damit dies nicht geschieht, läuft der Bytecode eines Java-Applets in einer so genannten »Sandbox«, das heißt, wenn eine Java-Applikation übertragen und auf dem Client-Rechner ausgeführt wird, dann kann diese Anwendung nur sehr eingeschränkt auf die lokalen Ressourcen des Clients zurückgrei-

fen. So ist es nicht möglich, dass diese Applikationen unbemerkt auf das Filesystem zugreifen oder über Schnittstellen auf andere Systeme übergreifen können.

Das Sandbox-Prinzip wurde mit der Implementierung von Java 1.0 in den ersten Browser mit Java-Unterstützung, dem Netscape Navigator 3.0, aufgenommen. Die Regularien sahen vor, dass eine Applikation, die von extern auf den Rechner gelangte, nur in der Sandbox-Umgebung laufen konnte. Lokal gestartete Applikationen konnten dagegen über die Java Virtual Machine direkt mit dem Betriebssystem interagieren.

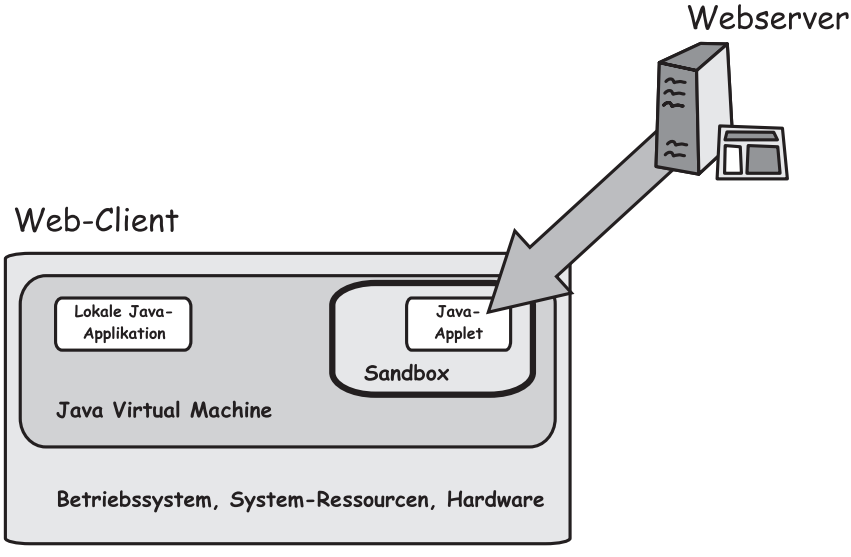


Abbildung 7.10: Sicherheitsmodell von Java 1.0

Für die Regeln der Sandbox gilt, dass insbesondere folgende Aktionen eines fremden Applets nicht unterstützt werden:

- Lokale Dateien können nicht gelesen, beschrieben, umbenannt, gelöscht oder angelegt werden.
- Netzwerkverbindungen über den Ursprung des Applets hinaus dürfen nicht aufgebaut, protokolliert oder angenommen werden.
- Benutzerinformationen wie Name, Account-Daten oder Home-Verzeichnis können nicht protokolliert und übermittelt werden.
- Ein Applet kann keine lokalen Applikationen starten.
- Ein Applet kann den Java-Interpreter nicht beenden.

Eine Reihe weiterer Einschränkungen verhindern, dass die Wechselwirkungen von Java-Applets mit dem lokalen System zu mächtig werden und ein Sicherheitsrisiko für den Client darstellen.

Mit der Implementierung von Java 1.1 wurde das Sandbox-Prinzip erweitert. So konnten Applikationen, die von extern geladen wurden, nun ebenfalls auf die lokalen Ressourcen zugreifen, allerdings mit der Einschränkung, dass der Anwender dies explizit erlauben muss. Ist dies der Fall, so hat die Applikation die gleichen Rechte wie Code, der als lokale Applikation läuft.

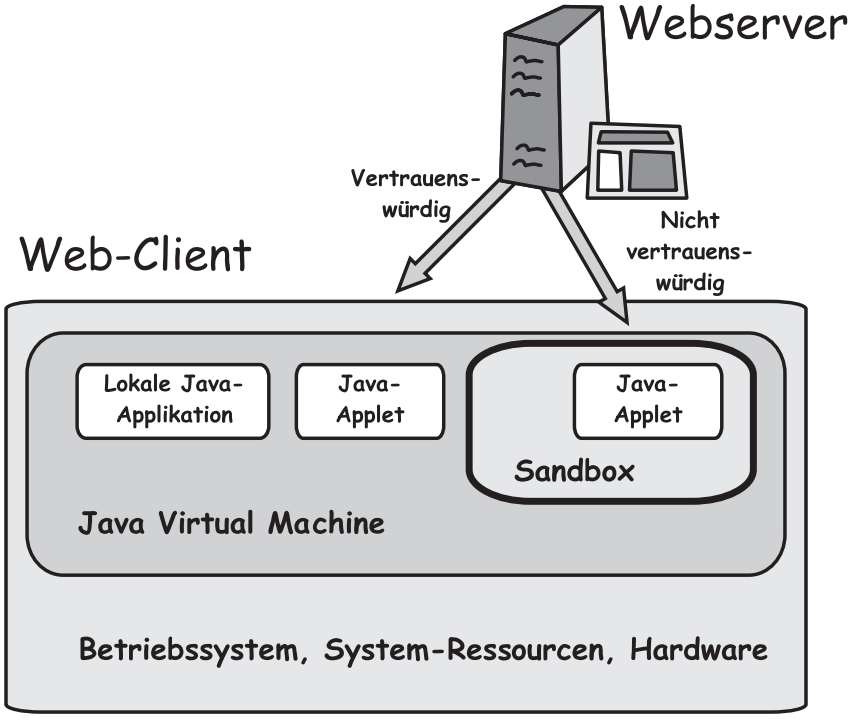


Abbildung 7.11: Prinzip des Sicherheitskonzepts von Java 1.1

Eine weitere Unterstützung der Sicherheit wurde mit der digitalen Signatur eines Applets erzielt. Mit Hilfe von PKI-Mechanismen werden Applets von ihrem Urheber signiert. Vereinfacht bedeutet dies, dass der Hash des Applet-Codes mit Hilfe des PrivateKey des Entwicklers verschlüsselt und als Signatur dem Applet mitgegeben wird. Der Public Key des Entwicklers wurde dabei von einer anerkannten Certificate Authority zertifiziert. Diese Signatur ist nach den Prinzipien der PKI fälschungssicher, da eine öffentlich anerkannte CA für die Authentizität des Urhebers und dessen geleisteter Unterschrift bürgt. Damit kann man bei einem schädlichen Applet den Programmierer ausfindig machen und zur Verantwortung ziehen.

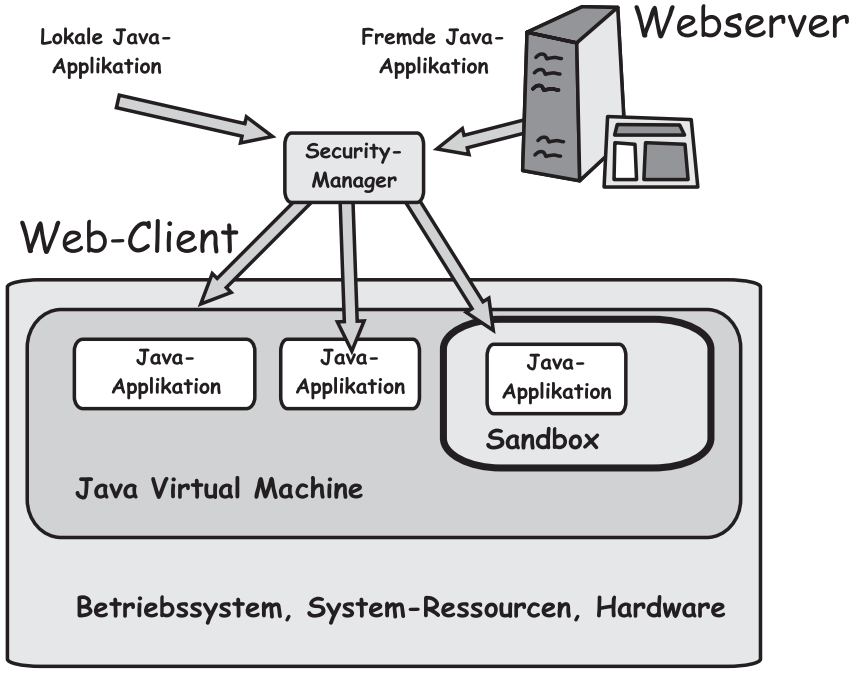


Abbildung 7.12: Prinzip des Sicherheitskonzepts von Java 1.2

Eine weitere Verfeinerung des Sicherheitsmodells der Applets wird in Java 1.2 verwendet. Dabei kann der Benutzer sehr viel differenzierter Zugriffsrechte einer Applikation auf die eigene Ressource gewähren, als es das frühere »Alles-oder-Nichts«-Konzept von Java 1.0 bzw. 1.1 vorsah. So wacht ein Policy-Manager über die Einhaltung dieser Sicherheitsbestimmungen und erlaubt sandbox-übergreifende Zugriffe auf lokale Ressourcen nur dann, wenn sie mit den Security-Bestimmungen des Clients konform gehen. Damit kann ein Benutzer selbst entscheiden, wie viele Rechte er fremden Applikationen auf dem eigenen Rechner einräumt. Dabei wird die Unterscheidung zwischen lokalem und fremden Code aufgehoben, Applikationen werden von der Java Virtual Machine nur noch nach dem zugrunde liegenden Sicherheitskonzept des Security Managers ausgeführt.

JavaScript/JScript/ECMA Script

JavaScript ist eine Sprache, mit der Webseiten mit zusätzlicher Funktionalität versehen werden. Die Bezeichnung scheint eine Beziehung zur Programmiersprache Java nahezulegen, was aber nicht so ist. Beide Sprachen sind voneinander unabhängig und selbst wenn der Code ähnlich aussieht, so haben sie nichts miteinander gemeinsam.

Die Entwicklung der Sprache erfolgte durch Netscape, um eine Webpräsentation mit Funktionalität zu versehen, die mit standardisiertem *html* nicht umsetzbar ist. Der erste javascriptfähige Browser war der Netscape Navigator 2.0 mit einer Script-Implementierung, die damals noch Live-Script hieß.

Der Internet Explorer 3.0 von Microsoft verfügte über eine Implementierung einer Scripting-Sprache, die JScript getauft wurde und die zunächst mit JavaScript hundertprozentig kompatibel war. Im Laufe der Zeit divergierten die Sprachen und heute sind zahlreiche Funktionen entweder nur in dem Browser von Netscape oder dem von Microsoft verwendbar.

Die European Computer Manufacturers' Association (ECMA) versuchte, über eine Standardisierung beide Scriptsprachen wieder zusammenzuführen. Die dabei entstandene Version wird als ECMA Script bezeichnet. Im allgemeinen Sprachgebrauch wird der Name JavaScript weiterverwendet, da sich dessen Entwicklung sehr eng an die Vorgaben der ECMA Script-Spezifikation hält.

Der Anwender kennt JavaScript als Anweisungen, die in *html*-Dokumenten eingebettet sind. Vom Webserver an den Client übergeben, werden solche Anweisungen vom Browser erkannt und ausgeführt, vorausgesetzt, er versteht JavaScript oder die jeweils verwendete Version dieser Sprache.

JavaScript ist eine objektorientierte Scriptsprache. Es besteht im Wesentlichen aus einem zentralen Befehlssatz, dem Core JavaScript, das einen Kern von Objekten wie Array, Date und Math sowie Sprachelemente wie Operatoren, Kontrollstrukturen und Anweisungen besitzt.

Der Befehlssatz von JavaScript, der für die Ausführung auf Client-Seite vorgesehen ist, ist eine Erweiterung des Core JavaScript. Neben dem Client Site JavaScript (CSJS) existiert eine andere Erweiterung des Core JavaScript, das Server Site JavaScript (SSJS).

Während CSJS darauf fokussiert, browser-relevante Objekte bereitzustellen (Funktionen, wie Maus-Klicks auf Buttons, On-Mouse-Over-Events über *html*-Objekte usw.), zielt SSJS darauf, serverseitige Aktionen wie zum Beispiel Datenbankzugriffe oder Wechselwirkungen mit anderen Servern zu kontrollieren.

Java Server Pages

Java Server Pages sind Webseiten, in denen Java-Code eingebettet ist. Dieser Java-Code wird auf Seiten des Webservers ausgeführt, bevor der sich daraus ergebende *html*-Inhalt an den Client übergeben wird.

Java Server Pages unterstützen in vollem Umfang die Trennung von Business-Logik und Präsentationsschicht. So können Webprogrammierer auf Java-Objekte zurückgreifen, ohne diese selbst implementieren zu müssen. Damit können Geschäftsprozesse vollkommen losgelöst von der Darstellung umgesetzt werden und die Java-Direktiven in der Webseite binden nur noch die Ergebnisse dieser Applikationen ein.

Diese Form der Implementierung wird nicht erzwungen, sondern lediglich unterstützt. So kann man durchaus auch Webseiten über JSP programmieren, die dem Paradigma des Page-Centric-Modells entsprechen, in dem sämtliche Applikationslogik innerhalb des einen JSP-basierten Dokuments

eingebettet ist. Die Hauptintention dieser Architektur ist es allerdings, den Webseitenentwickler von der Applikationslogik zu entbinden und ihm lediglich die Präsentationsprogrammierung zu überlassen. Damit werden Webseiten dieser Art vom Programmieraufwand her überschaubarer, leichter zu administrieren und sie sind insbesondere skalierbar. Eine JSP-Seite steht dann als Mittler zwischen der Präsentation von Daten in Form von *html*-Content und der Applikation im Hintergrund. So stellen Java-Beans oder EJB-Komponenten die Funktionalität dar, beispielsweise eine Verbindung zu einer Datenbank oder die Verknüpfung verschiedener Business-Objekte. Webseitendesigner können über angepasste Tags Java-Methoden innerhalb der Webseite aufrufen, ohne selbst für deren Implementierung sorgen zu müssen.

Beispiel :

```
<%@ page info= "Triviales Beispiel für eine JSP-Seite« %>
<html>
  <head><title>Unser Parlament</title></head>
  <body>
    <h2>Hier unser Parlament</h2>
    In dieser dynamischen Grafik ist die Abgeordnetenverteilung
dargestellt :<br>
    <%@ include file= "Abgeordnetenverteilung.jsp" %>    <br>
    Bitte wählen Sie aus:
    <ul>
        <li><a href="geschichte.html">Geschichte des Parlaments</a>
        ...
  </body>
```

In diesem Beispiel werden nur zwei Java-Befehle eingesetzt: die Page- und die Include-Direktive.

In der Page-Direktive werden der JSP-Engine Informationen übermittelt, die das gesamte JSP-Source-File betreffen. In diesem Beispiel ist es nur ein Informationstext, es können aber auch Hinweise über die Scripting-Sprache hinterlegt werden, Angaben über Error-Pages, die für den Fall eines Fehlers innerhalb dieser Seite aufgerufen werden, und vieles mehr.

Die Include-Direktive zieht eine weitere Datei in den Kontext unseres Beispiels hinein. Dabei handelt es sich um eine Applikation, die über grafische Mittel eine Abgeordnetenverteilung des Parlaments darstellt.

Innerhalb einer JSP-Seite kann man demnach beliebig zwischen *html*- und Java-Programmierung wechseln. Es können komplette Applikationen entwickelt und in den *html*-Kontext eingebettet werden. Will man dagegen die Präsentationslogik von der Applikationslogik trennen, unterstützt JSP die Verwendung von Java Beans.

Innerhalb einer Java Server Page werden Tags definiert, die auf Java-Beans zugreifen. Ein Bean ist nichts weiter als eine Java-Klasse, die einer bestimmten Namens- und Designkonvention folgt. Diese JavaBeans-Spezifikation

wurde unter der Federführung von Sun Microsystems entwickelt. In diesen Beans steckt die Logik der Applikation. Die Verwendung eines solchen Bean muss über das Laden und die Initiierung erfolgen. Anschließend kann der Webdesigner auf die Fülle der Methoden zurückgreifen, die diese Applikationen bietet, ohne sich um deren Implementierung Gedanken machen zu müssen.

Mit Java Server Pages bietet sich die Möglichkeit, den vollen Umfang der Java-Programmierumgebung in den Kontext von statischen *html*-Seiten einzubetten. Sie unterstützen die Möglichkeit der Trennung von Applikations- und Präsentationslogik und vereinfachen damit die Weiterentwicklung der Seite sowohl für den Web- als auch den Anwendungsprogrammierer.

Java Servlets

Java Servlets sind in der Programmiersprache Java implementierte Anwendungen, die auf dem Webserver hinterlegt sind und durch die dynamischer *html*-Content erzeugt wird. Aufgrund der Programmiersprache sind sie server- und betriebssystemunabhängig. Die Entwicklung von Applikationen mit dieser Architektur erlaubt den direkten Zugriff auf alle Java-basierten Programmierschnittstellen der Systemumgebung, beispielsweise den Zugriff auf Datenbanken über die JDBC-Schnittstelle, ohne dass die Applikation die Java-Umgebung verlässt.

Wird eine Applikation als Servlet entwickelt, so verwendet sie das von dieser Architektur gelieferte »Java Servlet Development Kit«. In dieser Umgebung sind die wichtigsten Funktionen und Schnittstellen, aus denen man weitere Applikationen ableitet, bereits implementiert. Der Java-Quelltext wird in Bytecode kompiliert und von der Servlet Engine abgearbeitet.

Wird ein Servlet über einen Client-Request angesprochen, so initialisiert die serverseitige Servlet Engine die Applikation und erzeugt einen so genannten Thread, der für die weitere Kommunikation des Clients mit dem Server verantwortlich ist.

Die gesamte Applikation wird mit dieser Architektur in der Webserver-Umgebung gehalten, die Systemsicherheit ist durch die Verwendung des Java Security Manager gegeben, der die Zugriffe der Anwendung auf die lokalen Ressourcen des Servers kontrolliert.

PHP

Das (rekursive) Akronym PHP steht für Hypertext Preprocessor. Dabei handelt es sich um eine Programmiersprache, die in *html*-Dokumenten eingebettet ist und vom Webserver interpretiert wird, bevor die Webseite an den Client ausgegeben wird. Dabei werden die PHP-Sequenzen durch eigene Tags innerhalb des *html*-Kontext gekennzeichnet. Bei einem Aufruf eines PHP-Dokuments wird der Quelltext vom Webserver geparst, die PHP-

Sequenzen werden ausgeführt und in den Gesamtkontext des Webdokuments eingebunden. Betrachten wir wieder das Beispiel aus Abbildung 7.7, das als Eingabeformular Benutzerdaten aufnimmt und eine Applikation aufruft, die in PHP implementiert ist (PHP-Dateien werden mit der Endung ».php3« auf dem Filesystem des Webservers hinterlegt):

```
<form method="get" action="http://www.meinefirma.com/php/Form.php3">
<p>Bitte geben Sie Ihren Vornamen an:
   <input type=text name=Vorname size=25>
<br>
<p>Firmenmitglied:
   <input type=checkbox name="Firma"><br>
<!-- Hier übertragen wir noch eine versteckte Variable: -->
   <input type=hidden name="Testwert" value="42">
<p>Senden Sie das Formular ab:
   <input type=submit value="Senden">
</form>
```

Der einzige Unterschied zur CGI-Version ist die action-Methode des form-Aufrufs »http://www.meinefirma.com/php/Form.php3«.

Der Webserver erhält mit dem Aufruf der PHP-Datei die Übergabeparameter aus den Eingaben des Anwenders in das vorangegangene Formular. Diese werden mit einfachen Mitteln in der Applikation aufgerufen:

```
<html><head><title>Vielen Dank</title></head>
 <body>
  <h1>Vielen Dank</h1><br>
  Hier Ihre Daten:<br>
  <?
    echo("Vorname = $Vorname<br>\n");
    echo("Firma = $Firma<br>\n");
    echo("Testwert = $Testwert<br>\n");
  ?>
  <br>
 </body>
</html>
```

Der Output, den der Webserver nach Evaluierung der PHP-Sequenzen liefert, ist der gleiche wie in Abbildung 7.8. Die PHP-Sequenzen sind fett markiert. Bemerkenswert ist die einfache Übertragung der Formulareingaben, die direkt aus dem PHP-Kontext heraus aufgerufen werden können.

Die Syntax der in PHP verwendeten Befehle ist an Perl, Java und C angelehnt und erweitert durch PHP-eigene Features wie z.B. Kommandos zur Integration von Datenbanken, Schnittstellen zu IMAP4, *ldap* und *snmp*. Voraussetzung für die Verwendung von PHP ist die Fähigkeit des Webservers, PHP zu interpretieren. Dies ist bei manchen Servern gar nicht bzw. nur nach Installation von Erweiterungen möglich.

SSI

So wie PHP bieten Server Side Includes (SSI) die Möglichkeit, dynamische *html*-Seiten serverseitig zu erzeugen. Innerhalb von *html*-Dokumenten können mit dieser Technik

- Variablen gesetzt, ausgewertet und verarbeitet werden,
- CGI-Programme gestartet werden, deren Ausgabe mit dem *html*-Kontext kombiniert wird,
- Dateien eingebunden werden,
- aktuelle Informationen wie z.B. Datum und Uhrzeit aus den Systemvariablen herausgelesen und ausgegeben werden.

Webseiten, die auf SSI basieren, haben normalerweise die Endung ».shtml«.

Voraussetzung, dass ein Webdesigner auf SSI zurückgreifen kann, ist die Fähigkeit des Servers, diese Architektur zu interpretieren. Aufgrund von Sicherheitsrisiken werden Server Side Includes eher selten eingesetzt.

7.5.4 Webseiten-Management

Webseiten sind *html*-Dokumente und -Applikationen, die auf einem Webserver hinterlegt werden, auf die der Server zugreift, wenn diese angefordert werden. Damit greift ein Client über den Webserver auf die Festplatte des Servers zu und fordert ein Dokument an. Um zu verhindern, dass der Anwender auf das gesamte Verzeichnis aller Files Zugriff bekommt, müssen Sicherheitsmaßnahmen umgesetzt werden. Eine der ersten Maßnahmen ist es, dem Webserver nur einen eingeschränkten Zugriff auf die Festplatte zu gestatten und nur auf die Daten zugreifen zu lassen, die für seinen Betrieb notwendig sind. Weitere Maßnahmen sind die Einschränkung des Verzeichnislistings und die Definition der Zugriffsrechte über einen individuellen Access-Control. Diese Themen werden in den folgenden Abschnitten angesprochen.

Dokumentverzeichnisse

Die Organisation von Dokumenten, die veröffentlicht werden können, wird in einem so genannten Document-Root-Verzeichnis hinterlegt, ein Directory auf dem Filesystem. Unter diesem Directory können weitere Unterstrukturen aufgebaut werden, die sich im Aufbau der URL wiederfinden (solange nicht vom Administrator anders konfiguriert).

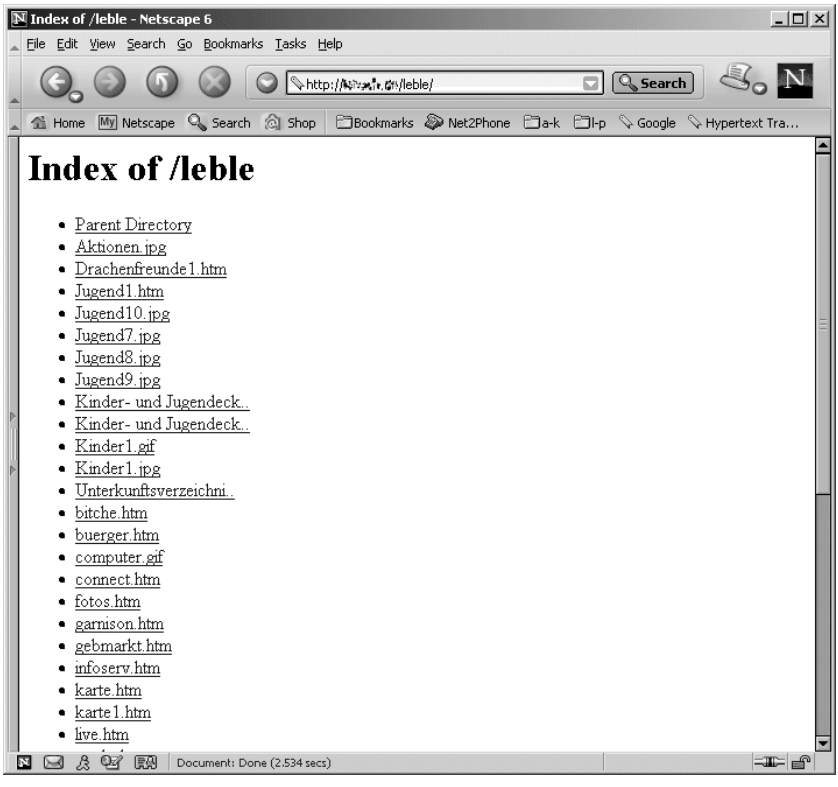
Weitere Dokumentverzeichnisse können auf anderen Partitionen entweder dazugelinkt oder in der Konfiguration des Servers angemeldet werden.

Zumindest im Document-Root-Verzeichnis muss der Webserver ein so genanntes Indexfile haben. Auf dieses greift er zu, wenn von einem Client der Request in Form einer URL aufkommt, bei der kein direktes Dokument referenziert ist.

Beispiel:

Wie bereits gesehen, adressiert die URL »http://www.meinefirma.com« lediglich den Webserver, ohne explizit ein Dokument anzufordern. Diese URL wird vom Browser um einen Slash am Ende ergänzt. Über *http* wird (mit GET/*http* 1.1) direkt in das Document-Root-Verzeichnis verwiesen, aus dem der Server das Indexfile liefert. Auf dem Filesystem ist das Document-Root-Verzeichnis in »/opt/server/docs/« organisiert. Auf diesen Ordner greift der Webserver zu und liefert (beispielsweise) index.html aus.

Abbildung 7.13: Fehlendes Indexfile in einem Unterverzeichnis



Indexfiles können auch in Unterverzeichnissen hinterlegt werden, so dass Client-Zugriffe über Directory-Verweise in URLs ebenfalls direkt mit einem Dokument versehen werden. Was aber geschieht, wenn dort die Indexfiles nicht vorliegen? Die Hersteller der Webserver bieten für diesen Fall zwei verschiedene Konfigurationsmöglichkeiten:

1. Es wird eine Fehlermeldung übermittelt, über die der Client über das Fehlschlagen des Dokumentzugriffs informiert wird.
2. Der Server listet die Dokumentenstruktur auf, wie sie auf dem Filesystem hinterlegt wird. Hier wird nicht das gesamte Filesystem des Servers dem Client geöffnet, sondern nur die Topologie des Dokumentenver-

zeichnisses. (Man kann natürlich auch das gesamte Filesystem öffnen, wenn man es geschickt konfiguriert.) Manche Webserver unterscheiden hier zwischen einer ausführlichen und einer vereinfachten Darstellung: In der einfachen werden nur die Dokumentnamen aufgelistet, in der ausführlichen Darstellung werden darüber hinaus auch noch Größe und Generierungsdatum dargestellt.

Beispiel:

Die URL »http://www.meinefirma.com/leble/« verweist im Document-Root-Verzeichnis auf ein Unterdirectory »leble«. Weil kein Indexfile vorliegt und der Server so konfiguriert wurde, erscheint das Filesystem unterhalb des Document-Root-Verzeichnisses.

Eine solche Darstellung kann insbesondere dann zu unerwünschten Nebeneffekten führen, wenn der Benutzer in dieser Darstellung direkt in Verzeichnisse einblicken, Dokumente unmittelbar aufrufen kann und eventuell Dateien findet, die nicht für die Veröffentlichung bestimmt sind. Sind dann noch Modifikationsfunktionen wie zum Beispiel die »DELETE«-Methode im *http*-Protokoll am Webserver nicht deaktiviert, stehen dem Benutzer alle Wege offen, eine Webseite zu manipulieren.

7.5.5 Link-Management

Webseiten werden miteinander verlinkt und dabei ist es unvermeidlich, dass sich im Laufe der Zeit einzelne Verweise ändern. Insbesondere können Dokumente, auf die verwiesen wird, verschwinden, so dass der Link auf dieses Dokument ins Leere läuft. Eine gute Webseitenpflege beinhaltet auch einen Mechanismus, der in regelmäßigen Abständen die Aktualität der Verweise im Dokumentenbestand überprüft und nicht mehr gültige Links meldet. Damit beugt man dem schlechten Eindruck einer schlecht gepflegten Webseite vor.

Um ein Link-Management einzurichten, kann man einerseits den einzelnen Anwender auffordern, »tote« Links zu melden, um sie unmittelbar vom Webadministrator korrigieren zu lassen. Der bevorzugte Lösungsansatz ist aber die Verwendung eines Link-Robots, der über einfache Mittel des *http*-Protokolls tote Links ausfindig macht, protokolliert und sie dem Webadministrator meldet.

Der Ansatz für eine automatische Link-Validierung ist die »HEAD«-Methode, wie sie bereits in Abschnitt 7.3.1 vorgestellt wurde. Hier war zu sehen, wie nach einer Telnet-Anmeldung auf einen Webserver diese Methode zur Übertragung von Dokumenteninformationen genutzt wurde, ohne dass der gesamte Inhalt der Seite übertragen wird.

Ein Link-Checker parst somit eine Startseite nach auftretenden Links, die in einer Liste von noch abzuarbeitenden Überprüfungen gelistet werden. Anschließend wird jede einzelne Seite auf Aktualität geprüft. Ein Link ist dann gültig, wenn er einen ordnungsgemäßen Header des verlinkten Doku-

ments liefert. Werden dagegen Fehlermeldungen übertragen, insbesondere Fehler der Klasse 400, so deutet dies auf einen ungültigen Link hin. Ein Verweis mit dem Fehler 404, »Not found«, deutet auf ein entferntes Dokument hin. Bei den anderen kann es zu einer Änderung des Zugriffsrechts gekommen sein, so dass das Zieldokument vielleicht immer noch vorhanden ist, aber die Berechtigung für diese Datei nicht vorliegt.

Fehler der Klasse 500 deuten auf einen Server, der ein (temporäres) Service-Problem hat. Hier sollte ein Linkchecker zu einem späteren Zeitpunkt noch einmal überprüfen, ob das Dokument erreichbar ist oder nicht.

7.6 Zugriffssteuerung auf Webseiten

Führt man eine webbasierte Publikation ein, so muss klar definiert sein, welcher Anwender auf welches Dokument zugreifen kann. Wie im »Papier-Umfeld« gestattet man nicht unmittelbar jedem, in Dokumentationen, Kalkulationen oder Forschungsergebnisse einzusehen. Bei der Implementierung einer elektronischen Präsentation gelten diese Anforderungen ebenfalls. Bevor man also Daten einem unmittelbaren Zugriff preisgibt, muss man die Berechtigungen für den lesenden, schreibenden und löschenden Zugriff sorgfältig planen. Anschließend werden diese durch die Mittel des Servers umgesetzt.

Für die Realisierung einer Zugriffspolitik auf Dokumente stellen Server unterschiedliche Möglichkeiten bereit. Dabei kann man zwischen kontrollierter aktiver Steuerung des Servers und passiver Zugriffskontrolle unterscheiden: Bei der aktiven Kontrolle analysiert der Server aktiv die Verbindung, hinterfragt die Identität des Clients und verlangt Authentisierungsinformationen. Bei der passiven Steuerung ist der Server mit all diesen Techniken von dieser Verantwortung enthoben und der Betreiber der Webseite verlässt sich auf andere Mittel, die die Steuerung übernehmen. Im Folgenden werden die einzelnen Formen dieser Kontrolle im Einzelnen untersucht.

7.6.1 Aktive Zugriffssteuerung

Nicht auf jedes Dokument sollte uneingeschränkt zugegriffen werden können, meist ist der Zugriff nur lesend oder auch schreibend erlaubt. Um eine solche Regelung umzusetzen, werden Zugriffsrechte auf die Daten vergeben. Solche Rechte werden als »Access Rules« oder »Access Control Information« (ACIs) beschrieben. (In manchen Publikationen wird ACI auch als »Access Control Item« bezeichnet. Unabhängig davon, welche der beiden Interpretationen des Akronyms verwendet wird, beschreiben beide die gleichen Anweisungen.) Eine Liste von ACIs bezeichnet man als »Access Control List« (ACL), in der alle Zugriffsrechte zusammengefasst sind.

Um eine ACI zu definieren, werden zwei Seiten miteinander verglichen:

- Der Client: Dies kann eine Person oder Applikation sein, die vom Server Daten anfordert. Der Client wird entweder in der Webserver-eigenen Datenbank gehalten oder er wird über *ldap* vom Directory Server erfragt. Gelangt ein Client-Request an den Webserver und dieser erkennt, dass es sich hier um ein geschütztes Dokument handelt, so fordert er vom Anwender über ein Pop-up-Fenster auf Seiten des Browsers die Angabe von User-ID und Passwort an. Mit diesen Daten versucht sich der Server anschließend am Directory Server anzumelden: Schlägt dies fehl, so ist der Anwender nicht im Besitz des korrekten Passworts und kann sich somit nicht gegenüber dem Webserver (bzw. dem Directory Server) identifizieren.

  Aber nicht nur die Identifizierung des Clients definiert einen Client-Zugriff auf die Daten. Vielmehr können die Zugriffe auf Wochentage, Tageszeiten, IP-Adressbereiche oder Domänen eingeschränkt werden. Weiterhin kann festgelegt werden, ob ein anonymer Zugriff erfolgt oder ob die Authentifizierung über User-ID/Passwort oder digitales Zertifikat erfolgt, ob der einzelne User einen Zugriff erhält oder ob die Regel für eine Gruppe von Usern gültig ist.

  Damit hat eine ACI ein sehr mächtiges Regelwerk, um festzulegen, welcher Client in welcher Form einen Zugriff auf die Daten erhält.

- Die Ressource: Dies ist das Dokument selbst, das vor unberechtigtem Zugriff geschützt wird. Dabei können einzelne Files, Directorys oder aber ganze Serverinhalte angesprochen werden.

- Die Art des Zugriffs: Hier wird festgelegt, ob es sich um einen lesenden und/oder schreibenden Zugriff handelt und ob dieser erlaubt oder verboten ist.

Im Allgemeinen ist eine Defaultregel auf Seiten eines Webservers installiert, die besagt, dass ohne eine bestimmte ACI der lesende Zugriff auf die Dokumente erlaubt ist. Geht man von dieser allgemeinen Regel aus, so kann man eine Sequenz Deny-Regeln davor setzen, die diesen Zugriff unterbinden. Wie im Directory Server, nur unter umgekehrten Vorzeichen, wird dabei das ACI-Regelwerk abgearbeitet: Zunächst werden alle Allow-Regeln überprüft, anschließend werden alle Deny-Regeln kontrolliert. Erst wenn keine Bedingung für ein bestimmtes Dokument greift, wird die Defaultregel umgesetzt. So greift jede »Allow«-Regel vor jeder »Deny«-Regel, die wiederum vor der Standard-Allow-Regel umgesetzt wird.

Dieses Verfahren ist nicht einheitlich. Wichtig ist, dass es möglich ist, eine sehr feine Zugriffsstruktur aufzubauen.

Beispiel: Man kann mit einfachen Mitteln die Regel umsetzen, in der festgelegt wird, dass

- allen Clients aus einer bestimmten Gruppe,
- die sich zu einer bestimmten Tageszeit anmelden,

- der lesende Zugriff auf die Daten erlaubt wird,
- wenn es sich um ein Dokument aus dem Verzeichnis »Marketing« handelt.

Während die ACIs auf Seiten des Webservers vorgehalten werden, um die Zugriffssteuerung zu regeln, werden Account-Informationen mehr und mehr einem Directory Server überlassen.

Für die Authentifizierung an einem Webserver sind zwei Methoden gebräuchlich: Die einfache Authentifizierung über eine User-ID/Passwort-Kombination und die kryptische über SSL. Weitere Authentifizierungsmethoden wie über Digest oder Kerberos werden seltener verwendet. Wir werden uns mit den beiden erstgenannten eingehender beschäftigen.

Authentifizierung über User-ID und Passwort

Bei dieser Authentifizierungsform werden die Account-Daten über das *http*-Protokoll folgendermaßen ausgetauscht:

1. Der Client fordert ein bestimmtes Dokument über eine URL vom Webserver an, das in einem geschützten Webserverbereich liegt. Der Client kann im ersten Zugriff noch nicht wissen, dass Authentifizierungsdaten übermittelt werden müssen, und fordert die Seite ohne Authorization Flag im *http*-Protokoll an. Der Webserver lehnt diesen Zugriff mit dem Statuscode 401 ab:

```
GET /kunden/secure/faq/kundenfragen.html HTTP/1.1
host: www.meinefirma.com

HTTP/1.1 401 Unauthorized
Server: NetWare-Enterprise-Web-Server/5.1
Date: Sun, 14 Oct 2001 15:05:30 GMT
WWW-authenticate: Basic
realm="Kundenbereich"
Content-type: text/html
Last-modified: Thu, 16 Aug 2001 11:36:58 GMT
Content-length: 3281
Accept-ranges: bytes
```

2. Der Code 401 veranlasst den Browser, die Authentifizierungsdaten vom Anwender über ein Pop-up-Fenster einzufordern. »WWW-authenticate« informiert den Browser darüber, dass eine Authentifizierung über User-ID und Passwort erwartet wird, der Wert des »realm«-Attributs wird als Textbaustein im Pop-up-Fenster verwendet, um den Anwender zu informieren, in welchem Webserver-Bereich er sich gerade authentifizieren möchte.
3. Der Anwender übergibt die Account-Daten über das Pop-up-Fenster an den Browser, der seinerseits einen erneuten Request über *http* formuliert, in dem aber diesmal die Account-Daten (<Authorization: Basic base64 (username:password)>) übermittelt werden.

Abbildung 7.14: Pop-up-Fenster des Browsers zum Einfordern der Authentifizierungsdaten

4. Der Server evaluiert die Account-Daten und versucht über *ldap* einen Bind an den Directory Server. Schlägt dieser fehl, gibt der Webserver eine Fehlermeldung an den Client weiter.
5. Ist der Bind an den Directory Server erfolgreich, »weiß« der Webserver, welcher User den Request formuliert hat.

Nun ist dem Webserver bekannt, um welchen Anwender es sich handelt. Im nächsten Schritt muss er herausfinden, ob es eine Regel gibt, die den Zugriff für diesen User erlaubt. Für den Fall, dass gemäß der ACI eine bestimmte Gruppe für den Zugriff bestimmt ist, muss der Webserver über einen erneuten *ldap*-Zugriff herausfinden, ob der User Mitglied dieser Gruppe ist. Andernfalls vergleicht er die weiteren Client-Spezifika (IP-Adressbereich, Domain, Tages- oder Nachtzeit usw.) mit dem Target-Satz, auf den der Client zugreifen möchte.

Es sei noch einmal darauf hingewiesen, dass der Webserver Authentifizierungsdaten nicht speichert. Damit ist mit jedem Request des Clients über *http* in einen geschützten Bereich eine erneute Authentifizierung erforderlich. Für ein *html*-Dokument, das mit Grafiken versehen ist, bedeutet dies, dass für jedes Bild, jede Multimedia-Datei, falls sie ebenfalls aus dem geschützten Berecht stammt, eine neue Verbindung über *http* aufgebaut wird, die gleichfalls einer Authentifizierung bedarf. Damit der Anwender nicht für jedes Dokument erneut seine User-ID und sein Passwort angeben muss, werden diese Daten im Memory des Browsers vorgehalten (solange die Browserinstanz läuft). Mit jedem weiteren Zugriff in diese Domain verwendet das Client-Programm diese Daten.

Dieser Zugriff über User-ID und Passwort wird gemeinhin als unsicher angesehen. Dies resultiert insbesondere daraus, dass die Authentifizierungsdaten ungeschützt durch das Netz wandern: Der Anwender übergibt seine Account-Daten (User-ID und Passwort) an den Browser, dieser enkodiert mittels base64 die Daten und übersendet sie im *http*-Protokoll im Header über das Authorization-Flag:

```
Authorization: <Basic base64 (username:password)>
```

Hierbei handelt es sich nicht um einen Schutz, nicht einmal um eine Verschlüsselung. Die Daten werden reversibel kodiert, damit der Webserver sie auch wieder dekodieren kann. Dies kann aber auch jeder andere, der im Protokoll auf dem Netz lauscht. Somit werden die Account-Daten mehr oder weniger im Klartext über das Netz gesendet. Eine sicherere Methode der Authentifizierung wird im nächsten Abschnitt vorgestellt.

Authentifizierung über SSL

Die Authentifizierung über SSL geschieht mit anderen Protokollalgorithmen. Diese werden kryptografisch behandelt und sind vom Aufwand her viel umfangreicher.

Ein Client, der eine Webseite »betritt«, die eine Authentifizierung über SSL verlangt, muss ein gültiges Zertifikat vorweisen, um den Zugriff auf das gewünschte Dokument zu bekommen. SSL erzwingt nicht in jedem Fall die Client-Authentifizierung. Vielmehr steht in den überwiegenden Fällen, bei denen SSL zum Einsatz kommt, die Server-Authentifizierung im Vordergrund und nur der Client wird informiert, mit welchem Server er auf der anderen Seite verbunden ist.

In manchen Fällen wird aber auch die clientseitige Zertifizierung verlangt, die vom Prinzip her genauso verläuft, wie der Server sich gegenüber dem Client ausweist. Bereits Kapitel 6.8.1 hat die Grundprinzipien der SSL-Authentifizierung gezeigt. Der Einfachheit halber werden hier die ersten Schritte noch einmal komprimiert vorgestellt und um die Client-Authentifizierung erweitert:

- Ein Benutzer startet seinen Browser und öffnet die URL https://www.internet-music-online.de.
- Der Browser bekommt ein serverseitiges Zertifikat übermittelt, dessen Gültigkeit er überprüft.
- Der Server seinerseits fordert ein clientseitiges Zertifikat an, das der Browser bereitzustellen hat. Hierzu verlangt der Browser seinerseits vom Anwender das Passwort für die Datenbank, in der die Client-Zertifikate und insbesondere der Private Key des Anwenders hinterlegt sind. Dies ist notwendig, da sich der Browser damit versichert, dass nur der berechtigte Anwender den Rechner nutzt. (Ein unbeaufsichtigter Rechner könnte einen unbefugten Anwender in die Lage versetzen, Zugriff auf Daten zu bekommen, die nur dem Eigentümer der Maschine vorbehalten sind. Da aber nur der wirkliche Eigentümer das Passwort für den Zugriff auf den Private Key und die Certificate-Datenbank kennt, ist hier ein gewisser Schutz vor Missbrauch gewährleistet). Mit dem Passwort öffnet der Browser die Datenbank der Zertifikate und fordert den Anwender auf, das für diese Webseite geeignete Zertifikat auszusuchen.
- Der Anwender wählt ein Zertifikat aus, das für diese Webseite erforderlich ist, der Browser präsentiert dieses Zertifikat dem Server.
- Der Server überprüft die Angaben des Zertifikats (über die digitale Unterschrift der CA).
- Der Server überprüft die Authentizität des Anwenders, indem er einen mit dem Public Key des Clients verschlüsselten Datensatz vom Client über dessen Private Key entschlüsseln lässt. Damit hat der Server die Gewähr, dass der anfordernde Anwender der Eigentümer des Zertifikats ist.

Sind alle diese Schritte erfolgreich absolviert, »weiß« der Server, welcher Anwender den Zugriff auf ein geschütztes Dokument anfordert. Gemäß den Regularien der Access-Control-Informationen entscheidet er nun über die Freigabe des Dokuments.

7.6.2 Passive Zugriffssteuerung: Backdoors

Manche Internetdienste bieten Sammelaccounts an, mit denen man nicht nur auf einer Webseite Zugriff erhält, sondern auf vielen. Gerade im Bereich der Erotikseiten bieten so genannte »Adult-Verification-Systeme« (AVS) dem Interessenten gegen einen bestimmten Betrag den Zugriff auf Sammlungen einschlägiger Webseitenanbieter. Das Geschäftsmodell gestaltet sich üblicherweise wie folgt:

Der AVS-Anbieter stellt eine Liste der teilnehmenden Webseiten bereit. Der Interessent bekommt vom AVS-Anbieter gegen einen Beitrag ein so genanntes Passwort, mit dem er sich auf allen Webseiten einloggen kann, die an diesem AVS-System teilnehmen.

Die teilnehmende Webseite bekommt einen Teil des Beitrags des Kunden ausgezahlt (entweder eine Pauschale oder eine nach Aufkommen abgerechnete Gewinnbeteiligung). Damit macht sowohl der AVS-Anbieter als auch die teilnehmende Webseite einen Gewinn.

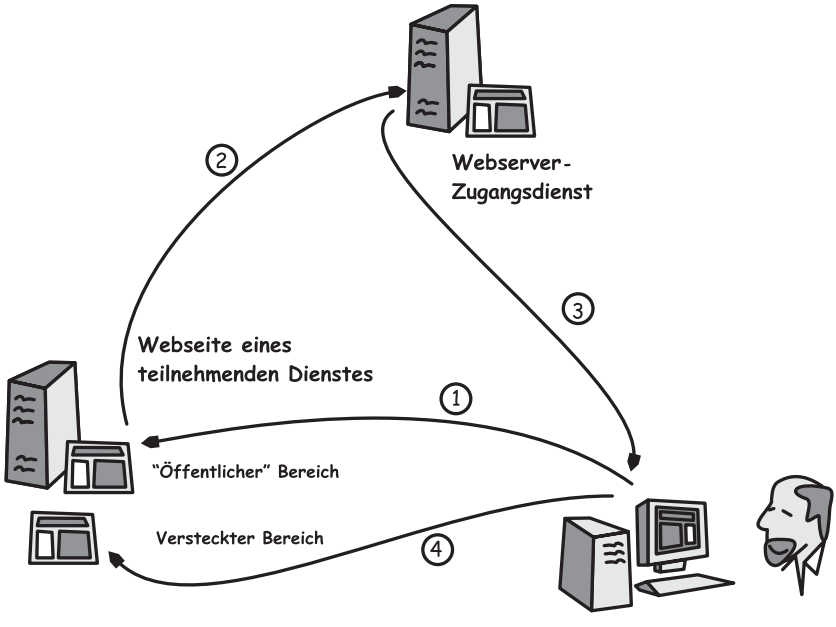


Abbildung 7.15: Backdoor-Verfahren: 1. Eingangsseite einer kommerziellen Webseite 2. Redirect mit Passwort auf den Zugriffsdienst 3. Bekanntgabe der Backdoor 4. Direktzugriff auf die Backdoor

Die Zugriffssteuerung geschieht über ein kleines *html*-Formular auf der Seite der teilnehmenden Webseite, auf der der Interessent sein Passwort eingibt. Der Submit-Button führt über einen URL-Redirect auf die Seite des Zugriffsdienstes, der das Passwort prüft und für den Erfolgsfall den Interessenten wieder auf die ursprüngliche Webseite in den geschützten Bereich führt.

Da es sich hier nicht um einen aktiven Zugriffsschutz auf Filesystem-Ebene handelt, sondern nur um eine Verifikation eines Passworts auf Seiten einer dritten Instanz, kann der direkte Zugriff in den geschützten Bereich, sofern man die dafür vorgesehene URL kennt, direkt erfolgen.

Diese Art der Zugriffssteuerung bezeichnet man als Hintertür oder »Backdoor«. Der Zugriff ist nicht durch aktive Kontrolle der User-Daten auf Protokollebene realisiert, der geschützte Bereich wird vielmehr versteckt und nur vom AVS-Anbieter verraten.

Der Anreiz, eine solche Architektur zu implementieren, liegt in der Einfachheit des Geschäftsmodells.

- Der Interessent kann mit einem einzigen Passwort Zugang zu vielen Webseiten bekommen. Er kann beim AVS-System auf eine Liste der verfügbaren Webseiten zugreifen, die in einem Katalog aufgelistet und verschlagwortet bereitgestellt werden.
- Der AVS-Dienst bietet die Liste der Webseiten an und gewährt nur denjenigen Interessenten Zugriff, die ein Passwort für eine gewisse Dauer »gekauft« haben. Mit wachsendem Umfang der Webseitenliste wird dieser Dienst attraktiver. Die Verantwortung für den Inhalt der teilnehmenden Webseiten wie deren Wartung ist nicht Bestandteil dieses AVS-Dienstes. Dessen einzige Aufgabe ist die Verwaltung der Interessenten auf der einen, die Liste der Webseiten und deren Backdoors auf der anderen Seite.
- Der Betreiber einer teilnehmenden Webseite kann mit einfachen Mitteln einen Dienst bereitstellen, den er entweder selbst betreut oder vom ISP hosten lässt. In letzterem Fall muss er nur den Inhalt der Webseite bereitstellen und braucht sich nicht einmal um die Server-Hard- oder Software zu kümmern.

  Für manche Dienste dieser Art ist eine Zugangskontrolle zum Beispiel aus Jugendschutzgründen Pflicht. Diese Form der Zugangssteuerung benötigt nicht einmal die Kooperation des Systemadministrators, auf dessen Webserver man seine Seite einrichtet.
- Und schließlich verdient der Betreiber mit dieser Architektur relativ einfach Geld, indem er auf seiner Seite lediglich das Formular-*html*-Snipplet des Zugangsdienstes auf der Eingangsseite einbaut.

So interessant dieses Geschäftsmodell erscheint, so schwach ist der eigentliche Zugangsschutz. Der Verweis auf eine Hintertür innerhalb der Webseite ist keine wirkliche Sicherungsmaßnahme, sie stützt sich nur auf die Verschleierung eines Zugangs, den man mehr oder weniger gut versteckt. Darü-

ber hinaus ist nicht sichergestellt, dass der »Eigentümer« eines Passworts der einzige ist, der es verwendet. So gesehen mag diese Architektur ein Geschäftsmodell sein, das unter Umständen lukrativ ist, einen wirklichen Schutz vor unberechtigtem Zugriff auf eine Webseite bietet sie jedoch nicht.

7.7 Einrichten einer webbasierten Informationsstruktur

Vor einiger Zeit wurde der Begriff des »papierlosen Büros« geprägt. Man hatte dabei die Vision der Kommunikation auf elektronischem Weg und der Informationsbeschaffung über Netzsysteme, darunter insbesondere Webdienste, über die man Dokumentationen publizieren kann. Selbst wenn eine solche Infrastruktur unter heutigen Gesichtspunkten nicht oder nur teilweise zu realisieren ist, bietet die elektronische Publikation immer noch sehr viel Potenzial, um Informationsbeschaffung und -bereitstellung zu vereinfachen. So kann eine digitale Verteilung von Dokumenten helfen, deren Druckkosten zu sparen, ihre Administration zu vereinfachen und die Aktualität zu gewährleisten.

Die folgenden Abschnitte sollen die notwendigen Schritte für die Einrichtung einer digitalen Verteilung elektronischer Daten über Web-basierte Dienste aufzeigen.

7.7.1 Identifikation von zu publizierenden Daten

Die Publikation von Daten erfolgt aus unterschiedlichsten Motiven. Die meisten Seiten werden von Firmen angeboten, die mit ihrer Internetpräsenz Kunden anwerben und Produkte verkaufen möchten. Es werden Webseiten betrieben, die Dienstleistungen wie Übersetzungsservices, Auskunfts- oder Nachschlagedienste anbieten (beispielsweise Telefonnummern oder Lexika). Teilweise ist der Betrieb einer solchen Seite uneigennützig, in der Regel aber versucht man, entweder direkt oder indirekt einen wirtschaftlichen Vorteil aus der Bereitstellung dieser Dienste zu erreichen.

- Direkte kommerzielle Nutzung: Der Betreiber fordert einen Beitrag vom Kunden ein, bevor er die Seite für die Nutzung freigibt. In der Regel wird dies über die Bereitstellung einer User-ID/Passwort-Kombination, seltener in Form von digitalen Zertifikaten durchgeführt. Die Zugriffsrechte haben entweder eine permanente oder eine zeitlich limitierte Gültigkeit.
- Indirekte Abrechnung über Drittanbieter: Der Betreiber bietet die Dienste scheinbar kostenlos an. Dabei wird allerdings häufig übersehen, dass so genannte Bannerwerbung einen teilweise nicht unerheblichen Aufwand in Form von Online-Zeit und Bandbreite erfordert, die vom Anwender bezahlt werden muss. So kann ein Anbieter einer Service-Seite einen Teil der Seite als Werbefläche verkaufen, die in der Regel umso teurer wird, je höher die Zugriffsraten auf die Seite sind.

Diese Werbung kann nachträglich noch weiter verrechnet werden, wenn die Dienstleistungsseite nachweist, wie viele Anwender tatsächlich auf die Bannerwerbung geklickt haben und auf die Seite des Werbenden gelockt wurden.

Es kann aber auch noch einen weiteren Grund für die Platzierung von Bannern geben: Das Ausspionieren der Surfgewohnheiten von Usern. Dabei wird über Cookies nachgestellt, welchen Weg der Surfer durch das Internet genommen hat. Die genaue Beschreibung dazu wird in Abschnitt 7.8 über die Verwendung, die Vorteile und die Gefahren von Cookies vorgestellt.

Gegenüber der externen Publikation von Daten steht im Intranet in der Regel die Information der Mitarbeiter. Mit dem Aufkommen der ersten Webseiten erkannte man schnell, dass Handbücher, Dienstvorschriften oder Dokumentationen in *html*-Form für den Mitarbeiter immer verfügbar sind und dass deren Aktualisierung relativ einfach und zeitnah geschehen kann, ganz im Gegensatz zur gedruckten Papierform. Ein beliebtes Beispiel ist das ausgedruckte Telefonverzeichnis. Wird ein solches Exemplar an jeden Mitarbeiter ausgeliefert, veralten die Daten und alle Mitarbeiter haben die gleichen falschen Daten bis zur nächsten ausgedruckten Version. Eine Webseite wird ein einziges Mal geschrieben oder aktualisiert und ist sofort für alle in der korrekten Form verfügbar.

Für die Planung des Aufbaus einer Publikationsumgebung sollte die Identifizierung der zu publizierenden Daten an allererster Stelle stehen. Wenn es für die in Frage kommenden Dokumente kaum Interessenten gibt, sollte man sich überlegen, ob die Papierform nicht eine kostengünstigere Alternative ist. Finden allerdings die Publikationen einen Anwender- oder Interessentenkreis, so ist eine digitale Veröffentlichung eine sehr viel bessere Form der Datenbereitstellung.

7.7.2 Definition der Wechselwirkungen mit anderen Systemen

Wie bereits gesehen, werden statische Seiten überwiegend für Dokumentationen benutzt, die sich nicht häufig ändern. Viele Homepages von Privatanwendern bestehen aus Texten und Bildern, bestenfalls von Adressdaten wie Telefonnummer und E-Mail-Adresse. Diese Seiten sind ausschließlich im Webumfeld installiert und interagieren nicht mit anderen Systemen.

Spätestens mit der Bereitstellung von Dienstleistungen verlässt die Webseite den reinen Darstellungsbereich und liefert angepasste Informationen. Dies ist nur möglich, wenn der Webserver über geeignete Schnittstellen hinaus auf andere Systeme zugreift. So wird ein Formular, das ein Anwender ausgefüllt an den Webserver zurückschickt, von einer Applikation ausgewertet, die Daten werden verarbeitet und der Server dient nur noch als Präsentationsmittel für den eigentlichen Service. Flugbörsen im Internet sind an Datenbanken angeschlossen, die weit über den lokalen Standort des Webser-

vers hinaus auf anderen Systemen verteilt sind. Firmen-Preislisten für Produkte werden nicht »hart« in eine Webseite programmiert, sondern Applikationen auf dem Webserver greifen in Datenbanken hinein und generieren dynamisch Webseiten, die die Produktpreise auflisten. Webbasierte Telefonbücher haben Eingabemasken für den Anwender, die damit verbundenen Applikationen greifen über *ldap*-Schnittstellen auf den Directory Server zu und generieren dynamischen *html*-Inhalt.

Wenn eine Webseite über eine reine Darstellung von statischen Daten hinausgeht und Dienstleistungen anbietet, müssen die Wechselwirkungen mit anderen Systemen in Betracht gezogen werden. Damit verbunden ist aber auch ein Audit der Konsequenzen, die ein solcher Dienst mit sich bringt. Mit dem Öffnen von Datenbanken über Webseiten für den lesenden oder sogar schreibenden Zugriff müssen die Sicherheitsrichtlinien der Daten neu überdacht werden. Ist es beispielsweise wünschenswert, dass der Einkaufspreis einer Ware auf der Webseite erscheint? Darf ein Kunde seinen Stammdatensatz lesen oder gar verändern? Bietet eine Web-Applikation die Transaktionssicherheit, um die Datenintegrität zu bewahren? Werden dem Anwender am anderen Ende der Leitung vielleicht Hintertüren gegeben, die die Datensicherheit gefährden? Kann eine Webapplikation Performance-Einbußen mit sich bringen, die den Arbeitsablauf aller anderen Systeme beeinflusst?

Das Einrichten einer webbasierten Publikationsstruktur sollte sich in dieser Phase eng an die Vorgaben einer bereits bestehenden Informationspolitik halten und keine Löcher in die Sicherheitsbestimmungen und des Datenschutzes reißen. Ein Audit an dieser Stelle sollte die Gefahren und Risiken aufdecken und als Ergebnis eine Risikoabschätzung liefern.

7.7.3 Festlegung der Zugriffsberechtigungen

Dokumente, die auf digitalem Weg erreichbar sind, müssen eine Spezifikation der Zugriffsberechtigung aufweisen. So wie jedes physikalische Dokument einen Adressatenkreis hat, so folgen digitale Dokumente einer Zugriffspolitik.

Die allgemeinsten Formen einer solchen Kontrolle sind:

- absolut freier Zugriff für jeden auf alle Dokumente oder Teilbereiche
- absolut restriktiver Zugriff auf alle Dokumente und deren Inhalt (was allerdings niemandem mehr, auch dem Autor nicht, ein irgendwie geartetes Zugriffsrecht erlaubt)

Zwischen diesen beiden Extremen sind alle Facetten der Zugriffskontrolle denkbar, die es zu identifizieren und umzusetzen gilt.

Bereits Abschnitt 7.6 hat Möglichkeiten aufgelistet, wie man den Zugriff auf Daten beschränken und kontrollieren kann. Über die dort beschriebenen Möglichkeiten hinaus gibt es noch eine Reihe weiterer Techniken (z.B. Firewalls oder VPNs), die regulierend auf die Freigabe von Daten wirken, hier aber nicht weiter ausgeführt werden sollen.

Bei der Vergabe von Zugriffsrechten kann man zwei Wege gehen:

- Freischaltung des jeweiligen Zugriffs für alle mit anschließendem Ausschluss Einzelner. Dies hat zur Folge, dass die Liste all der Personen oder Personengruppen ständig gepflegt werden muss.
- Allgemeines Verschließen des Dokuments vor jedem mit anschließender Freigabe einzelner Personen oder Personengruppen. Diese Form der Freigabe wird am häufigsten verwendet.

Der »freigiebigste« Ansatz ist die Erlaubnis des anonymen Zugriffs, sei es nun lesend und/oder schreibend. Man kann keine weitere Einschränkung über einen anonymen Zugriff legen, da diese einer Bedingung entspräche, die vom Server geprüft werden müsste.

Die Identifizierung eines Clients erfolgt unter unterschiedlichsten Aspekten, aufgrund

- seiner Accountdaten oder seines digitalen Zertifikats,
- des Netzwerks oder der Domain, aus dem/der der Request erfolgt,
- des Zeitpunkts, zu dem der Request auftritt,
- anderer, seltenerer Identifikationsmerkmale, wie zum Beispiel Betriebssystem oder Browser-Variante des Clients.

Die Umsetzung von Regelungen über ACLs wurde bereits in Abschnitt 7.6.1 diskutiert. Für die Festlegung der Zugriffe auf den Dokumentenstamm sollte an dieser Stelle ein Audit stehen, in dem die Bereitstellung der Daten für den dafür bestimmten Personenkreis definiert wird. Als Ergebnis sollte sich ein Zugriffsregelwerk ergeben, aus dem die Access-Control-Liste für den Webserver abgeleitet werden kann.

7.7.4 Dokumentenpflege

Der Inhalt einer Webseite ist selten so gestaltet, dass er nicht ständig neu überarbeitet werden muss. Insbesondere die Verweise auf andere Seiten müssen ständig überprüft werden, da sich hier Änderungen ergeben können, die auf der eigenen Seite berücksichtigt werden müssen. Diese Änderungen betreffen die inhaltliche Aussage des verlinkten Dokuments oder auch dessen Verfügbarkeit. Aber auch der eigene Inhalt sollte ständig auf Aktualität überprüft werden. Oft finden sich Dokumente mit in die Zukunft gerichtetem Datum, das inzwischen längst abgelaufen ist.

7.7.5 Performance-Aspekte

Was immer auf einer Webseite präsentiert wird und wie umfangreich ihr Service-Angebot auch ist, sie hat zunächst die Anforderung zu erfüllen, innerhalb der ersten zehn bis fünfzehn Sekunden vollständig auf dem Client-Browser zu erscheinen. Dies ist der Zeitraum, den der durchschnitt-

liche Anwender bereit ist, auf den kompletten Download der Seite inklusive der Grafiken und Applikationen zu warten. Für den Entwickler einer Seite sollte damit oberste Priorität haben, dem in Frage kommenden Personenkreis die gewünschten Daten in der entsprechenden Zeit zu liefern.

Webseitendesign

Für die Gestaltung einer Webseite gibt es verschiedene Ansätze, die sich unterschiedlich auf die Darstellbarkeit auf Seiten des Clients auswirken. So ist nicht alles, was auf einer Webseite machbar ist, auch für eine performante Datenübertragung sinnvoll.

Da Webseiten in erster Linie Informationen liefern, die auf Texten basieren, werden ASCII-Zeichen des Betriebssystems genutzt. Selbst mit Formatierungsanweisungen werden Texte, die von einer Webseite über ASCII-Zeichen übertragen werden, performant übertragen.

Grafiken sind in Hinblick auf die Performance um einiges problematischer. Viele Webseiten sind damit überladen, was den schnellen Seitenaufbau verhindert. Damit soll nicht gesagt werden, dass man grundsätzlich auf grafische Elemente in Webseiten verzichten soll. Gerade diese haben ja stark dazu beigetragen, dass sich das Web in dieser kurzen Zeit so dramatisch entwickeln konnte. Man sollte sie aber für den Internetgebrauch optimieren.

Einige grundsätzliche Regeln sollte man beim Design einer Webseite beachten:

- Texte sollten als Texte übertragen werden, nicht als Grafiken.
- Grafiken müssen im richtigen Format bereitgestellt werden. So sind Zeichnungen und Skizzen eher im CompuServe Graphics Interchange-Format (GIF) sinnvoll, während Fotos im JPEG-Format vorliegen sollten. Ganz zu vermeiden sind Bitmaps (BMP und TIFF), die über keinerlei Komprimierungsmethoden verfügen.
- Bilder sollten vom Datenvolumen her klein sein. Als Faustregel kann man sagen, dass Bilder über 20 Kbyte zu groß sind und dass schon ein guter Grund vorliegen muss, um solche Grafiken in der Webseite einzubauen. Insgesamt sollten pro *html*-Seite nicht mehr als zehn grafische Elemente verwendet werden.
- Hintergrundbilder sind Geschmackssache und auch hier gilt: je kleiner, umso besser. Wichtig ist auch die Musterung. Je unruhiger eine Hintergrundgrafik ist, umso schlechter lässt sich der Inhalt der Seite lesen. Man kann auch ohne diese Technik (oder mit nur sehr geringen Mitteln) sehr ansprechende Webseiten erstellen.
- Nicht die Fülle der Bilder macht eine gute Seite aus, sondern deren Aufbau. Gleiches gilt auch für Animationen: Animated GIFs sind vielleicht interessant, machen aber den Seitenaufbau unruhig und lenken vom eigentlichen Inhalt ab.

- Werbebanner tragen unter Umständen zur Finanzierung der Seite bei. Interessanter machen sie sie aber in der Regel nicht, vielmehr wirken sie eher störend und lenken vom Inhalt der Seite ab.
- Applikationen sollten vermieden werden. Gerade die lange Wartezeit, die man für das Herunterladen von Applets benötigt, kostet Geld und Geduld der Anwender. Und Applikationen wie Uhren oder Counter gehören nur in den seltensten Fällen auf eine Webseite.
- Während vor einiger Zeit Frames noch eher unerwünscht waren, weil sie nicht von jedem Browser unterstützt wurden, kann man heute davon ausgehen, dass diese Technik Standard ist und verwendet werden »darf«.
- Cookies sollten vermieden werden. Das Misstrauen der Anwender vor diesen Datenschnipseln hat sich seit ihrer Einführung nicht geändert. Wenn die Funktionalität einer Webseite auf diesen Daten beruht, muss man sich überlegen, was man den Anwendern anbietet, die auf ihren Browsern die Verwendung von Cookies grundsätzlich abgeschaltet haben.
- Grundsätzlich sollte man nicht für einen bestimmten Browser programmieren. Ein gutes Webdesign sollte alle gängigen Browser auf allen verbreiteten Betriebssystemen unterstützen.

Ein einfacher Aufbau eines Webdokuments kann viel besser informieren als eine grafisch aufwändig aufbereitete Seite. Mit einigen gut platzierten Bildern, einem angemessenen Zeichensatz und einem klaren Seitenaufbau werden Webseiten anwenderfreundlich und gestatten einen schnellen Aufbau auf dem Client-Browser.

7.7.6 Implementierung eines Katalogs

Jede Information ist nur so gut, wie der Aufwand gering ist, sie zu finden. Gerade in der Fülle von Daten, die das Internet liefert, ist eine Katalogstruktur unentbehrlich. Eine Webseite wie Amazon.de wäre ohne Suchmaske, über die man gezielt nach Büchern suchen kann, nicht brauchbar. Entsprechend sollte im Gesamtkonzept der Implementierung einer digitalen Publikationsstruktur eine Volltextsuche oder zumindest ein Katalog nicht fehlen. Kapitel 8 wird sich sehr detailliert mit dem Aufbau eines Katalogdienstes auf einer Webseite befassen.

7.7.7 Monitoring

Ein Webserver bietet die Möglichkeit, Logfiles zu erstellen, in denen die Aktivitäten der Clients protokolliert werden. Diese Files ermöglichen es, im Nachhinein festzustellen, was im Falle eines Fehlers passiert ist oder welches Surfverhalten die Anwender an den Tag legen. Statistiken über Access-Files können helfen, die Attraktivität einer Seite zu verbessern, Error-Logs können

Aufschluss darüber geben, welche Funktionen zu einem Fehlverhalten führen oder aber ob sich Anwender unberechtigten Zugriff auf Dokumente verschaffen wollten.

Die Server der verschiedenen Hersteller gestatten es festzulegen, auf welchem Loglevel der Server Informationen dieser Art liefert. Eine umfangreiche Protokollierung der Aktivitäten auf dem Server unterstützt im Fehlerfall die Auswertung, sie generiert aber auf der anderen Seite ein nicht unerhebliches Datenaufkommen. Je nach Protokollierungsgrad werden mehr oder weniger viele Daten anfallen, was dazu führen kann, dass das Filesystem vollgeschrieben wird und im schlimmsten Fall der Server seinen Betrieb einstellen muss.

Damit dies nicht geschieht, gibt es unterschiedliche Strategien der Protokollierung. So kann man den Server veranlassen, ein Protokollfile täglich, idealerweise zu Zeiten der geringsten Last, abzuschließen und ein neues File anzulegen. Das abgeschlossene File kann über entsprechende Cron-Jobs (automatisch durchgeführte und regelmäßige Betriebssystemaktivitäten) archiviert und von der Arbeitspartition entfernt werden. Ein anderer Ansatz ist das Rotieren der Logfiles. Um unnötigen »Datenmüll« zu vermeiden, werden Logfiles sequentiell abgeschlossen und auf der Log-Partition hinterlegt. Nach Abschluss von *n* Files wird das jeweils älteste File überschrieben. Damit hält man die Menge an Protokolldaten überschaubar und hat immer noch ein Zeitfenster, in dem man eventuell aufgetretene Fehler nachforschen kann.

Neben dem Einsatz von Logfiles kann der Server im laufenden Betrieb überwacht werden. Bestimmte Aktionen lösen dabei einen Alarm aus, der den Administrator zu Aktionen veranlasst. Diese Systemüberwachungstools werden teilweise mit dem Server selbst, teilweise aber auch als eigenständige Produkte angeboten und helfen dem Betreiber einer Seite, dafür zu sorgen, dass das Webangebot ständig vorhanden ist und Fehler rechtzeitig gemeldet werden.

7.7.8 Backup und Recovery

Das Gesamtkonzept der Implementierung einer digitalen Publikationsumgebung wird durch ein sehr wichtiges Thema abgerundet, die Datensicherung. Bei der Umsetzung der im Gesamtkonzept vorgestellten Schritte in die Realität kann die Gesamtinvestition an Arbeitsaufwand und Geldmitteln von einer Sekunde auf die andere zunichte gemacht werden, wenn (warum auch immer) ein totaler Verlust an Hard- und Software sowie der Daten eintritt. Um diesem Szenario zu entgehen, verwendet man Datensicherungen. Diese sind so gestaltet, dass auch unter den unwahrscheinlichsten Umständen die Wiederaufnahme des Dienstes in endlicher Zeit und mit endlichen Mitteln umgesetzt werden kann, so dass der alte Status quo wieder hergestellt ist. Für die Publikationsarchitektur bedeutet dies nicht nur die Sicherung des Dokumentenbestands, sondern auch der Serverbinaries

und deren Konfiguration. Darüber hinaus kann man über besondere Hardware-Umgebungen sicherstellen, dass die Downtime des Systems minimiert wird. Es liegt im Ermessen des Anbieters der Daten, in welchem Rahmen und mit welchen Mitteln er diese Bereitstellung sichern möchte und wie schnell er nach einem Daten-GAU wieder betriebsbereit ist.

Zu berücksichtigen ist, dass sich die Priorität des Themas Backup/Recovery ständig ändern kann. So ist bei der Aufnahme des Betriebs ein Webserver sicher noch nicht so »bekannt«, als dass eine Downtime längeren Ausmaßes ins Gewicht fallen würde. Zu einem späteren Zeitpunkt aber können es wirtschaftliche Aspekte zwingend notwendig machen, dass das Thema Datensicherheit und die Wiederbeschaffung der Daten zu einem zentralen Punkt im Betrieb eines Webservers wird.

7.8 Session-Tracking

Eine *http*-Verbindung ist in den meisten Fällen nicht persistent, das heißt, ein Request, der an einen Webserver gestellt wird, erfolgt über eine Netzwerkverbindung, der Request des Clients wird übermittelt und vom Server beantwortet, indem dieser das Dokument (oder eine Fehlermeldung) übersendet. Anschließend wird die Netzverbindung abgebaut.

Diese Architektur wirft die Frage auf, wie man Webseiten umsetzt, die auf dem Inhalt und den Ergebnissen der vorherigen Seiten aufbauen. Beispielsweise kann man bei einem Einkauf in einem Online-Shop den virtuellen Einkaufskorb mit Waren beladen. Wie aber gelangt man mit diesen Informationen an die »Kasse«, da der Server bereits nach dem nächsten Klick auf einen Link alle vorherigen Einstellungen vergessen hat?

Nicht nur die Online-Shops haben ein Interesse, Session-Informationen auszuwerten. Gerade für die Marktforschung ist es interessant, wie sich ein Anwender auf einer Webseite bewegt und welche Aktivitäten er durchführt. In den folgenden Abschnitten werden die verschiedenen Möglichkeiten der Speicherung von Session-Informationen beschrieben.

7.8.1 URL-Rewriting

Ein Client, der zum ersten Mal eine Webseite anfordert, gibt zunächst noch keine Informationen über sich preis (mit Ausnahme des Browsertyps, des Betriebssystems, der IP-Adresse und weiteren Daten aus dem Attributfeld User-Agent des Client-Request im *http*-Protokoll). Damit eine Webseite die weiteren Aktivitäten des Users mitverfolgen kann, ohne stets eine Verbindung zum Client aufrechterhalten zu müssen, werden dem Client Daten mitgegeben, anhand deren sich der Webserver an frühere Verbindungen »erinnert«.

Damit die Erinnerung an den Client gelingt, übergibt der Server dem Client eine SessionID in Form einer Identifikationsnummer, die im Response des Servers enthalten ist. Bei jeder erneuten Verbindung des Clients mit dem Server wird diese ID in der URL mit übergeben.

Der Server seinerseits hält bis zu einem Timeout die SessionID zusammen mit den darin enthaltenen Daten im Speicher vor. Mit der übergebenen URL des Clients inklusive SessionID kann der Server nun jeden Client individuell identifizieren und die dieser Session zugehörigen Daten zuordnen.

Aber nicht nur SessionIDs können über URL-Rewriting die Sessionparameter übergeben. Vielmehr können Parameter und Attribute einschließlich ihrer Werte in der URL übergeben werden.

Der Nachteil dieser Methode ist, dass die Sessiondaten in Klartext in der URL übermittelt werden. Damit hat ein Angreifer eine Methode, mit der er einem Anwender schaden kann, das so genannte »URL-Hijacking«: Ein Angreifer kann eine URL verwenden, in der die SessionID eines anderen Users hinterlegt wurde. Auf diese Weise ist es beispielsweise möglich, dass ein Angreifer die Session einer Mailverbindung eines anderen übernimmt.

Das Verfahren der URL-Verwendung zur Übertragung von Sessiondaten wird in der expliziten Erläuterung der Anatomie einer URL in Kapitel 13 näher erläutert.

7.8.2 Versteckte Informationen: Hidden Fields

In einer Webseite können Felder eingesetzt werden, die bei der Darstellung der Seite nicht angezeigt werden. Diese versteckten Felder (Hidden Fields) werden damit vom Client nicht unmittelbar wahrgenommen, sie können aber dennoch Daten übertragen, die für den Webseitenbetreiber von Interesse sind.

Beispiel:

Eine Startwebseite wird so programmiert, dass sie in einer Formular-Umgebung eine SessionID des Benutzers einfügt, die von einem versteckten Feld zum nächsten beim Wechsel der Seiten übertragen wird. Es können auch Daten übermittelt werden, die verraten, von welcher Seite aus die Nachfolgeseite aufgerufen wurde. Beide Informationen geben dem Betreiber der Webseite interessanten Aufschluss über das Surfverhalten von Anwendern in seinen Seiten.

Ein solches Verfahren hat sogar ein eigenes kommerzielles Geschäftsmodell. Wenn beim Aufruf von bestimmten Webseiten plötzlich der Desktop mit Pop-up-Browserfenstern tapeziert wird, so hat das den Grund, dass der Betreiber dieser Seite für jeden Webaufruf auf eine andere Seite mit einer Pauschale bezahlt wird.

Beispiel:

Der Anwender besucht eine Seite für Hundezüchter. Der Betreiber dieser Seite programmiert die Seite so, dass mittels JavaScript-Aufruf weitere Fenster geöffnet werden, in denen Inhalte anderer Webserver angezeigt werden, beispielsweise Angebote für Hundefutter oder Tierbedarf. Das Geschäftsmodell sieht vor, dass in den Hidden Fields der Webseiten der Hundekuchenanbieter die Seite des Hundezüchters übergeben wird. Somit kann der Werbende identifizieren, woher der Anwender auf die eigene Webseite gelockt wurde. Für jeden »Kunden«, der auf diese Art zur Webseite des Werbenden geführt wird, kassiert der Betreiber der Anfangsseite einen Geldbetrag.

7.8.3 Cookies

Eine weitere Lösung für die Speicherung von Sessioninformationen sind »Cookies«. Diese Informationsschnipsel erinnern den Server an die Einstellungen und erlauben es bei einer *http*-Verbindung, die Informationen von einer Seite auf die nächste zu übertragen. Ein Vergleich aus dem realen Leben wäre eine Quittung der Garderobenfrau im Theater. Aufgrund dieses Stücks Papier wird die eigene Lederjacke später eindeutig wiedergefunden. Genauso wird auf Seiten des Browsers eine Information in der Webseite übergeben, die die Erinnerung des Servers nach dem nächsten Klick wieder auffrischt.

Cookies finden vielfach Anwendung. Ob man sich bei einem Online-Buchshop Literatur aussucht oder eine Portalseite eingerichtet hat, überall werden diese Informations-Snipplets eingesetzt, um den Inhalt den individuellen Client-Anforderungen anzupassen. Leider ist diese Technik etwas in Verruf geraten, weil damit nicht nur den Client-Anforderungen entsprochen wird, sondern auch Informationen unbeabsichtigt preisgegeben werden. So sammeln Agenturen wie Doubleclick diese Daten, um daraus ein Benutzerprofil bezüglich des Surfverhaltens zu erstellen. Diese Daten können dann an interessierte Unternehmen weiterverkauft werden, die dann sehr kundenorientiert eine Bewerbung mit ihren Produkten durchführen.

Inhalt der »Kekse«

Wie aber sehen diese Informationen aus und wo werden sie gespeichert? Zunächst einmal kommen diese Informationen in Form von spezifizierten Daten im Protokoll der *http*-Übertragung:

```
Content-type: text/html

Set-Cookie: dvd=The Matrix; path=/; expires Mon, 12-Jan-2001 08:12:12
GMT
```

Dieser Code wird vom Browser gelesen, der in seinem Memory diese Variablen (im Beispiel »dvd«, »path«, »expires«) mit den entsprechenden Werten (»The Matrix«, »/«, »Mon, 12-Jan-2001 08:12:12 GMT«) belegt. Ohne die explizite Angabe eines Verfallsdatums (Expire) ist die Lebensdauer eines

Cookies auf die Sessiondauer des Browsers beschränkt. Andernfalls werden die Daten in ein Textfile auf die Festplatte geschrieben. Hier gibt es eine obere Grenze: Beim Netscape Browser werden maximal 300 Cookies verwaltet, neu hinzukommende verdrängen die ältesten Daten. Der Microsoft Explorer speichert die Daten zusammen mit dem Cache in den »Temporary Internet Files«, deren Größe konfigurierbar ist. Die durchschnittliche Größe eines Cookies liegt bei etwa 100 Byte.

Ein Cookie beinhaltet mehr als nur einfach eine Variable und deren Wert. Vielmehr sind weitere Parameter vorgesehen, die folgendermaßen spezifiziert sind:

- Name=<Wert>

  Der Name des Cookies wird im Feld <Wert> belegt.

- Expires=<Datum>

  Ohne die Angabe eines Expiration-Zeitpunkts lebt das Cookie nur im Memory des Browsers und das auch nur so lange, wie der Browser selbst läuft. Ist dagegen ein Verfallsdatum gesetzt, das über die Laufzeit des Browsers hinausgeht, speichert dieser die Cookies in Textfiles auf dem Filesystem. Wird das Verfallsdatum überschritten, löscht der Browser eigenständig die Daten. Die Zeitangabe wird in Unix-Zeit angegeben, das heißt Anzahl der Sekunden seit dem 1.1.1970.

- Path=<Pfad>

  Der Gültigkeitsbereich eines Cookies kann durch die Spezifikation des Werts »Path« modifiziert werden. Wird es durch ein Dokument der URL www.meinefirma.com/marketing/ gesetzt, ist es per Default nur unter diesem Verzeichnis gültig. Soll es darüber hinaus verfügbar sein, lässt sich das über dieses Attribut festlegen: Path=»/« macht das Beispielcookie auch unter www.meinefirma.com/support/ sichtbar.

- Domain=<Domäne>

  Cookies können einzelnen Maschinen oder einer ganzen Domain zugewiesen werden. Voraussetzung ist allerdings, dass innerhalb der Domain-Angabe mindestens zwei Punkte vorhanden sind. Grund ist, dass die Domain-Angabe nicht missbraucht werden soll. Es ist also nicht möglich, im Namen einer anderen Domain ein Cookie zu setzen.

- Secure

  Mit diesem Attribut wird für den Wert »True« das Cookie nur verwendet, wenn es über eine SSL-Verbindung ausgetauscht wird, andernfalls nicht.

Sicherheitsbetrachtungen

Um es vorweg zu sagen: Man kann den verwendeten Browser veranlassen, keine Cookies anzunehmen. Damit kommt man nicht in den Genuss der personalisierten Interaktion mit dem Server, ist sich aber auch sicher, dass über

diesen Weg keine persönlichen Informationen preisgegeben werden. Einen weniger radikalen Weg verfolgen Strategien, dass persönliche Informationen nur dem Server mitgeteilt werden, von dem sie zuvor gesetzt wurden. Damit sollte die Auswertung dieser Daten durch Dritte erschwert werden. Andere Strategien sehen die selektive Akzeptanz von Cookies aus bestimmten Domänen vor. Ist eine Domäne nicht in der Liste der akzeptierten Server, kann sie keine solchen Informationen platzieren. Den unbeschwertesten Weg wählt man, wenn man von jedem Server alle Cookies akzeptiert.

Cookies können nur von Servern aus der Domain ausgelesen werden, aus der sie geschrieben wurden.

Beispiel: Eine Server-Applikation aus der Domain »www.herpes-opfer.de« setzt im Browser von Joe User ein Cookie mit der sinngemäßen Information »User hat die Seite ‚Hilfe bei Herpes' angeschaut, seine Mailadresse lautet ‚beispiel.user@irgendeine.domain.de'«. Surft der User auf eine andere Webseite, beispielsweise »www.profit-mit-medikamenten.de«, so kann diese Webseite (laut Spielregeln für die Implementierung von Cookies in Browsern) das Cookie der vorherigen Domain nicht auslesen. (Letztere hätte ein kommerzielles Interesse an einer solchen Information, da sie mit der Mailadresse den User direkt und zielgerichtet bewerben könnte).

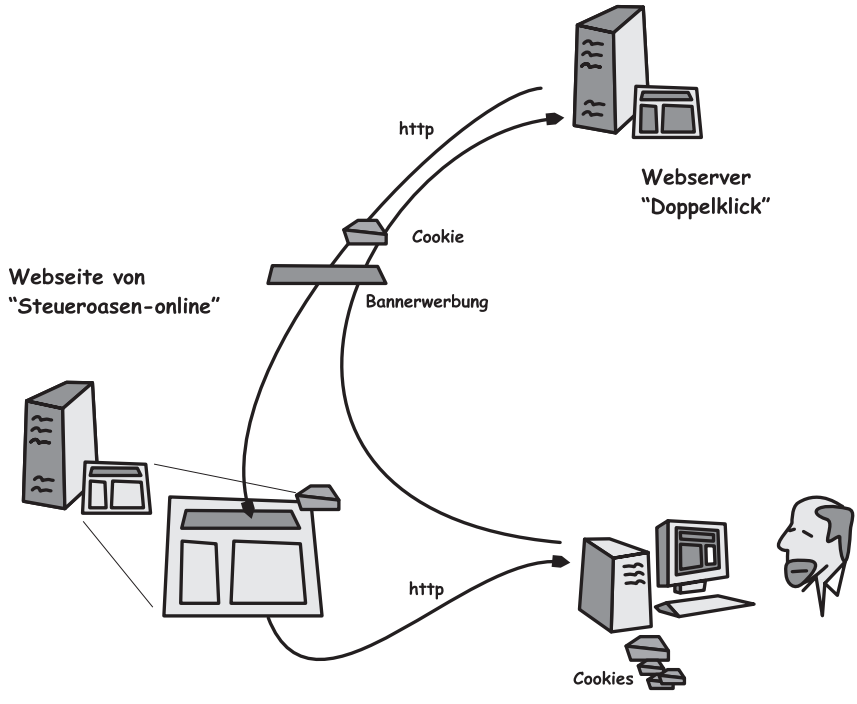


Abbildung 7.16: Ermittlung des Surfverhaltens von Anwendern durch Cookies

Mit Cookies allein lässt sich Benutzerverhalten noch nicht auswerten, gäbe es nicht die Fremdverlinkung über Bannerwerbung: Angenommen, auf der Seite »www.herpes-opfer.de« befindet sich ein Banner der Firma »Dop-

pelklick.com«. Doppelklick bezahlt für die Platzierung einen monatlichen Betrag. Joe User, der nun auf »www.herpes-opfer.de« gelangt, lädt damit zusätzlich von »Doppelklick.com« das Bannerlogo. Außerdem gelangt ein Cookie von »Doppelklick.com« auf seinen Browser, dessen Inhalt etwa folgendermaßen aussehen könnte:

```
Content-type: text/html

Set-Cookie: userid=0815-4711-herpes-opfer.de; path=/; expires Mon, 12-
Jan-2050 08:12:12 GMT
```

Der Inhalt seines Cookie-Files würde durch diesen Aufruf beim Netscape Communicator um einen Eintrag erweitert werden, der folgendermaßen aussehen könnte:

```
doppelklick.com     TRUE / FALSE     7145801453
userid    0815-4711-herpes-opfer.de-akute_hilfe
```

Surft er nun auf andere Seiten im Internet, die ebenfalls Bannerwerbung der besagten Firma beinhalten, so darf der Server, der die Bannerwerbung liefert (also der Server der Firma »Doppelklick«) den Inhalt sämtlicher von ihr gesetzten Cookies lesen. Insbesondere findet der Betreiber dieser Domain heraus, dass der gleiche User, der vorher auf der Seite von »herpes-opfer.de« war und die User-ID »0815-4711-herpes-opfer.de« hat, nun auf dieser anderen Seite vorbeikommt. Und so kommen weitere Cookies von »Doppelklick.com« über die entsprechende Bannerwerbung hinzu, je nachdem, auf welcher Seite das Banner platziert ist:

```
.doppelklick.com     TRUE / FALSE     7145801453
   userid    0815-4711-usa_reisen-online.com-wohnmobile
.doppelklick.com     TRUE / FALSE     4391049384
   userid    0815-4711-buchshop.com-whisky_buecher
.doppelklick.com     TRUE / FALSE     4391049384
   userid    0815-4711-buchshop.com-scheidungsrecht
.doppelklick.com     TRUE / FALSE     4391049384
   userid    0815-4711-buchshop.com-aktien_und_schwarzgeld
.doppelklick.com     TRUE / FALSE     4391049384
   userid    0815-4711-buchshop.com-steueroasen
.doppelklick.com     TRUE / FALSE     8374656830
   userid    0815-4711-herpes-opfer.de-akute_hilfe
```

Alle diese Informationen dürfen nun von »Doppelklick.com« ausgewertet werden, da sie ja offiziell von dieser Firma gesetzt wurden. Nun ist es möglich, über die User-ID das Surfverhalten von Joe User im Internet nachzuvollziehen und ein Persönlichkeitsprofil zu erstellen. Ist nur eine der Firmen, die die Bannerwerbung von »Doppelklick.com« anbietet, bereit, persönliche Daten des Benutzers herauszugeben, kann diese User-ID einer realen Person zugeordnet werden. (»herpes-opfer.de« sammelt beispielsweise E-Mail-Adressen für Mailinglisten und gibt sie insgeheim an »Doppelklick.com« weiter.) Somit kennt »Doppelklick.com« das Surfverhalten von Joe User inklusive seiner Mailadresse bzw. sogar detaillierteren Personendaten und kann diese Informationen gewinnbringend weiterverkaufen.

7.9 Absichern einer Webseite

Viele Webseiten im Internet sind sehr bekannt und deshalb oft Ziel von Angriffen, sei es auf den Inhalt der Seite oder auf die Erreichbarkeit.

Die Gründe solcher Aktionen sind vielfältiger Natur. Das Spektrum reicht von politischen Motiven bis hin zu reinem Vandalismus. Die Herausforderung, eine solche Seite zu »knacken« oder nicht mehr erreichbar zu machen, wird durch die Berichterstattung in den Medien belohnt. Was für die Angreifer oft nur »Spaß« bedeutet, kann für das angegriffene Unternehmen oder die Personen, die auf diese Informationen angewiesen sind, einen enormen wirtschaftlichen Schaden bedeuten.

- Ein Unternehmen, das über das Internet seine Bücher verkauft und darauf angewiesen ist, dass seine Seite fehlerfrei und immer erreichbar ist, kann binnen Stunden hohe Verluste erleiden, wenn die Webseite nicht erreichbar ist.
- Ein Finanzportal, das Wirtschaftsdaten veröffentlicht, beeinflusst mit diesen Daten die Kaufentscheidungen vieler Aktionäre. Ein Angriff auf eine solche Seite könnte falsche Unternehmensdaten einschleusen, um so einen Aktienkurs zu manipulieren. Daraus kann wiederum der Angreifer seinen Gewinn ziehen.
- Eine Webseite, deren ursprünglicher Inhalt durch einen offensichtlichen oder versteckt falschen Inhalt ersetzt wird, kann der Reputation des Eigentümers schaden. Beispiele zu diesen »gehackten« Seiten werden im Internet in einschlägigen Seiten aufgelistet. Besonders »gelungene Hacks«, in denen der Inhalt der Seite parodiert wird, rufen vielleicht nur Schadenfreude beim Webseitenbesucher hervor. So wurden die Internetpräsenzen diverser Automobilfirmen oder Künstler von Angreifern so offensichtlich modifiziert, dass sie – zwar für den Besitzer ärgerlich – keinen weiteren Schaden zufügten. Das Einspielen der originalen Seite und das Verschärfen der Sicherheit beheben in der Regel das Problem.

  Wird aber der Inhalt einer Seite unter Beibehaltung des Look-and-Feel versteckt modifiziert, so dass dem Betreiber der falsche Inhalt erst sehr viel später auffällt, kann der Schaden weitaus größer sein. Die Spanne der Beispiele reicht von Modifikationen von Preislisten bis hin zu veränderten Informationen auf Webseiten von politischen Parteien. In beiden Fällen wird der Besucher dieser Webseite getäuscht, er bringt den falschen Inhalt der Seite mit dem Besitzer in Zusammenhang und zieht daraus seine Schlüsse.

Wie aber werden solche Angriffe auf Webseiten möglich? Wir haben in den vorangegangenen Beispielen die verschiedenen Aspekte von Angriffen gesehen und können zwischen den folgenden Arten unterscheiden:

- Modifikation von Inhalten: Hier wird der originäre Inhalt der Webseite durch einen anderen ersetzt. Die Aktion wird durch jemanden durchgeführt, der nicht vom Besitzer der Seite beauftragt wurde.

  Bei einer Vielzahl von Webseiten ist es relativ einfach, den Inhalt auszutauschen: Das *http*-Protokoll hat eigene Kommandos, mit deren Hilfe man Inhalte auf einem Webserver hinterlegen oder ersetzen kann. Diese wurden bereits in Abschnitt 7.3.1 vorgestellt: Mit »PUT« wird eine referenzierte Datei auf dem Webserver hinterlegt, ist eine Datei gleichen Namens bereits vorhanden, wird diese überschrieben. Mit »DELETE« wird eine referenzierte Datei gelöscht. Beide Befehle stammen aus der Version 1.1 des *http*-Protokolls und sollten aus Sicherheitsgründen vom Webserver unterbunden werden.

  Aus Sicht eines Angreifers wird die Modifikation einer Webseite schwieriger, wenn die *http*-Unterstützung verhindert wird. Er muss sich einen Zugang auf das Filesystem des Webservers verschaffen und die Webdokumente direkt modifizieren. Für diese Art des Angriffs müssen andere Sicherheitsmaßnahmen ergriffen werden. Diese sehen den Einsatz von Reverse Proxys in Verbindung mit wohl definierten Routing-Regeln durch die dahinter liegende Firewall vor: Der Inhalt der Webseite liegt auf einem Server hinter der Firewall, ein Zugriff durch die Firewall darf nur durch den davor liegenden Proxy erfolgen. (Eine solche Konfiguration würde einen Hacker herausfordern, den Reverse Proxy zu übernehmen und über diesen den hinter der Firewall liegenden Server zu modifizieren. So ergeben sich Maßnahmen und Gegenmaßnahmen, deren Beschreibung den Rahmen dieses Buchs sprengen würde.)

- Umleiten von Webseiten: Bei einem solchen Angriff wird die originäre Webseite nicht modifiziert, vielmehr werden die Zugriffe umgeleitet. Ein Angreifer baut eine Webseite auf, deren Inhalt er anschließend unter der URL der anderen Seite anbietet. Die ursprüngliche Seite ist nicht mehr erreichbar, nach außen hin wird die gefälschte Seite offeriert. Für den User, der diese Webseite ansteuert, gibt es zunächst keine Möglichkeit herauszufinden, ob es sich bei der angebotenen Seite um das Original oder die Fälschung handelt.

  Beispiel: Der User steuert die Webseite seiner Bank an, um über Internetbanking seine Überweisungen zu tätigen. Ein Angreifer hat zuvor den kompletten Inhalt der Seite kopiert und baut auf einem eigenen Webserver mit einer eigenen IP-Adresse die Seite der Bank weitgehend identisch nach. Schließlich gelingt es ihm, durch Manipulationen im DNS die Namensauflösung von »www.meine-bank.de« von der ursprünglichen IP-Adresse der Originalseite hin zur gefälschten Seite umzuschreiben. Damit werden User, die nichtsahnend ihren gewohnten Link zur Hausbank verfolgen, eine Seite vorfinden, die in ihrem Aussehen identisch mit der Originalseite ist, sich aber in Händen einer vollkommen anderen Person befindet. Den Manipulationen stehen nun alle Wege offen: Der Angreifer kann die PINs und TANs der Anwender abfangen und sie für

seine eigenen Zwecke missbrauchen. (Wir haben bereits gesehen, dass ein solcher Angriff über den Einsatz eines serverseitigen digitalen Zertifikats unterbunden wird. Voraussetzung ist aber, dass die Zertifizierungsinformationen, die beim Zugriff auf eine abgesicherte Webseite auf dem Monitor des Users erscheinen, aufmerksam gelesen werden und die Richtigkeit der Daten überprüft wird. Leider ist es eher die Praxis, dass diese Informationen als Ärgernis empfunden und ohne Verifikation weggeklickt werden.)

Die Umlenkung des Request eines Users auf eine gefälschte Seite wird durch Modifikation des DNS-Systems erreicht. Hierbei macht man sich den Umstand zunutze, dass die Namensauflösungen in IP-Adressen nicht nur zentral vorgehalten, sondern in diversen Caches zwischengespeichert (gecached) werden. Kann der Angreifer nun einen DNS-Cache, der die originale Namensauflösung beinhaltet, ändern und die IP-Adresse zugunsten seiner eigenen IP-Adresse austauschen, werden alle Anwender, die über diesen DNS-Cache die in Frage kommende Seite aufsuchen, umgeleitet. Einen solchen DNS-Cache, der auf diese Weise modifiziert wurde, nennt man »vergiftet«. Ein ähnliches Szenario wurde bereits in Kapitel 6.6 beschrieben.

- Denial of Service (DoS): Wird eine Webseite gleichzeitig mit Tausenden von Anfragen bombardiert, stellt der Webserver irgendwann seinen Dienst ein und steht nicht mehr zur Verfügung. Eine Möglichkeit ist die Verwendung des »ping«-Befehls: Mit »ping <host>« kann die Erreichbarkeit eines Rechners über das Netzwerk überprüft werden.

  ```
  >ping www.meinefirma.com
  www.meinefirma.com is alive
  ```

  Nun kann der Ping-Befehl mit dem Flag »-f« dazu gebracht werden, sofort nach der Antwort des Zielrechners erneut eine Ping-Abfrage zu starten. Dies kann den Zielrechner unter Umständen bereits ziemlich stressen und die Erreichbarkeit für andere Clients und Dienste erschweren. Dies ist der Grund, warum Ping-Befehle bei Firewalls von vorneherein auf der Liste der zu blockenden Dienste stehen. Die beabsichtigte oder ungewollte Verwendung einer solchen Technik zu Ungunsten Dritter wird als Straftat angesehen.

- Einbruch in Webseiten: Seiten, die über User-ID und Passwort geschützt werden, können über einen verteilten Angriff angegriffen werden. Der geläufigste Angriff ist der so genannte Brute-Force-Angriff: Mit Wörterbuchattacken werden alle möglichen Kombinationen ausprobiert. Um die Identität des Angreifers zu verbergen, wird der Brute-Force-Angriff nicht direkt durchgeführt, sondern parallel über eine Liste von öffentlichen Proxyservern verteilt.

Alle diese Angriffe werden täglich im Internet durchgeführt. Nur eine sorgfältig geplante Sicherheitspolitik schützt die eigene Webseite vor diesen und weiteren Attacken.

7.10 Zusammenfassung

Dem Internet gelang der Durchbruch über das World Wide Web. Das Surfen auf Webseiten steht bei Anwendern an erster Stelle, wenn sie sich zu Hause einen Internetanschluss auf ihrem Computer installieren. Mittlerweile ist fast jede Nachrichtensendung im Fernsehen und jede Werbefläche mit einer URL versehen, über die man sich nähere Informationen einholen kann.

Seit den ersten Webseiten haben sich eine Vielzahl von Technologien rund um *html* entwickelt, die eine sehr große Vielfalt an Programmiermöglichkeiten bieten. Man kann neben statischen Informationsseiten Applikationen mit neuen Dienstleistungen anbieten. So ist es möglich, von zuhause aus Bankgeschäfte zu tätigen, mit Aktien zu handeln, Behördengänge zu erledigen oder Waren einzukaufen.

Mit diesen Möglichkeiten wachsen aber auch die Anforderungen an die Sicherheit des Mediums. Kommerzielle Seiten benötigen eine verlässliche Absicherung gegen Missbrauch. Dadurch entwickeln sich auch andere Bereiche weiter, angefangen von der Kryptografie bis hin zur Benutzerdatenhaltung.

Unternehmen tragen die Verantwortung, die Datensicherheit benutzerfreundlich und sicher umzusetzen, da der durchschnittliche Anwender nicht mehr in der Lage ist, jeden Aspekt der Kommunikation im Internet zu durchschauen und Sicherheitslücken zu umgehen.

Aus Unternehmenssicht ist es ein Vorteil, dass die über einen Webserver publizierten Dokumente mit geringem Aufwand stets aktuell gehalten werden können. Man erspart sich Druckkosten und unterstützt die Aktualität der Daten.

Im Vergleich zu anderen Medien bietet das Web eine Möglichkeit, sich die interessanten, wichtigen Informationen direkt und zu jeder Zeit herauszusuchen. Über geeignete Suchmechanismen ist man in der Lage, diese Daten zu finden. Mit diesem Thema befasst sich das nächste Kapitel.

8 Suchmaschinen und Kataloge

Wie findet man im Internet Informationen? Viele Internet Provider unterstützen den Neuling im Netz und liefern ihm einen Browser mit Suchmaske aus. In dieser kann einen Suchbegriff eintragen und die zugrunde liegende Applikation bietet daraufhin eine Sammlung von weiterführenden Verweisen. Erfahrenere Anwender kennen »ihre« Suchmaschinen und nutzen automatisch oder über Bookmarks die einschlägigen Dienste. Was sich hinter einer solchen Eingabemaske verbirgt, ist Thema der folgenden Abschnitte.

8.1 Einführung

Im Internet finden sich gerade im World Wide Web eine Fülle von Servern, die ihren Inhalt über eine eigene URL anbieten. Diese Bereitstellung der Daten erfolgt praktisch ungeordnet. Natürlich werden die Rechner über Domains und IP-Adressen geordnet, der Zugriff auf den Inhalt einer Webseite erfolgt aber ohne jede weitere Hierarchie. Man ist gewohnt, dass ein Webserver auf dem Port 80 arbeitet, dies ist allerdings nicht zwingend vorgeschrieben. Gleiches gilt für den Zugriff auf die Webseite eines kommerziellen Anbieters. Man erwartet, dass der Server »www« heißt und dass es eine Domain gibt, die mit dem Herstellernamen im Einklang steht, aber auch dies ist nirgends vorgeschrieben. Insbesondere wenn es mehrere Hersteller gleichen Namens gibt, wird man nicht immer die richtige Seite des gesuchten Anbieters treffen.

Suchmaschinen im Internet helfen einem Benutzer. Dieser kann über eine Eingabemaske einen Begriff spezifizieren und nach kurzer Suche steht eine Auswahl an möglichen Zielen bereit.

Für einen persönlichen schnellen Zugriff auf Daten, die man bereits »irgendwo mal gesehen« hat, wurden Lesezeichen (Favoriten oder Bookmarks) eingeführt, die es dem User erlaubten, sich bestimmte Seiten zu merken. Man baute sich eine Liste von Bookmarks auf und bekam ein Gefühl dafür, wo in etwa man bestimmte Informationen suchen konnte.

Die Weiterentwicklung des Internets wäre aber sicherlich gescheitert, hätte man nicht einen zentralisierten Server bereitgestellt, der es einem Anwender erlaubte, eine Suche über einen Datenbestand von Bookmarks auszuführen, um eine Liste von Links zu erhalten, die zu der gewünschten Information führten.

Grundsätzlich muss man zwischen einer Suchmaschine und einem Katalog unterscheiden. Beide erfüllen zunächst einmal den Zweck, einem Anwender Hilfestellung zu leisten, wenn er bestimmte Informationen sucht. Der grundlegende Unterschied zwischen den beiden Systemen ergibt sich aus der Vorgehensweise:

- Eine Suchmaschine bietet eine Eingabemaske, in der der Anwender den gewünschten Begriff eintragen und eine Suche starten kann, die – wie beschrieben – eine Liste von Links liefert, die den Begriff in irgendeiner Art enthalten. Bei der Volltextsuche wird der gesamte Text des ausgewählten Bestands durchsucht. Taucht der Begriff auf, wird der Link angezeigt. So ist, je nach Begriff, die Ergebnismenge mehr oder weniger brauchbar. Beispiel: Die Suche über die Wörter »Formel« und »Eins« kann zu Themen wie Rennsport führen, aber auch zu Zahnpastareklame oder wissenschaftlichen Abhandlungen.

  Um solche Ergebnisse beurteilen zu können, liefern die Suchmaschinen nicht nur einfach Links zu den Themen. Sie versuchen außerdem, den Kontext, in dem der gesuchte Begriff gefunden wurde, in Form von mehreren Zeilen Information zu beschreiben.

- Ein Katalog ist ein Dienst mit vorsortierten Themen, der Begriffe in einem Kontext liefert. Um bei dem vorangegangenen Beispiel zu bleiben, würde ein rennsportbegeisterter Anwender in einem Katalog die Kategorie »Sport« suchen, die weiter in »Motorsport« und »Formel 1« unterteilt ist.

  Die Themen wie die Zuordnungen werden meist redaktionell erstellt, um die thematische Zuordnung einer Webseite im richtigen Kontext sicherzustellen. Manche Kataloge erstellen diese Kategorien dadurch, dass ein intelligentes Programm, ein so genannter Robot, eine Webseite analysiert und anhand von Schlüsselwörtern beurteilt, in welcher Kategorie eine Webseite einzuordnen ist. Weiterhin können Webseiten-Programmierer ihre Seiten bei den Internet-Katalogen direkt in den entsprechenden Kategorien anmelden. Zusätzlich sind Moderatoren einzelner Kategorien laufend damit beschäftigt, Korrekturen dieser Sammlungen vorzunehmen.

  Die Katalog-Anbieter haben die Möglichkeit, bestimmte Seiten zu priorisieren, indem diese in einer Kategorie sehr weit oben stehen. Dadurch werden Anwender bei einer Suche schnell auf die wichtigsten Seiten gelenkt, andererseits besteht hier die Gefahr einer Beeinflussung seitens des Moderators dieser Kategorie.

Suchmaschinen und Kataloge sind im Ansatz und in der Bereitstellung der Dienstleistung vollkommen unterschiedlich. Die eine führt eine Suche über einen Datenbestand aus, der andere stellt bereits sortierte Themen über ein Linkverzeichnis bereit.

Im Internet haben sich inzwischen eine Reihe verschiedener Suchmaschinen etabliert, die bei Recherchen unterschiedliche Ergebnisse liefern können. Diesen Umstand machen sich so genannte Metasuchmaschinen zunutze,

indem sie einen Suchbegriff entgegennehmen und an eine Reihe von verschiedenen Suchmaschinen weitergeben. Die Ergebnismenge wird anschließend zusammengefasst und aufbereitet, so dass keine mehrfachen Ergebnisse aufgelistet werden, und das Resultat wird dem Anwender übergeben. So kann ein solcher Dienst beispielsweise den Suchbegriff an Yahoo, Altavista und Lycos weitergeben und die Ergebnismenge einsammeln und bereitstellen.

Schließlich gibt es noch die Kategorie der spezialisierten Suchmaschinen, die sich ausschließlich auf ein bestimmtes Themengebiet konzentrieren.

Die folgenden Abschnitte werden die Architektur einer Suchmaschine genauer beschreiben.

8.2 Architektur einer Suchmaschine

Die im Internet betriebenen Suchmaschinen basieren nicht auf Standardprodukten, sondern sind überwiegend eigene Entwicklungen. Dies liegt daran, dass es keinen großen Markt für ein solches Produkt gibt. Trotzdem existieren viele Gemeinsamkeiten bei diesen Architekturen und es ist interessant, wie eine solche Maschine arbeitet.

Die hier beschriebene Architektur basiert weder auf einem bestimmten Produkt noch auf einer bestimmten Suchmaschine. Vielmehr soll eine realitätsnahe Architektur vorgestellt werden, die die Arbeitsweise eines solchen Dienstes verdeutlicht.

8.2.1 Übersicht

Ein Suchdienst arbeitet nach dem folgenden Prinzip:

- Ein oder mehrere Automaten (Robots, Spider, Crawler) bekommen eine Startseite zugewiesen. Diese wird von ihnen sowohl auf Links wie auf Schlagworte hin untersucht. Die gefundenen Begriffe sowie die Adresse dieser Seite werden in einer Datenbank hinterlegt.
- Nachdem der Robot die Seite verschlagwortet hat, liegt eine Liste von Links vor, die von dieser Seite aus verzweigen. Diese Links werden vom Robot weiterverfolgt und ebenfalls indiziert. Der Robot unterscheidet dabei zwischen lokalen und externen Verweisen. Die lokalen Links werden zuerst abgearbeitet, bevor externe verfolgt werden.

Damit werden ganze Webseiten systematisch untersucht und verschlagwortet, jedes verlinkte Webdokument wird gefunden und in die Suchmaschine aufgenommen. Dokumente, die nicht verlinkt sind, werden nicht gefunden.

Der Betreiber kann bestimmen, welche Domänen untersucht und wie viele Links auf einer Seite akzeptiert werden. Über ein Bewertungsverfahren wird einer gefundenen Seite eine Priorität zugeteilt, die ihre Platzierung in einer Ergebnisliste von Treffern festlegt.

Folgende Funktionen gehören zu einer Basisarchitektur: ein URL-Server, der die Arbeitszeit des Prozesses bestimmt (damit dieser beispielsweise bei einer Webseite eines Unternehmens nicht den laufenden Betrieb stört). Eine Datenbank nimmt die Schlagwörter und die damit verbundenen Adressen auf. Ein Suchinterface bietet dem Interessenten eine Möglichkeit zur Formulierung einer Abfrage. Ein Bewertungsmaßstab wertet die Relevanz eines Schlagworts in einer Seite aus, auf der es auftritt.

Eine Suchmaschine hat eine Architektur mit vielen Komponenten, die alle aufeinander abgestimmt sein müssen.

8.2.2 Architektur

Eine Suchmaschine kann grob in drei Komponenten unterteilt werden:

1. Datenbeschaffung

   Der Datenbestand im Internet wird ständig erweitert. Damit die Datensammlung der Suchmaschine ständig aktualisiert wird, durchkämmen selbständige Prozesse, so genannte Robots, permanent das Netz und liefern die gefundenen Daten in der Datenbank ab. Die Begriffe »Robot«, »Spider«, »Crawler« oder »Worm«, die alle die gleiche Funktionalität beschreiben, sind etwas irreführend. Sie erwecken den Eindruck, als würden sie sich auf den Webseiten bewegen und von Zeit zu Zeit ihren Inhalt am zentralen Server abliefern. In Wirklichkeit ähnelt ein solcher Prozess eher einem Webbrowser, der über eine URL ein Dokument auf einem Server anfordert. Aus der Fülle der verschiedenen Begriffe werden wir hier »Robot« verwenden.

2. Datenhaltung

   Die von den Robots gelieferten Daten werden aufbereitet, verschlagwortet und in einer Datenbank hinterlegt. Man schätzt, dass eine gute Suchmaschine den Inhalt von etwa 40% des Datenbestands im Internet abdeckt.

3. Präsentation

   Über ein Webinterface wird dem Anwender eine Möglichkeit gegeben, ein Suchkriterium zu spezifizieren. Die nachfolgende Suche über den Datenbestand der Datenbank liefert eine Ergebnismenge, die dem Anwender in Form einer Liste angezeigt wird.

Robots

Eine der wichtigsten Funktionen einer Suchmaschinen ist die Beschaffung von Informationen über Dokumente, die im Internet verfügbar sind. Dabei kommen nicht nur *html*-Dokumente in Betracht, sondern je nach Anbieter auch reine Textdateien, PDF- und Word-Dokumente. Bilder, Grafiken oder andere nicht textorientierten Daten werden nicht berücksichtigt. Theoretisch machbar wäre die Berücksichtigung von Archiven wie Tar- oder Zip-

Dateien, in der Praxis aber ist die Dekomprimierung und die anschließende Auswertung zu zeitaufwändig, als dass sie Berücksichtigung in heutigen Robot-Implementierungen findet.

Ein Robot kann nicht mit geschützten Seiten umgehen, da diese Seiten nur über eine Authentifizierung erreichbar sind. User-ID/Passwort-Kombinationen oder digitale Zertifikate stehen einer Suchmaschine nicht zur Verfügung. Entsprechend bleiben die »inneren« Webseiten eines geschützten Serverbereichs unberücksichtigt. Unberücksichtigt bleiben auch die Inhalte dynamischer Webseiten, die über Funktionen oder Applikationen erzeugt werden.

Da jeder Robot gleichzeitig eine Reihe von Webdokumenten anfordert, werden eine hohe Zahl paralleler Verbindungen aufgebaut, die über eine gut ausgebaute Netzwerkinfrastruktur abgefangen werden müssen. Der parallele Download vieler Seiten erfordert eine genügend große Bandbreite, um alle Daten zu übermitteln. Darüber hinaus muss für die permanenten Namensauflösungen eine optimierte DNS-Architektur vorhanden sein.

URL-Server

Der Robot wird von einem so genannten URL-Server gesteuert. Dieser hält die Queue der Links vor, die abgearbeitet werden soll. Ist eine Seite geladen, bekommt der Robot eine neue URL übertragen. Die Queue ergibt sich aus den Daten bereits besuchter Webseiten. Einfache Suchmaschinen arbeiten mit unkontrollierten Robots, die ohne Strukturen und Präferenzen Links akkumulieren, um sie im Laufe der Zeit zu bearbeiten. Aufwändigere Maschinen definieren dem Robot eine Vorgabe, um auf bestimmte Domänen oder Subnetze Schwerpunkte zu setzen. Im Grenzfall geht dies bis hin zur Einschränkung auf eine einzelne oder einige wenige Domänen bzw. eine Liste von konfigurierten URLs. Eine Suchmaschine für »meinefirma.com« würde ausschließlich in dieser Domain operieren.

Über weitere Spezifikationen könnte man bestimmte Seiten (z.B. www.finance.meinefirma.com) oder Dokumente (*.doc, *.txt, ...) von der Indizierung ausschließen.

Der Inhalt des URL-Servers wird aus der Datenbank erzeugt. Diese stellt eine Liste von Verweisen bereit, die entweder neu hinzugekommen sind oder bei denen eine Überprüfung der Aktualität des damit verknüpften Dokuments ansteht.

Der URL-Server funktioniert mit einem Policy-Manager, der

- verhindert, dass die eingesetzten Robots eine übermäßige Last im Netzwerk oder auf einem Server erzeugen.
- Schwerpunkte auf bestimmte Domains setzen kann oder bestimmte Domänen ausschließt, die aus politischen, juristischen oder moralischen Gründen nicht erwünscht sind.

- das Robot Exclusion Protocol berücksichtigt. Dabei handelt es sich um ein Verfahren, mit dem Webmaster den Zugriff von Robots auf ihre Seite kontrollieren können. Dadurch können Webadministratoren den Robot anweisen, welche Bereiche der Seite indiziert werden sollen und welche nicht.
- Dokumenttypen ein- oder ausschließen kann.

Die noch abzuarbeitenden Links des URL-Servers stammen aus der Datenbank und sind vom Policy-Manager akzeptiert worden. Der Robot bekommt aus der Queue des URL-Servers eine neue URL zugewiesen, die er kontaktieren soll. Der Inhalt der zugewiesenen Seite wird nur dann untersucht, wenn sich die Seite seit dem letzten »Besuch« nicht verändert hat. (Der Request basiert wie beim Proxy-Cache auf der HTTP-Syntax »get-if-modified-since«, siehe Kapitel 10.) Die Datumsinformation kommt aus dem bereits vorhandenen Datensatz zu dieser URL.

Datenbank

Nachdem ein Robot ein Dokument gefunden und geladen hat, wird die Seite analysiert und das Ergebnis abgespeichert. Es werden Titel, Metadaten und Texte verschlagwortet. Dagegen werden Bilder oder Videos, Formatierungs- oder Tabellenanweisungen nicht berücksichtigt und verworfen. Der Umfang der Verschlagwortung hängt vom Dienst ab. Manche nehmen nur die Titel und die Metainformationen, für andere sind die ersten Absätze eines Dokuments relevant, wiederum andere berücksichtigen den gesamten Inhalt einer Seite. Letzteres führt zu riesigen Datenbanken, die den enormen Aufwand an Informationen speichern.

Für eine schnelle Performance werden die Daten indiziert. Dazu werden Teile der Datenbank in Form von neuen kleineren Index-Datenbanken hinterlegt, die das Vorkommen einzelner Begriffe bereitstellen. Man bekommt eine bessere Suchperformance, weil nicht mehr die gesamte Datenbank durchforscht werden muss, sondern nur noch der Index. In der Regel wird dabei der Kontext nicht berücksichtigt, das heißt, der Begriff »Lager« unterscheidet nicht zwischen »Kugellager« und »Zeltlager«.

Manche Indizierungen berücksichtigen Singular- und Pluralkonstrukte. So kann eine Suche nach »Taschen« ebenso auch auf Seiten führen, auf denen »Tasche« vorkommt.

Um die Menge der zu verwaltenden Daten zu reduzieren, werden für die Verschlagwortung Artikel, Adjektive oder Pronomen vernachlässigt, da diese so genannten Stopwörter für die inhaltliche Klassifizierung eines Dokuments nicht relevant sind.

Die Speicherung in der Datenbank erfolgt unterschiedlich. So werden die Inhalte der gefundenen Seiten im *Nordic Web Index* über einen eigenen Prozess in eine XML-ähnliche Syntax gebracht, die anschließend in der Datenbank als Seitenbeschreibung hinterlegt wird. Andere speichern nicht die Worte eines Textes, sondern deren Vorkommen und Position im Dokument.

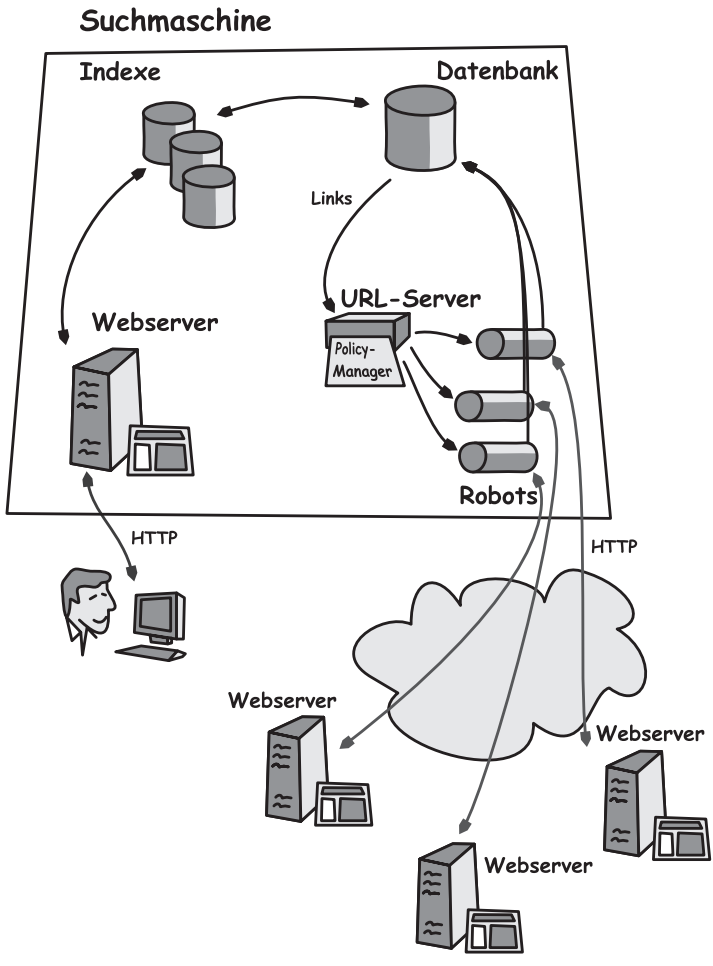


Abbildung 8.1: Architektur einer Suchmaschine

Fallbeispiel einer bekannten Suchmaschine (Google):

Die vom Robot gesammelten Seiten werden von einem Storeserver übernommen. Dieser nimmt die gefundenen Dokumente entgegen und zerlegt sie in ihre Bestandteile. Eine Auswertungsfunktion klassifiziert die Seiten und baut aus diesen Ergebnissen eine Datenbank auf.

Die Auswertung geschieht wie folgt:

Zunächst wird dem gefundenen Dokument eine eindeutige Dokumenten-ID vergeben. Hierfür sind 27 Bit vorgesehen, das heißt, mit dieser Nomenklatur können 2^{27} = 134.217.728 Dokumente verwaltet werden.

In der Datenbank sind nicht die einzelnen Wörter hinterlegt, vielmehr werden diese über eine eindeutige Wort-ID referenziert. Im vorliegenden Beispiel sind dafür 24 Bit vorgesehen, damit können 2^{24} = 16.777.216 Wörter verwaltet werden.

Bei einer Analyse eines Dokuments wird bei jedem Wort (ausgenommen die Stopfwörter) ein Treffereintrag (ein so genannter »Hit«) berechnet. Dabei werden zwei Bewertungskategorien benutzt: Ein Wort kann »fancy« oder »plain« sein: Ein Wort ist fancy, wenn es innerhalb der URL, eines Titels, einer Überschrift, eines Metatags oder eines Links vorkommt. Plain ist ein Wort, wenn es im Text keine herausragende Stellung einnimmt. Kommt beispielsweise das Wort »Fallschirmspringen« im Dokument im Titel oder in Verweisen vor, so deutet das darauf hin, dass sich der Inhalt des Dokuments intensiv mit diesem Begriff auseinander setzt. Kommt es dagegen nur im allgemeinen Textfluss vor, ist die Wahrscheinlichkeit gering, dass sich der Inhalt des Beitrags mit dem jeweiligen Begriff auseinander setzt.

Für die Beschreibung eines Hits sind 2 Bytes vorgesehen. Sie bieten folgende Informationen:

- Bit 1: Das Wort wurde groß- oder kleingeschrieben.
- Bit 2 bis 4: relative Fontgröße des Worts. Insgesamt stehen dafür sechs verschiedene Werte zur Verfügung, die darüber Aufschluss geben, ob sich das Wort aus dem Gesamtkontext über einen größeren oder kleineren Schriftsatz auszeichnet. Der siebte Wert (Bitfolge 111) ist für die »fancy«-Charakteristik reserviert.
- Bit 5 bis 16:
  - plain Hit: Vorkommen des Worts innerhalb des Textes. Insgesamt sind damit 2^{12} = 4096 Positionen des Begriffs darstellbar, was für eine exakte Beschreibung in jedem Dokument mit weniger als 4096 Wörtern ausreicht.
  - fancy Hit: Hier sind vier Bits für den Wortkontext vorgesehen. Damit ergeben sich insgesamt 2^4 = 16 verschiedene Möglichkeiten zu beschreiben, ob das Wort in einer URL, einem Titel, Link oder an anderer exponierter Stelle vorkommt. Diese Form der Charakterisierung kann durch eine hierarchische Strukturierung als Ranking-Mechanismus herangezogen werden.

    Die verbleibenden acht Bit werden wie folgt verwendet:

    – Statischer Text: es sind 2^8 = 256 Positionen für ein Wort innerhalb eines Textes möglich.

    – Link: Hier sind vier Bits für den Hash der Dokumenten-ID vorgesehen. Dieser hilft später beim Auffinden des Dokuments in Zusammenhang mit dem Auftreten des Hits. Die verbleibenden vier Bits dienen zum Positionieren innerhalb des Textes. Damit können 16 Links in einem Dokument verwaltet werden.

Damit enthält ein Hit nicht nur die Information, wo sich ein bestimmter Begriff in einem Dokument befindet, sondern es wird gleichzeitig ein Qualitätsfaktor impliziert, der den Zusammenhang des jeweiligen Begriffs mit dem Kontext des Dokuments aufweist.

Abbildung 8.2 macht noch einmal deutlich, wie ein Begriff als Treffer innerhalb eines Dokuments beschrieben wird.

Ein Dokument wird zunächst auf die vorhandenen Wörter und die entsprechende Hit-Charakteristik analysiert. Zusammen mit der Dokumenten-ID wird eine Liste der vorhandenen Wörter und der entsprechenden Treffer gebildet. Dabei wird zwischen einem »direkten« und einem »reversen« Index unterschieden:

- Direkter Index: Dieser Index listet die Dokumenten-ID zusammen mit einer Auswertung der Charakteristik auf, in der die vorkommenden Wörter und deren Hits aufgelistet sind (Abbildung 8.3).
- Reverser Index: Die Sortierung erfolgt mit Blick auf die Wörter (Word-IDs). Jedes Wort zeigt auf eine Liste von Dokumenten, in denen es ein- oder mehrfach vorkommt. Dieser Index wird später vom User-Interface herangezogen, wenn Anwender nach Verweisen auf bestimmte Begriffe suchen.

Mit Hilfe dieser Hit-Beschreibung wird eine Klassifizierung eines Dokuments umgesetzt, die mit der 27-bit Dokument-ID beginnt. Ein solcher Eintrag hat eine Reihe von verschiedenen Wörtern, jedes mit der 24-bit Wort-ID-Kennung. Weitere acht Bits beschreiben die Anzahl der Treffer für dieses Wort innerhalb des Dokuments, gefolgt von einer Liste der Hits, wie sie im vorangegangenen Abschnitt beschrieben wurden. Eine Datenbank des direkten Index setzt sich aus der Summe aller Dokumente im beschriebenen Format zusammen. Die Architektur der Bitfolge ist in Abbildung 8.3 skizziert.

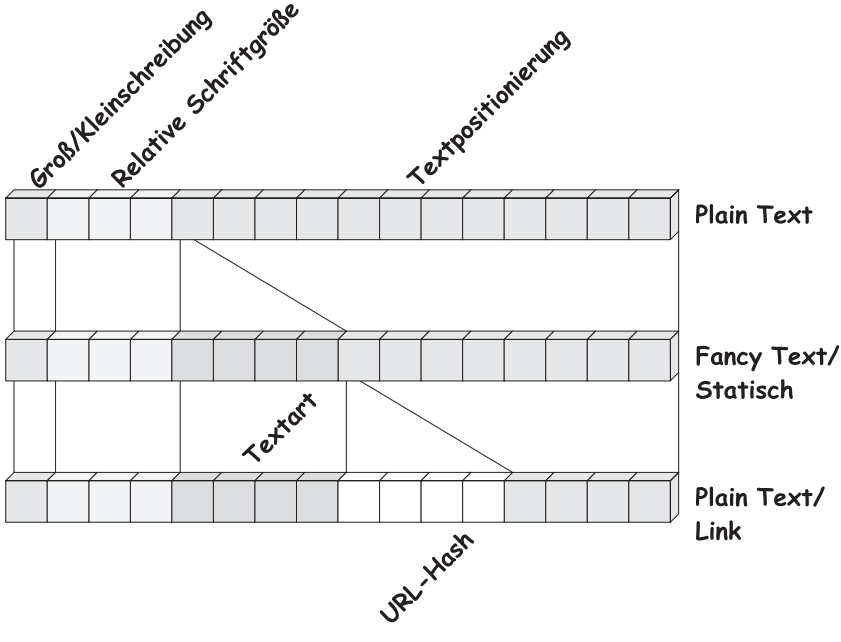


Abbildung 8.2: Beschreibung eines Hits über 2 Bytes (nach Brin/Page [24])

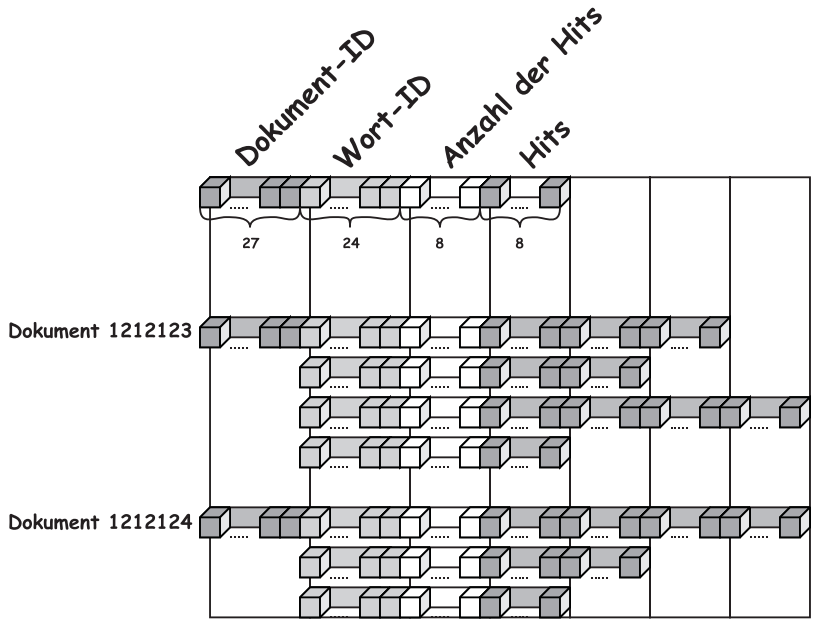


Abbildung 8.3: Wortstatistiken einzelner Dokumente in einem direkten Index (nach Brin/Page aus [24])

Für eine durchschnittliche Webseite ergibt sich etwa folgende Einschätzung:

- Pro Seite sind etwa 2000 Wörter vorhanden, die ein- oder mehrfach vorkommen. Zählt man die Begriffe jeweils einzeln (ohne die Wiederholungen), ergeben sich im Mittel etwa 500 Wörter. Subtrahiert man die Stopfwörter (also diejenigen, die keinen Beitrag zur Charakteristik des Textes liefern), findet man etwa 250 Begriffe, die den Text inhaltlich beschreiben.
- Ein Wort wird mit einem 24-Bit-Code verwaltet. Sieht man für die Dokumentation der Anzahl der Hits eines jeden Worts weitere acht Bit vor und nimmt man an, dass sich im Mittel die Begriffe viermal wiederholen (was über vier Hits dokumentiert wird), so wird pro Wort innerhalb eines Dokuments 96 Bit Information gespeichert.
- Ein Dokument liefert etwa 250 charakteristische Wörter, mithin sind 24.000 Bit bzw. etwa 3 Kbyte für eine Dokumentklassifizierung notwendig. Für eine Klassifizierung von 100 Millionen Webseiten ergibt sich daher ein Datenaufkommen in der Größenordnung von etwa 300 Gbyte.

Der inverse Index ordnet jedem Wort einen Zeiger auf eine Liste von Dokument-IDs zu, in denen der Begriff vorkommt. Ein Ranking-Mechanismus kann diese Zeiger nach einem Bewertungsschema nachsortieren, um bei einer Ausgabe nach einer Begriffssuche eine Priorisierung der Verweise durchzuführen. Damit bekommt der Anwender nach einer Suche eine Liste von Verweisen, in der die ersten Links die höchste Relevanz aufweisen. Je weiter unten ein Link erscheint, umso weniger wahrscheinlich ist es, dass das angezeigte Dokument die Erwartungen des Anwenders erfüllt.

Jeder Dienst hat seine eigene Strategie für die Implementierung der einzelnen Komponenten. Die Anbieter solcher Services halten sich aber mehr oder weniger bedeckt, was die technische Umsetzung angeht. Das besprochene Fallbeispiel orientiert sich an einer großen Suchmaschine, deren Details unter [24] dokumentiert sind.

Request Dealer

Um ein User-Interface bereitzustellen, betreiben die Suchmaschinen einen Webserver mit einem Formular, über das man einen Suchbegriff eintragen kann. Der Request wird von einer Applikation übernommen und ausgewertet. Dieser Request Dealer geht wie folgt vor:

Zunächst wird der Request geparst. Blanks werden als Trennzeichen identifiziert, gegebenenfalls werden weitere Trennzeichen (wie zum Beispiel Kommata) akzeptiert. Eine Fehlerbereinigung der Eingabe kann folgen, bei der Klammern, Sonderzeichen und Ähnliches isoliert und entsprechend den Vorgaben interpretiert werden. Gegebenfalls werden Regular Expressions ausgewertet, andere Maschinen akzeptieren Anführungsstriche zum Definieren von festen Begriffskombinationen (Beispiel: »Tauchen auf Bali«). Die meisten Maschinen können mit booleschen Ausdrücken umgehen. Nachdem der Parser den Request analysiert und bereinigt hat, wird er in der Datenbank gesucht.

Hier gibt es wieder eine Reihe von unterschiedlichen Vorgehensweisen. Im Fallbeispiel werden die Begriffe eines Request in Wort-IDs umgewandelt, anschließend wird im reversen Index die Liste der Treffer zusammengestellt. Bei kombinierten (insbesondere booleschen) Anfragen werden die Trefferlisten kombiniert und in einer gemeinsamen Ergebnisliste zusammengefasst. Der Ranking-Mechanismus definiert die Reihenfolge, in der die Daten ausgegeben werden. Ein festes Limit liefert eine maximale Anzahl von Links aus, die über das User-Interface ausgegeben werden. Der Request Dealer wandelt diese Ausgaben wieder in *html*-Format um und übergibt die Ausgabe an den Webserver.

Webserver

Der Webserver ist die erste Anlaufstelle für einen Internetanwender. Hier findet er die Suchmaske, über die er eine Internetsuche formulieren kann. Für den Betreiber des Dienstes bietet sich hier die Möglichkeit, Werbebanner gewinnbringend zu platzieren. Je erfolgreicher eine Suchmaschine ist, umso mehr User kommen auf der Seite vorbei und umso effizienter ist eine darauf platzierte Werbung.

8.2.3 Ranking

Für die Bewertung eines gefundenen Schlagworts innerhalb einer Seite gibt es verschiedene Ansätze.

Bewertungstechniken

- **Wortspektrum**

  Eines der einfachsten Verfahren einer Bewertung ist das Zählen eines Schlagworts im Dokument. Kommt es mehrmals vor, so ist die Wahrscheinlichkeit hoch, dass das Wort in engem Zusammenhang mit dem Inhalt des Dokuments steht. Erscheint es dagegen nur ein oder zweimal, liegt die Vermutung nahe, dass andere Seiten mit höherer Trefferquote in engerem Zusammenhang mit dem Begriff stehen.

- **Textgestaltung**

  Ein anderes Kriterium ist der Ort, wo ein Schlüsselwort steht: Taucht es im Titel auf, ist zu erwarten, dass sich das Dokument ausführlich mit diesem Begriff befasst. Eine Abstufung der Hierarchie erfolgt anhand der Überschriften-Tags, beginnend bei <H1> in absteigender Folge.

  Weitere Auswertungsverfahren dieser Art sind Kursiv- und Fettformatierungen, die darauf schließen lassen, dass der so hervorgehobene Text von besonderer Relevanz für das Dokument ist. Gerade in Webseiten ist eine solche Auswertung über die Tags <B> oder <STRONG> relativ einfach. Auch ist die Lage des Begriffs im Dokument von Bedeutung: Kommt das Schlüsselwort relativ weit vorne im Text vor, so erhält das Dokument für diesen Begriff ein höheres Ranking, als wenn es sich im unteren Bereich befindet.

- **Metatags**

  Weitere Datenquellen für Suchmaschinen sind insbesondere bei Webseiten die Metatags. Hier sind die beiden Tags *keywords* und *description* von besonderem Interesse.

  keywords: Dieser Tag beschreibt eine Seite, indem das Thema des Dokuments mit Hilfe eines Satzes von Schlagwörtern beschrieben wird. Die Syntax lautet:

  ```
  <meta name="keywords" content="(Begriffe)">
  ```

  Gerade mit diesen Mitteln kann man als Webseitenprogrammierer ein gutes Ranking erreichen, indem die Seite so präzise wie möglich über Schlüsselwörter beschrieben wird. Selbst wenn ein Interessent bei einer ungenauen Suche nicht auf die beschriebenen Webseite gelangt, so ist die Wahrscheinlichkeit doch hoch, dass der gleiche Interessent bei einer sehr viel spezielleren Suche (über entsprechend weniger vorhandene Seiten) das Dokument findet.

  Beispiel: Wir beschreiben auf einer Webseite eine Fahrt mit dem Wohnmobil durch Death Valley. Mit den Schlüsselwörtern »USA« und »RV« ist diese Webseite sicher eine von vielen. Wird diese Seite allerdings mit den Schlüsselwörtern »Death« »Valley« »Wohnmobil« und »Reisebericht« umschrieben, ist die Wahrscheinlichkeit ungleich höher, dass Interessenten, die nach eben diesen Begriffen suchen, unsere Seite finden.

description: Die Syntax lautet:

```
<meta name="description" content="(Beschreibung)">.
```

Dieser Tag dient dazu, bei einem Treffer auf einer Suchmaschine das Dokument zu beschreiben. Damit kann ein Benutzer sehr schnell den Kontext des Links auf die Webseite sehen und entscheiden, ob diese Seite für ihn interessant ist oder nicht. Leider wird dieser Tag von den Webdesignern kaum verwendet, da er nicht unmittelbar auf der Seite zu sehen ist. Erst wenn sich die Seite bei einer Trefferliste mit diesem Text beschreibt, kommt der eigentliche Wert eines solchen Tags ans Licht. Fehlt dieser Tag, wird von einer Seite lediglich das umschließende Textfragment angezeigt, in dem sich der gesuchte Begriff befindet. Wiederum andere Suchmaschinen werten diese Art der Information gar nicht erst aus. So ist dieser Tag eher eine weitere Möglichkeit der Dokumentbeschreibung, die nicht zwingend von Suchmaschinen berücksichtigt wird.

- **Sprache**

  Mit Hilfe des Metatags »Content-Language« wird der Suchmaschine darüber informiert, in welcher Sprache das Dokument verfasst ist:

  <meta name=”content-language”, content=”de”> teilt mit, dass das Dokument in Deutsch vorliegt. Mittlerweile können moderne Suchmaschinen aufgrund der Häufigkeitsverteilung der Buchstaben oder der Wahl der verwendeten Wörter auf die Sprache schließen, ohne auf dieses Metatag des Dokuments zurückgreifen zu müssen.

- **Aktualität**

  Ein Dokument, das einmal von einer Suchmaschine bewertet wurde, kann über die Aktualität nachträglich neu überprüft werden. Seiten sinken im Laufe der Zeit im Ranking, wenn sich die Größe oder das Datum der Modifizierung nicht ändert. Damit soll verhindert werden, dass in der Ergebnisliste der Suchmaschinen alte Daten an oberster Stelle stehen.

- **Verlinkung**

  Hier ist nun nicht die Zahl der Links gemeint, die von dieser Seite wegführen, sondern derjenigen, die auf diese Seite zeigen: Sind viele Seiten im Internet vorhanden, auf denen ein Link auf unsere Webseite eingetragen ist, so wird diese hoch bewertet, da sie häufig zitiert wird. Dabei zählt eine Suchmaschine in der Datenbank die Links auf die Seite.

  Ausgefeiltere Suchmaschinen bewerten die Qualität der Links nach den Seiten, von denen sie ausgehen. So ist ein direkter Link von der Yahoo-Startseite wesentlich höher bewertet, als der Verweis von einer Homepage, die nur sporadisch aufgerufen wird.

  Die mathematische Bewertung eines Dokuments basierend auf einer Verlinkung wird in [25] beschrieben. Hier wird berechnet, wie ein einzelner Link von einer »wichtigen« Seite eine Anzahl von Verweisen von weniger

wichtigen Seiten wettmacht. Das mathematische Fundament wird auf ein Eigenwertproblem einer rekursiven Gleichung heruntergebrochen, das von der Suchmaschine für jede Seite aufs neue berechnet wird. Diese Form der Gewichtung kommt im Suchserver »Google« zum Einsatz.

Die verschiedenen Bewertungskriterien werden meist kombiniert verwendet, gute Suchdienste verwenden alle Maßstäbe, um ein Webdokument zuzuordnen und der gefundenen Seite einen Qualitätsfaktor zu verleihen.

Die Bewertung der Webseiten ist ein wichtiger Faktor im Service einer Suchmaschine. Ein gutes Ranking steigert die Zufriedenheit eines Anwenders mit dem verwendeten Dienst, da er direkt unter den ersten Treffern die gesuchte Information findet.

Missbrauch des Rankings

Während es für die Suchmaschine notwendig ist, eine Webseite beurteilen zu können, ist es für einen Webseiten-Programmierer wichtig, bei einer Suche über den Suchdienst möglichst weit oben auf der Liste der gefundenen Dokumente zu erscheinen. Ein Anwender, der bei der Suche etwa fünfstellige Ergebnismengen bekommt, wird sich maximal an den ersten zwanzig Treffern orientieren.

Um möglichst weit oben in der Liste zu erscheinen und damit die Auffindbarkeit der eigenen Seite zu erhöhen, versuchen manche Programmierer, die Suchmaschinen zu einer unverhältnismäßig hohen Priorisierung zu verleiten.

Beispiel:

Der Betreiber einer Webseite für den Verkauf von Briefmarken erzeugt eine Seite, auf der das Wort »Briefmarke« mehrere hundert Mal vorkommt. (Was nicht heißt, dass die Seite nur aus diesem Begriff besteht: Der Text kann beispielsweise besonders klein gehalten und zusätzlich dadurch versteckt werden, dass man die Farbe des Fonts mit der des Hintergrunds gleichsetzt.) Andere Webprogrammierer setzen in die beschriebenen Metatags eine endlose Liste vieler beliebter Begriffe. Selbst die Fremd-Verlinkung auf eine bestimmte Seite kann manipuliert werden, indem sich größere Sammlungen von artverwandten Seiten gegenseitig verlinken und somit wechselseitig eine hohe Zahl von Hits erreichen. Hier muss ein Robot vorhandene Korrelationen erkennen und diesen Abhängigkeiten ein entsprechend niedriges Ranking bescheren.

Berücksichtigt man die beschriebenen Bewertungskriterien im vorangegangenen Kapitel bei der Erstellung einer Webseite, so kann man die Auffindbarkeit im Netz stark erhöhen. Als weiteres Hilfsmittel sind mittlerweile Dienste im Internet eingerichtet, mit denen man die eigene Webseite auf die Positionierung in einer Suchmaschine testen und optimieren kann.

Der Wettstreit zwischen unberechtigtem Erschleichen von hohen Ranking-Werten und der Bereitstellung zuverlässiger Recherchedaten führt zu einer rasanten Weiterentwicklung von Internetkatalogen.

Für die Wahrnehmung einer Webseite im Internet spielt das Erscheinen eines Links auf der Trefferliste einer Suchmaschine eine entscheidende Rolle.

Webadministratoren sind auf der einen Seite bestrebt, ihre Seiten in einer Trefferliste auf der Suchmaschine möglichst weit oben zu positionieren. Suchmaschinen stellen auf der anderen Seite Bewertungsmechanismen auf, um ein zuverlässiges Qualitätsmerkmal der gefundenen Seite zu bekommen.

8.3 Webserverseitige Robot-Kontrolle

Als Webserver-Administrator benötigt man eine gewisse Kontrolle über die Aktivitäten eines Robots einer Suchmaschine. Unter Umständen ist es nicht erwünscht, dass bestimmte Dokumente von der Suchmaschine aufgenommen werden oder dass der Robot eine zu große Last erzeugt, was den Betrieb stören kann.

Über die Datei »robots.txt« auf dem Webserver kann man den Suchmaschinen mitteilen, welche Daten für die Suchmaschine relevant sein sollen und welche nicht. Diese Informationen werden auf der Toplevel-Seite in einer Datei mit dem Namen hinterlegt.

Beispiel:

Der Webserver-Administrator legt die Datei »robots.txt« an, die unter der URL »http://www.meinefirma.com/robots.txt« erreichbar ist. In dieser Datei werden die Robots angewiesen, die Seite nach den Vorgaben des Webmasters zu indexen.

Für einen Robot einer Suchmaschine stellt dieses File lediglich eine Empfehlung dar, er ist nicht gezwungen, diese einzuhalten. Es gehört aber zum guten Ton, wenn die Wünsche eines Webadministrators bei der Indexierung einer Webseite vom Betreiber einer Suchmaschine berücksichtigt werden, insbesondere, wenn sie sich in der standardisierten Form des Robots.txt-Files äußern. Letztendlich sind aber alle Seiten, die von einem beliebigen Webclient aus erreichbar sind, für einen Robot einer Suchmaschine ebenfalls offen.

Ein Robots-File besteht aus der Auflistung zweier Attribute: *User-agent* und *Disallow*. Beide sind in Abhängigkeit zu sehen: Ein User-agent-Attribut definiert den Zugriff durch eine Suchmaschine, das darauffolgende Disallow-Attribut die Zugriffsrechte. Dabei kann das Disallow-Attribut ein oder mehrfach in Folge aufgelistet werden.

Hier ein Beispiel, wie ein solches File aussehen kann:

```
# /robots.txt file for http://www.meinefirma.com/
# mailaddress: webmaster@meinefirma.com
User-agent: lycos
Disallow:
User-agent: google
Disallow: /
User-agent: *
Disallow: /finance
Disallow: /marketing
```

Im vorliegenden Beispiel liest der Robot beim Kontakt mit der Seite die Textdatei und stellt fest, was er darf und was nicht: hier insbesondere, dass – falls der Robot von lycos kommt – die Seite fast ohne Restriktionen indiziert werden kann, für den Fall eines Robots von Google, dass keine Seite gelesen und ausgewertet werden soll. Für alle Robots gilt, dass sie die Dateien aus dem URL-Bereich »finance« und »marketing« nicht berücksichtigen sollen.

Neben dem Robots-File kann man einer Webseite auch explizit mitteilen, wie sie von einer Suchmaschine behandelt werden soll.

Hier ein Überblick über die Möglichkeiten:

- <meta name="robots", content="index"> teilt mit, dass die Webseite indexiert werden soll. Dies ist eigentlich der Normalfall, der auch ohne dieses Tag vom Suchmaschinen-Robot durchgeführt wird.
- <meta name="robots", content="noindex"> teilt dagegen mit, dass die Webseite ausdrücklich nicht indexiert werden soll.
- <meta name="robots", content="follow"> teilt dem Robot mit, die Links auf der vorhandenen Seite für die nachfolgende Berücksichtigung im Suchkatalog zu verfolgen. Dies ist der Normalfall und geschieht auch ohne das Vorhandensein dieses Tags.
- <meta name="robots", content="nofollow"> berücksichtigt die Links auf der vorhandenen Seite nicht und weist den Robot an, nach Indexierung die Seite ohne Berücksichtigung der darin vorhandenen Links zu verlassen.

Der Quasistandard Robots.txt ist unter [23] beschrieben.

Für die Steuerung von Robots der verschiedenen Suchmaschinen kann unterhalb des Indexseite ein File mit dem Namen Robots.txt hinterlegt werden, das die Auflagen des Webmasters für die jeweilige Indexierung der Seite beschreibt. Inwieweit sich ein Robot an diese Anforderungen hält, liegt im Ermessen des Betreibers der Suchmaschine.

8.4 Zusammenfassung

Für die Erschließung des immensen Datenbestands des Internet sind Suchmaschinen und Kataloge unerlässlich. Ohne solche Dienste könnte man sich nur durch Raten von URLs oder über die eigenen Bookmark-Sammlungen weiterhelfen, wobei die wirklich wichtigen Ressourcen und Informationen meist unentdeckt blieben.

Während sich die ersten Suchmaschinen vorwiegend aus der akademischen Forschung entwickelten, führten im Laufe der Zeit die kommerziellen Aspekte einer populären Webseite zur stetigen Verbesserung der Recherchierwerkzeuge. Einmal mehr ist die Werbung der Garant für die mehr oder weniger kostenlose Dienstleistung. Aufgrund der Konkurrenz der verschiedenen Suchmaschinen untereinander wurde stetig an der Qualität dieser Tools gefeilt. Im Laufe der Zeit haben sich einige Anbieter als »Marktführer« unter den Suchmaschinen etabliert, die mit einer umfangreichen Funktionalität und Qualität aufwarten.

Es hat sich gezeigt, dass die Einrichtung eines konkurrenzfähigen Suchdienstes eine ausgefeilte Architektur erfordert, die die Speicherung der Webdaten effizient gestaltet und die Aufbereitung der Daten zu Verweisen auf Internetquellen performant liefert. Auf der anderen Seite bietet eine solche Technik kaum Ansatzpunkte für einen Software-Hersteller, eine »Komplett-Paket-Suchmaschine« zu entwickeln. Vielmehr sind die Lösungen der Suchmaschinen Einzelentwicklungen. Es gibt durchaus Software-Pakete, die den Aufbau einer Suchmaschine gestatten, diese werden aber vornehmlich für einen unternehmensinternen Suchdienst verwendet, um Seiten, die nur in einem abgeschlossenen Intranet angeboten werden, für den Mitarbeiter anzubieten. Für einen offenen Internetdienst sind solche Lösungen kaum brauchbar.

9 Kalender-Server

9.1 Zeitplanung

Im modernen Geschäftsumfeld nimmt die persönliche Terminverwaltung einen immer höheren Stellenwert ein. Der traditionelle Weg über Timer in Papierform war lange Zeit die einzige zuverlässige Methode, Meetings und Aufgaben festzuhalten, und noch heute ist dieses Mittel für viele Menschen die einzige Möglichkeit, eine Übersicht über Einsatzorte und -zeiten zu verwalten.

Die Ausprägungen von Kalendern sind sehr vielfältig. Ein einfacher Taschenkalender passt noch in die Westentasche, bietet aber kaum Möglichkeiten, umfangreichere Beschreibungen beizufügen oder Adressen oder Telefonnummern aufzunehmen. Größere Kalender haben meist Ringbuchformat und können erweitert werden. Damit hat man zwar die Möglichkeit, mehr an Termininformationen aufzunehmen und Telefonnummern wie Adressen zu speichern, größere Datenmengen sind aber unhandlich zu verwalten, nicht mit anderen Kalendern zu synchronisieren und nur umständlich zu korrigieren.

Digitale Kalender, die auf einer speziell dafür optimierten Hardware laufen, haben etwa das Format einer Westentasche, bieten aber eine ungleich bessere Möglichkeit, große Datenmengen bereitzuhalten, zu synchronisieren und zu verwalten. Diese Geräte bieten auch weitere Funktionalitäten an als ein ringbuchbasierter Timer. Daten können mit anderen Geräten oft relativ einfach über Infrarotschnittstellen ausgetauscht werden, und zwar nicht nur unter gleichartigen Geräten, sondern auch hersteller- und geräteübergreifend. Diese Vielfalt von Funktionen bieten so genannte Persönliche Digitale Assistenten (Personal Digital Assistants, PDAs), die auch Notizen, Merkzettel, Spiele und elektronische Bücher aufnehmen.

So wie PDAs untereinander oder zwischen mobilen Telefonen Daten austauschen, haben sie auch eine Schnittstelle zum PC. Über diesen kann eine komfortablere Datenverwaltung erfolgen, insbesondere ist es auch möglich, Backup/Recovery-Aktionen durchzuführen. Gehen die Daten auf einem PDA verloren, können sie über das Einspielen einer Sicherungskopie wieder hergestellt werden.

9.1.1 Zugriffsverwaltung

Bisher wurde nur die Handhabung der Daten eines Organizers beschrieben. Den Zugriff darauf gestattet der Eigentümer üblicherweise nur sich und eventuell Vertrauenspersonen, da diese Informationen zur eigenen Privat-

sphäre gehören. Geht man von diesem Standpunkt etwas ab und lässt anderen die eigene Terminplanung zukommen, können diese ihren eigenen Zeitplan entsprechend abstimmen. Hier gilt es, einen Kompromiss zwischen Privatsphäre und notwendigem öffentlichen Zeitmanagement zu finden.

In Bezug auf Termindaten gibt es zwei Arten von Informationen: Zeitraum und Zweck.

- Zeitraum: Termine beginnen und enden zu bestimmten Zeitpunkten. Innerhalb dieser Zeitabschnitte ist eine Person nicht für andere Termine verfügbar. Gestattet ein Mitarbeiter den Einblick auf die Verfügbarkeit, können andere ihren eigenen Terminplan dementsprechend abgleichen.
- Zweck: Ein Termin hat eine Intention, sei es ein Meeting, eine Freizeitaktivität oder eine Reservierung.

Der Zugriff auf die Termindaten kann auf den Zeitraum reduziert sein oder auch den Zweck offenbaren. Welche der Daten veröffentlicht werden, kann von Termin zu Termin individuell bestimmt werden. Eine Klassifizierung kann in »Öffentlich« (Zeitraum und Zweck können von jedem eingesehen werden), »Privat« (der Zeitraum, aber nicht der Zweck wird veröffentlicht) und »Geheim« (keine der Daten werden angezeigt) erfolgen. Aus Sicht der Datenhaltung eines Timers geht man üblicherweise von einer geheimen Sicht aus. Öffnet man die Daten für andere, können die beiden anderen Zugriffsarten angeboten werden. Dies kann aber nur über einen digitalen Weg erfolgen, da ein papierbasierender Terminkalender kaum Möglichkeiten hierfür bietet.

Neben dem lesenden lässt sich auch der schreibende Zugriff auf einen Kalender für andere einrichten. Zwei Möglichkeiten bieten sich hierfür:

- Im Kalender oder über andere Wege werden Terminvorschläge unterbreitet, die der Eigentümer des Kalenders annimmt oder ablehnt.
- Termine können direkt von anderen Personen in den Kalender eingetragen werden.

Die Entscheidung zwischen den beiden Methoden hat viel mit dem Vertrauen zu tun, das man den anderen Personen entgegenbringt, die schreibenden Zugriff auf den eigenen Kalender haben.

9.2 Verteilte Kalender

Der Begriff »Verteilter Kalender« bezeichnet einen Terminplaner, der von mehr als einer Person gleichzeitig gelesen und beschrieben werden kann. Dieser Zeitplan ist im Besitz einer Person, die einer oder mehreren anderen Personen den Zugriff auf die eigenen Termindaten gestattet.

Die digitale Version eines verteilten Kalenders bietet eine Reihe von Vorteilen:

- Verfügbarkeit: Ein digitaler Kalender kann auf speziellen Devices oder auf dem heimischen PC installiert sein. In der Regel gibt es Synchronisationstools, die den Abgleich dieser Datenbanken vornehmen. Damit stehen die Termine einer einzelnen Person über mehrere Zugriffsmöglichkeiten zur Verfügung.
- Interoperabilität: Ein Kalender wird nicht mehr von einer einzigen Person gepflegt, vielmehr haben mehrere Personen oder Instanzen die Berechtigung, Kalenderdaten einzupflegen. Aus Datensicht existiert ein übergeordneter konsolidierter Kalender. Die (realen) Kalender, die die einzelnen Personen bei sich tragen, haben einen Datenbestand von einer Zeit *t*, der um Einträge von Personen erweitert wird, die in diesem Kalender die Schreibberechtigung haben. Der konsolidierte Kalender ergibt sich aus der Synchronisation aller beteiligten realen Kalender nach Auflösung aller eventuell vorhandenen Terminkonflikte.

Dieser Begriff eines verteilten Kalenders erfordert somit eine Umstellung des gewöhnlichen Zeitplanerbegriffs. Man gibt einen Teil der Informationshoheit an einen Dienst oder eine Instanz ab, erhält dafür aber ein effizienteres Zeitmanagement.

9.2.1 Synchronisation

Bisher haben wir einen Kalender als abstrakten Zeitplan betrachtet und angenommen, dass dieser immer in unmittelbarer Nähe des Eigentümers ist. Dies ist für papierbasierende Timer sicherlich richtig. Die Kalenderdaten werden von einer Person selbst gepflegt. Bestenfalls einem Assistenten oder einer sonstigen Vertrauensperson wird es gestattet, Termine einzutragen. Dies ist bei einem PDA nicht anders, mit dem Unterschied, dass man über Kopien des Datenbestands auf dem PC gleichfalls Daten pflegen kann. Diese lassen sich über die Synchronisation mit dem PDA abgleichen.

Geht man davon aus, dass der Kalender auf dem eigenen Timer oder PDA der einzig gültige Kalender ist, so steht man vor dem Problem, dass dieser nicht immer für alle verfügbar ist. Ist beispielsweise der Kalender einer Managerin für ihre Sekretärin nicht erreichbar, kann diese keine Termine eintragen, da sie nicht weiß, ob es zu Überschneidungen bei einem in Frage kommenden Zeitraum kommt. Tut sie dies dennoch und treten Konflikte in Form von Terminüberschneidungen auf, müssen diese aufgelöst werden.

Für die Synchronisation der Daten existieren verschiedene Datenflüsse: Termine werden von einem Kalender zum anderen (oder umgekehrt) synchronisiert oder Änderungen werden auf beiden Seiten gestattet. Im zweiten Fall können Terminkonflikte auftreten und der Synchronisationsmechanismus muss intelligent genug sein, die beiden Seiten miteinander abzugleichen.

Beispiel: Eine Abteilungsleiterin pflegt ihre Termine über ihren PDA, ihre Sekretärin über den PC. Der verwendete Synchronisationsmechanismus

gleicht beide Kalender miteinander ab, so dass auf beiden Seiten ein konsolidierter Kalender vorliegt. Für den Fall, dass für einen bestimmten Zeitraum auf dem PDA ein Termin gesetzt wird, der auf dem PC bereits von einem anderen Termin belegt ist, tritt ein Konflikt auf. Diesen muss man über eine Priorisierung auflösen, indem die Termine nachträglich geändert werden.

Ein Kalender hat also einen Eigentümer. Dieser erteilt anderen Personen unterschiedliche Rechte auf den Termindatenbestand. Die Daten werden in Form von Sichten auf den Kalender verteilt, die über den Eigentümer hinaus von anderen Personen beschrieben werden können. Der konsolidierte Terminplan ergibt sich somit aus der Zusammenfassung aller Kalender nach Auflösung der Terminkonflikte. Da eine Synchronisation mit allen Sichten nicht permanent erfolgt, ist der konsolidierte Terminplan eher die Ausnahme als die Regel.

Mit jeder neuen Person oder Instanz, die über eine eigene Version eines Kalenders verfügt, die im virtuellen Kalender abgeglichen wird, steigt die Wahrscheinlichkeit von Konflikten. Umgekehrt, je zeitnaher das Abgleichen der einzelnen realen Kalender mit dem übergeordneten konsolidierten Kalender erfolgt, umso weniger treten Überschneidungen von Terminen auf, wodurch die Konflikte zeitnah aufgelöst werden können.

Für die Auflösung der Terminkonflikte gilt die Frage der Informationshoheit: Wer hat das Recht der Priorisierung bei Konflikten oder Terminvorschlägen? Dies kann, muss aber nicht der Inhaber des Kalenders sein. Natürlich sollte das Selbstbestimmungsrecht dem einzelnen die Wahl lassen, selbst zu entscheiden, welche Termine er an welchem Ort wahrnimmt. Im Alltag gibt man aber allzu oft dieses Recht scheinbar ab und lässt sich sagen, wo man welchen Termin wahrzunehmen hat.

9.2.2 Gruppen- und Ressourcenkalender

Bisher wurden nur die Termindaten einer einzelnen Person betrachtet. Darüber hinaus können auch Sachen oder Lokalitäten Termine haben, die deren zeitliche Verfügbarkeit beschreiben: Beamer, Konferenzräume, Leihwagen. Hier ergibt sich eine neue Definition der Informationshoheit: Wer entscheidet über die Verfügbarkeit dieser Kalender, wenn es zu Konflikten kommt? Üblicherweise sollten Ressourcen dieser Art selbst verwaltend sein. Nur im Fall von Konflikten wird von einer Person oder Institution, die die Verantwortung für diese Ressource übernommen hat, bei Terminüberschneidungen die Priorisierung vorgenommen.

Andere Kalender, die nicht modifiziert werden können, aber einen wichtigen Beitrag in der Freizeitgestaltung leisten, sind Bundesligakalender oder Kinoprogramme. Werden diese Termine angeboten und in anderen, persönlichen Kalendern integriert, so kann der einzelne über die Priorisierung dieser Daten über die eigenen Termine entscheiden.

Unter diesen Gesichtspunkten ist eine digitale Terminverwaltung, die auf verteilten Kalendern beruht, eine sinnvolle Ergänzung zur persönlichen Datenverwaltung.

9.3 Kalender-Server

Führt man den Gedanken eines verteilten Kalenders konsequent weiter, so gelangt man zu einer serverbasierten Lösung, in der man seine eigene Sicht auf die Termine in regelmäßigen Abständen abruft. Hier werden Kalenderdaten gesammelt und organisiert, Konflikte aufgelöst und Zugriffe durch andere Personen verwaltet.

Kalender-Server kontrollieren die Zugriffe auf die Termindaten, sowohl lesend als auch schreibend. Sie sind selbst organisierend und haben Schnittstellen zu den verschiedensten Endgeräten. Der Einsatz einer solchen Architektur erfordert allerdings ein Umdenken in der herkömmlichen Verwendung von Termindaten. Der persönliche Terminplaner, den man immer bei sich trägt, verliert seinen Stellenwert, die Informationshoheit wandert auf den Kalender-Server über, in dessen Datenbank alle relevanten Termine eingetragen sind. Kopien, die über Synchronisationen auf andere Devices übertragen werden, sind Sichten auf diese Datenbank, die zu einem bestimmten Zeitpunkt erzeugt werden und die Änderungen aufnehmen, die bei der nächsten Synchronisation auf dem Server eingepflegt und gegen Konflikte aufgelöst werden.

Das heißt nicht, dass die Person, deren Kalender nun von zentraler Seite aus verwaltet wird, nicht mehr bestimmen kann, welche Termine sie annimmt. Vielmehr hat sie sehr hohe, wenn nicht die höchsten Rechte bei der Bestimmung, welche Termine akzeptiert werden und welche nicht. Als Eigentümer eines Kalenders gibt man damit nicht seine Zeitplanung auf. Vielmehr gestattet man anderen einen Einblick in die eigene Verfügbarkeit und erlaubt in bestimmtem Umfang anderen Personen einen schreibenden Zugriff auf die eigenen Daten.

Der Server übernimmt die Organisation von Kalenderdaten, verteilt Zeiträume, akzeptiert oder lehnt Terminvorschläge ab und hält jederzeit einen konsistenten Datenstamm vor.

Eine solche Architektur bietet zudem Schnittstellen zu diversen Endgeräten, sei es eine webbasierte Lösung, in der die Termindaten mit Hilfe eines Browser über das Internet abgefragt werden können, sei es ein Synchronisationsmechanismus, der die Datenbank mit einem PDA abgleicht, oder auch ein Drucker, der die Termindaten formatgerecht für den Timer ausdruckt.

Kalender-Server kommen im Intranet von Unternehmen zum Einsatz. Über diese werden die Termine der Mitarbeiter verwaltet und alle Kollegen haben die Möglichkeit herauszufinden, ob der einzelne Mitarbeiter für Meetings verfügbar ist. Man kann dabei seinen eigenen Kalender so konfigurieren, dass Zeiträume angezeigt werden, in denen man nicht verfügbar ist, ohne

die Einzelheiten der Termine preiszugeben. Andererseits lassen sich auch Detailinformationen freigeben, wenn diese für die Zusammenarbeit mit den anderen Mitarbeitern notwendig sind. Terminvorschläge können direkt in den Kalender eingetragen oder/und per Mail verschickt werden. Durch eine Bestätigung eines Termins werden in den Kalenderdaten aller teilnehmenden Personen die Zeiten festgelegt.

Über eine solche intranetbasierte Kalenderlösung werden bereits in zahlreichen Unternehmen die Einsätze der Mitarbeiter geplant. Gruppen-Kalender entstehen aus der Zusammenführung der Mitarbeiterkalender, damit jeder über den Zeitplan von wichtigen Events informiert ist. So kann sich eine Volleyballgruppe in einem Unternehmen bilden, die ihre Trainingszeiten über einen Volleyball-Kalender publiziert. Dieser Kalender wird mit denen der einzelnen Mitarbeiter zusammengelegt, die sich als Spieler eingetragen haben. Alle gruppenrelevanten Termine wie Training oder Festivitäten werden automatisch in den individuellen Kalendern angezeigt.

Kalender-Server kommen aber auch immer häufiger als freie Services zum Einsatz. So wie Webmail werden Terminverwaltungsdienste als freier Service über Portale angeboten, deren Betreiber sich dadurch eine erhöhte Kundenbindung erhoffen. Die Erfahrung hat allerdings gezeigt, dass Anwender ihre privaten Termine ungern einem kostenlosen Service anvertrauen.

Unter diesen Aspekten spielen Kalender-Server im Internet sicher noch eine untergeordnete Rolle, sie bieten aber einen interessanten Ansatz zur besseren Eigenorganisation.

9.4 Architektur einer Kalenderumgebung

Beim Aufbau einer Kalenderinfrastruktur bedarf es einer Architektur, die

- auf User-Daten zugreifen kann. User-Daten werden entweder lokal vorgehalten oder über *ldap* aus einem Directory Server integriert.
- eine Datenverwaltung beinhaltet: Die Datenverwaltung wird entweder über eine externe oder eine integrierte Datenbank realisiert.
- Schnittstellen zu den diversen Clients und Endgeräten hat: Diese bieten die Möglichkeit, Kalenderdaten mit anderen Clients und Endgeräten auszutauschen.
- interoperabel mit anderen Kalendern ist: Für den Fall, dass systemübergreifend Termindaten ausgetauscht werden müssen, beispielsweise zwischen zwei Unternehmen mit unterschiedlichen Kalender-Server-Produkten, sollten Standards den Datenaustausch unterstützen.
- skaliert. Eine Kalenderlösung sollte mit dem Datenaufkommen wachsen können.

Die nächsten Abschnitte befassen sich mit dem Aufbau einer netzverteilten Kalenderlösung. Die Hersteller solcher Lösungen bieten teilweise recht unterschiedliche Architekturen, wobei sich aber in den Grundzügen eine Tendenz in Richtung Standardisierung abzeichnet.

9.4.1 Standards und Protokolle

Viele Lösungen von Kalendertopologien haben ihre eigenen Datenformate und Übertragungsprotokolle und es ist recht schwierig, herstellerübergreifende Kommunikation umzusetzen. Gerade dies ist aber wichtig, wenn die diversen Endgeräte unterstützt werden sollen, die zwar auf diese Daten angewiesen sind, aber nicht mehr dem Einfluss des Kalenderherstellers unterliegen. So muss beispielsweise ein PDA mit einem Kalender-Server synchronisiert werden, was eine übereinstimmende und eindeutige Datenspezifikation auf beiden Seiten erfordert.

Den Bedarf einer internetweiten Standardisierung von Protokollen und Formaten für den Datenaustausch von Kalenderevents erkannte auch die Internet Engineering Task Force. Sie gründete einen Arbeitskreis »Calendaring and Scheduling« (calsch), der es sich zum Ziel gesetzt hat, ein Regelwerk von Standards festzulegen, an denen sich Hersteller solcher Software-Lösungen orientieren können.

iCalendar

Ein Kalenderformat, das weite Verbreitung gefunden hat, ist iCalendar.

iCalendar wird unter RFC 2445 [36] beschrieben. Hierbei handelt es sich um ein MIME-fähiges Format, das Termine in Form von Datumsangaben und -beschreibungen aufnehmen kann. Der MIME-Typ wird mit text/calendar beschrieben, die Daten liegen in einem strukturierten ASCII-Text vor.

Ein iCalendar-Datensatz bietet aber eine viel umfangreichere Funktionalität als nur die Dokumentierung eines Termins. Vielmehr handelt es sich hier um ein Kalenderobjekt mit vordefinierten Methoden, die von Terminabsagen bis zu Benachrichtigungen über Terminvorschläge reichen. Die Datenübertragung kann über bestehende Übertragungsprotokolle umgesetzt werden, insbesondere bieten sich *http* – für die Benachrichtigung über das Web – und *smtp* – über das Mailsystem – an.

Betrachten wir einmal ein iCalendar-Objekt. Die formale Struktur legt einen Rahmen fest, in dem ein Event spezifiziert wird:

```
BEGIN:VCALENDAR
calendarProperty 1
calendarProperty 2
...
component 1
component 2
...
END:VCALENDAR
```

Während die »calendarProperty *x*« kalenderspezifische Informationen beinhalten, legen die einzelnen Komponenten »component *y*« Termine und Aufgaben fest.

Beispiel für ein iCalendar-Objekt:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//HerstellerXY//NONSGML My Product//EN
CALSCALE:GREGORIAN
...
component 1
component 2
...
END:VCALENDAR
```

»VERSION« übermittelt die verwendete Version des Formats, »PRODID« steht für »ProduktID«. Dieses Attribut informiert über das Programm, das diese Kalenderdaten generiert hat, und »CALSCALE« legt den verwendeten Kalendertyp fest, in unserem Fall der Gregorianische Kalender.

Die nächsten Datensätze sind die »components«, also Events, Termine oder Aufgaben. Eine Komponente hat beispielsweise folgende Form:

```
BEGIN:VEVENT
DTSTART:20010921T190000+01:00
DTEND:20010921T223000+01:00
SUMMARY:Kinoabend mit Freunden
UID:012345678ABCDEFG@meinefirma.com
END:VEVENT
```

Events werden in der BEGIN/END-Klammerung definiert. Darin enthalten sind alle notwendigen Informationen für die Spezifikation eines Termins.

- DTSTART: Dieses Attribut legt den Beginn eines Termins fest. Die Syntax beruht auf dem ISO8601-Format: Jahr, Monat, Tag, »T«, Stunden, Minuten, Sekunden und Zeitzonen-Indikator. Fehlt Letzterer, so wird die lokale Zeit angenommen, ist er spezifiziert, liegt die koordinierte universelle Zeit (Coordinated Universal Time, UTC) vor. Diese ist für Deutschland eine Stunde im Vorlauf, also +01:00h.
- DTEND: Mit diesem Attribut wird das Ende eines Termins festgelegt. Die Zeitangabe entspricht der des Attributs DTSTART.
- SUMMARY: eine Beschreibung des Events.
- UID: Dem Event wird eine weltweit eindeutige EventID zugeordnet.

iCalendar lässt sowohl für die Kalenderspezifikation als auch für die Termindefinition noch eine Vielzahl von weiteren Attributen zu. Die vollständige Beschreibung ist in [36] dokumentiert.

Der vollständige iCalendar-Eintrag aus unserem Beispiel lautet somit:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//HerstellerXY//NONSGML My Product//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
DTSTART:20010921T190000+01:00
DTEND:20010921T223000+01:00
SUMMARY:Kinoabend mit Freunden
UID:012345678ABCDEFG@meinefirma.com
END:VEVENT
END:VCALENDAR
```

9.4.2 Architektur

Es ist schwierig, eine allgemeingültige Konzeption einer Kalender-Server-Umgebung zu beschreiben, da jeder Hersteller einer solchen Lösung seine eigene Philosophie verfolgt. Man kann aber von bestimmten Grundprinzipien ausgehen, die den meisten Implementierungen gemein sind.

Komponenten einer Kalenderinfrastruktur

Um eine Kalenderarchitektur aufzubauen, müssen verschiedene interne und externe Komponenten zusammengestellt werden, die aufeinander aufbauen.

Die externen Komponenten umfassen das User-Repository in Form eines Directory, einen Mailserver, der Notifications und Einladungen versendet, einen Webserver, der ein *html*-Frontend bereitstellt, sowie Schnittstellen zu Applikationen, um weitere Clients über Synchronisationstools mit dem Kalender-Server zu verbinden. Während diese Backend-Services die User-Daten und Grundfunktionalitäten bereitstellen, die für den Betrieb des zentralen Kalender-Servers notwendig sind, werden Termine von den Anwendern über die Calendar-User-Agents (CUA) übermittelt, die ebenfalls externe Komponenten darstellen. Zu CUAs gehören:

- Endgeräte wie PDAs, die über eine proprietäre Hard- und Software verfügen und sich über Synchronisationsmechanismen mit den Datensätzen des Kalender-Servers abgleichen (Termine und Aufgaben werden als Kopien der Datensätze auf diese Geräte übertragen und stehen anschließend offline zur Verfügung),
- Client-Programme, die proprietäre oder standardisierte Kalenderprotokolle verwenden, um Sichten auf die Datenbank umzusetzen (die durch diese Programme erzeugten Sichten stellen die Termine dar, seltener werden lokale Kopien der Daten erzeugt),

- Web-Browser, deren *html*-Sicht nur eine grafische Oberfläche liefert und bei denen *http* als universelles Übertragungsprotokoll verwendet wird (auf Seiten des Kalender-Servers generiert eine Webapplikation den *html*-Output zum Client; die Inhalte werden aus dem Kalender-Server erfragt).

Die Zugriffsmöglichkeiten dieser User Agents sind vielfältig. Je nach Kalender-Server-Implementierung können diese Agents über *http*, *https*, *cap* (Calendar Access Protocol) oder *wcap* (Web Calendar Access Protocol) mit dem Server kommunizieren.

Im Directory Server sind alle Benutzerdaten hinterlegt, die für den Betrieb erforderlich sind. Hierzu gehören Daten wie User-ID und Passwort für den authentifizierten Zugriff auf die persönlichen Daten, User-Profil-Informationen (Kalenderspezifikationen wie Sprache und Kalenderlayout, Land und Zeitzonen, Uhrzeit- und Datumsformat).

Der Kalender-Server muss ferner über eine Möglichkeit verfügen, über *smtp* Mails zu versenden. Dieser Weg wird verwendet, um Benachrichtigungen an den User zu übertragen, Einladungen auszuliefern und Terminbestätigungen zu empfangen.

Eine weitere externe Komponente kann ein SMS-Notification-Server sein, der vom Kalender Nachrichten bekommt, die an die Mobilnummer des Anwenders versendet werden sollen. Mit Hilfe eines solchen Dienstes kann ein User wenige Minuten vorher an einen Termin erinnert werden.

All diese Peripherie wird von einem zentralen Kalender-Server verwendet, um Daten einzuholen und Informationen zu versenden. Der innere Aufbau eines solchen Dienstes hat ebenfalls bestimmte Komponenten, die ihre eigenen Aufgaben haben: Die Datenbank speichert die Termindaten der User. Ob es sich bei dieser Komponente um ein kommerzielles Produkt oder um eine proprietäre Implementierung handelt, ist Sache des jeweiligen Herstellers.

Der Serverprozess hat mehrere Aufgaben:

- Er stellt Schnittstellen zur Verfügung, über die verschiedene Clients mit dem Server die Daten austauschen können.
- Er verwaltet den Verbindungsaufbau der Clients und nimmt deren Authentifizierungsdaten entgegen, um sie mit den User-Daten aus dem Directory zu vergleichen. Damit stellt er sicher, dass nur berechtigte Zugriffe auf die Datenbank möglich sind. Durch die User-Identifikation kann er sicherstellen, dass die Anwender nur ihre eigenen (oder die explizit freigegebenen) Daten lesen können.
- Er nimmt Datenanforderungen entgegen, erzeugt die jeweiligen Sichten und liefert sie an die CUAs gemäß den Vorgaben aus den Profildaten des Directory.

- Er nimmt Datenänderungen entgegen. Clients übertragen neue Termine oder modifizieren Daten, die vom Server in der Datenbank umgesetzt werden. Dabei kontrolliert er die Transaktionen und stellt die Datenkonsistenz sicher.
- Er macht auf Terminüberschneidungen aufmerksam, schlägt Alternativen vor und blockiert Zeitspannen, die reserviert werden.
- Er triggert Ereignisse, die durch Termine bedingt sind. So werden Erinnerungsnachrichten erzeugt, die über SMS oder Mail an die Clients versendet werden.
- In verteilten Kalenderumgebungen synchronisiert er sich mit anderen Kalender-Servern.

Ein netzbasierter Kalender reduziert sich damit nicht auf einen einzigen Dienst, sondern stellt ein komplexes Gebilde dar.

10 Proxy

10.1 Einführung

Mit der Einführung von multimedialen Daten in Netzwerken hat sich der Bedarf an Bandbreite drastisch erhöht. Während sich früher der Austausch weitgehend auf Textinformationen beschränkte, werden heute immer mehr Bilder, Musikstücke und Videos versandt. Selbst einfache Texte werden nicht mehr nur mit dem reinen Inhalt verschickt. Vielmehr werden diese Daten mit Darstellungsanweisungen für die Browser versehen, damit der Text in der richtigen Schrift mit der entsprechenden Farbe und einer angemessenen Größe dargestellt wird. Der eigentliche Nutzanteil der Informationen in Relation zum Gesamtaufkommen der Daten wird immer geringer. Um diesen Volumenzuwachs immer noch einigermaßen schnell durch das Netzwerk übertragen zu können, müssen die Übertragungswege performanter ausgelegt werden. Ein Maß für diesen Qualitätsfaktor ist die Bandbreite eines Netzwerks, die in Übertragungsraten gemessen wird. Mit einer einkanaligen ISDN-Datenübertragung hat man theoretisch eine Übertragungsrate von 64 Kbit/s, die man aber in der Regel nicht erreicht. Grund dafür ist der immer geringere Anteil an Nutzdaten im Vergleich zu den tatsächlich übertragenen Informationsmengen. Aber auch Engpässe auf Seiten des Providers oder im Internet können für die eingeschränkte Datenübertragungsrate verantwortlich sein. Dabei stehen Unternehmen vor der Entscheidung, eine teure breitbandige Datenverbindung zu einem Internet Provider einzurichten oder in Kauf zu nehmen, dass Mitarbeiter viel Zeit damit verbringen, auf Daten aus dem Netzwerk zu warten.

Ein anderer Aspekt bedeutet für viele Unternehmen ein wichtiges Anliegen: die Zugriffskontrolle auf das Internet. Hier findet sich eine solche Vielfalt an Informationen, dass ein Arbeitsplatz mit Internetanschluss den Mitarbeiter sehr leicht in Versuchung führt, dieses Medium während der Arbeitszeit nicht nur ausschließlich für dienstliche Zwecke zu nutzen. Es werden Bankgeschäfte getätigt, Nachrichten gelesen, Bücher bestellt und vieles mehr. Mancher Arbeitgeber versucht nun, die Zugriffe so zu reglementieren, dass nur noch unternehmenswichtige Daten freigeschaltet werden, während andere Inhalte mit aufwändigen Filtern abgewiesen werden.

Proxyserver setzt man bevorzugt für den Webzugriff ein, aber auch ftp-Zugriffe erfolgen über diesen Dienst. Die nächsten Abschnitte befassen sich mit den Möglichkeiten, Vorteilen, aber auch Randerscheinungen, die die Installation eines Proxyservers in einem Netzwerk mit sich bringt.

10.2 Caching

Jede Verbindung ins Internet hat eine obere Grenze der Übertragungskapazität. Ob man mit einem analogen Modem, über ISDN, DSL oder eine Standleitung die Verbindung aufgebaut hat, immer stellt die Bandbreite eine Beschränkung für die Geschwindigkeit dar, mit der die Daten aus dem Netz kommen. Aus dieser Überlegung heraus entstand die Idee, Daten, die immer wieder aufgerufen werden, zwischenzuspeichern. Gerade Bilder oder Grafiken, die sich relativ selten ändern, können »gecached« werden. Einmal aus dem Internet übertragen, werden diese Daten für spätere Gelegenheiten gespeichert und müssen nicht immer wieder neu aus dem Internet angefordert werden. Die Vorteile sind Zeit- und Kostenersparnis. Liegen die Daten bereits im lokalen LAN vor, können sie mit der vollen Ethernet-Geschwindigkeit übertragen werden und der Anwender braucht nicht lange auf die Seite zu warten. Der Kostenfaktor wirkt sich sowohl für den Privatanwender als auch für ein Unternehmen aus: weniger Online- wie Arbeitszeit wird für das Warten auf den Download verbraucht.

End-User, die mit dem Browser im Internet surfen, verwenden einen clientseitigen Cache, um bereits übertragene Daten zwischenzuspeichern. Dieser Cache hat aber auch seine Nachteile: die geringe »Intelligenz« bei der Verwaltung der Daten und die Belegung der Festplatte des Users mit temporären Internetdaten.

Unternehmen verwenden einen professionelleren Ansatz: den Cache eines Proxyservers. Diese Lösung sorgt für einen performanten Internetzugang, verbunden mit einer verlässlichen Aktualisierungsprüfung der bereits vorliegenden Daten. Sie haben ausgefeilte Speicherstrategien und können bei mehrstufigen Kaskaden eine intelligente Verteilung der Requests auf die verschiedenen Proxys umsetzen. Mit der Einführung eines Cache in einem Unternehmen reduziert sich automatisch das Datenvolumen der externen Internetanbindung. Das reduziert auch die Notwendigkeit für eine teure Aufrüstung dieser Verbindung.

Man geht davon aus, dass die Zeit, bis sich die Investition eines Proxyservers amortisiert hat, bei den meisten Unternehmen zwischen sechs Monaten und zwei Jahren liegt.

Betrachten wir als Beispiel wieder unsere Firma »meinefirma.com«. Der Geschäftsführer hat sich entschlossen, die Arbeitsplätze der Mitarbeiter mit einem Internetanschluss zu versehen, damit sie notwendige Daten schnell und bequem aus dem Internet laden können. Joe User möchte nun eine Unternehmenspräsentation eines Konkurrenzunternehmens von deren Webseite laden, um sie zu analysieren. Diese ist in Form einer *html*-Seite auf deren Webserver verfügbar und mit animierten Grafiken, Adobe-Dokumenten und Fotografien versehen. Das Datenvolumen ist insgesamt recht hoch und es erfordert vom Mitarbeiter insgesamt einige Minuten Wartezeit, um die gesamte Dokumentation aus dem Internet zu laden.

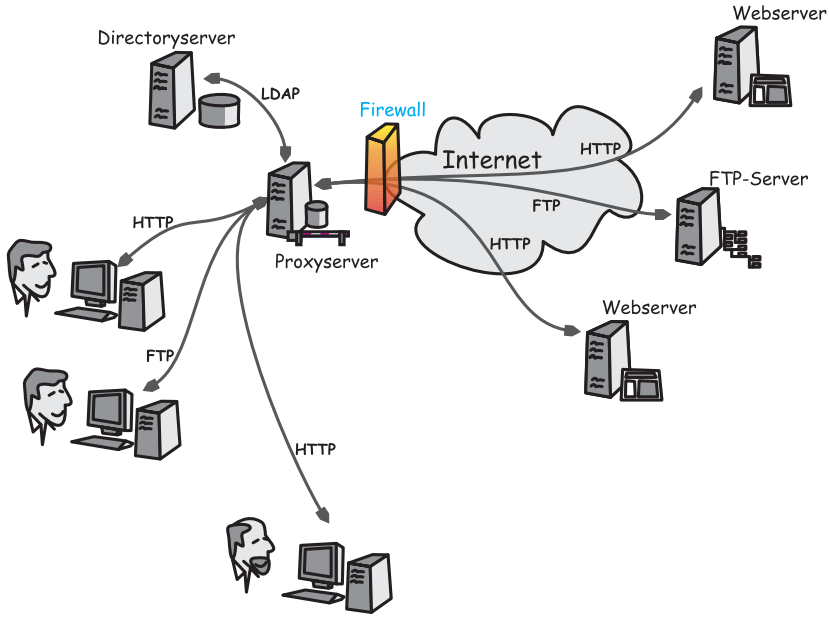


Abbildung 10.1: Einstufige Proxy-Topologie mit Cache und Zugriffskontrolle

Summiert man diese Wartezeiten für alle Mitarbeiter, die sich mit ähnlichen Aufgaben befassen, kommen je nach Unternehmensgröße über alle Mitarbeiter gerechnet pro Jahr schnell einige Manntage Wartezeit zusammen.

Geht in unserem Beispiel Joe User mit seinem Browser über einen Proxy-Dienst, der stellvertretend für ihn die Daten aus dem Internet besorgt und sie, bevor sie an den Anwender ausgeliefert werden, im eigenen Speicherbereich hinterlegt, können sie später wieder verwendet werden. Die Wartezeiten für nachfolgende Anwender, die auf die gleichen Daten zugreifen, verkürzen sich damit drastisch.

Viele Unternehmen sind heute auf Cache-Architekturen angewiesen. Ohne diese würde die externe Internetanbindung kaum ausreichen, um das benötigte Datenaufkommen bereitzustellen. Dies nutzen auch Internet Service Provider, die komplexe Cache-Topologien aufbauen, um dem Kunden einen schnellen Zugriff auf die Daten zu gewähren.

Die folgenden Abschnitte befassen sich mit diesen Topologien und Techniken.

10.2.1 Caching on Demand/Command

Die meisten Proxyserver bieten zwei Möglichkeiten des Downloads: bei Bedarf (»on Demand«) und auf Anweisung (»on Command«).

- »on Demand«: Der Benutzer, der über den Proxyserver ins Internet auf Daten zugreift, initiiert das Zwischenspeichern der Daten. Alle zwischengespeicherten Daten im Cache sind von einem Anwender angefordert worden.

- »on Command«: Per Konfiguration wird der Proxyserver veranlasst, selbständig auf Webseiten zuzugreifen, um diese für einen eventuellen späteren Zugriff eines Benutzers bereits im Cache vorzuhalten. So kann der Administrator eines Internet Service Providers die Webseite »www.total_wichtig.de«, die jeden Tag von vielen Benutzern angefordert wird, vorsorglich jede Nacht um zwei Uhr vom Proxy herunterladen lassen.

10.2.2 Aktualität der Daten

Das Zwischenspeichern von Dokumenten birgt das Problem, dass sich das Original verändern kann und damit alle Kopien der ursprünglichen Fassung veraltet sind. Für den Anwender, der einen Cache verwendet, bedeutet das, dass er unbewusst mit den alten Daten arbeitet. Um dies zu verhindern, benötigt der Proxyserver einen Mechanismus, der sicherstellt, dass alle an den Anwender ausgegebenen Daten in der jeweils aktuellen Fassung vorliegen.

Verfallsdatum eines Dokuments

Wird ein Dokument über einen Proxy angefordert, so werden durch *http* Header-Informationen bezüglich des Dokuments übermittelt.

Bei einem Request wird der Proxy aufgefordert, ein Dokument zu laden, das in der URL spezifiziert ist. Stellt der Proxy fest, dass dieses Dokument nicht im Cache vorhanden ist, muss er es entweder von einem übergeordneten Proxy oder direkt vom Quellserver laden. Die Version wird in einem Zeitstempel dokumentiert.

Bei der Beantwortung des Request durch den Original-Server wird im *http*-Header mit dem eigentlichen Dokument das »date«-Attribut übermittelt, in dem der Zeitstempel der Anfrage festgehalten wird. Diese Information wird von den tiefer liegenden Proxys unverändert übernommen.

Das Alter eines Dokuments im Cache ist die Differenz der Zeit zwischen einem erneuten Request und dem Wert des »date«-Attributs. Fordert also nach einiger Zeit ein anderer Client das gleiche Dokument erneut an, so findet der verwendete Proxy dieses Datum in seinem Cache und ermittelt das Alter dieser Kopie anhand der Differenz zwischen aktuellem Datum und dem Wert des »date«-Attributs.

Dem Proxyserver stehen zwei Vorgehensweisen offen: Er erfragt bei einem Client-Request selbständig die Aktualität des Dokuments am Ursprungsserver oder er liefert ein Dokument, das ein bestimmtes Alter nicht überschritten hat, direkt aus. Wird dieses konfigurierte Verfallsdatum überschritten (dieses Datum ergibt sich aus dem »date«-Wert plus einer Dauer, die am Proxyserver festgelegt wird), lädt der Proxyserver die Information neu, wenn sie angefordert wird. Manche Server löschen automatisch die Dokumente, deren Verfallsdatum überschritten ist, um Speicherplatz für den Cache freizumachen.

Der Ursprungsserver kann aber auch selbst bestimmen, wie die Aktualität des Dokuments eingehalten wird: Der Autor der Seite kann ein Verfallsdatum in die Meta-Informationen des Dokuments einsetzen oder ein Maximalalter bestimmen, das eine Kopie in einem Cache haben darf, ehe diese Kopie als veraltet gilt. Im ersten Fall handelt es sich um die Kombination »Expires: <Datum>«, im zweiten um »Cache-Control: <Datum>«. Mit dem Festlegen eines Verfallsdatums hat man als Autor einer Webseite das Problem, dass man es sich eigentlich nicht erlauben kann, vor Ablauf dieser Zeit Änderungen am Dokument vorzunehmen. Es befinden sich ja Kopien des Ursprungsdokuments bereits in unzähligen anderen Caches, die erst nach Ablauf des Verfallsdatums das nun geänderte Dokument anfordern. Ähnlich sieht die Situation für das »Cache-Control«-Attribut aus, das im schlimmsten Fall einen Proxy dazu verleitet, während der gesamten Dauer einer Alterungsperiode alte Daten auszuliefern.

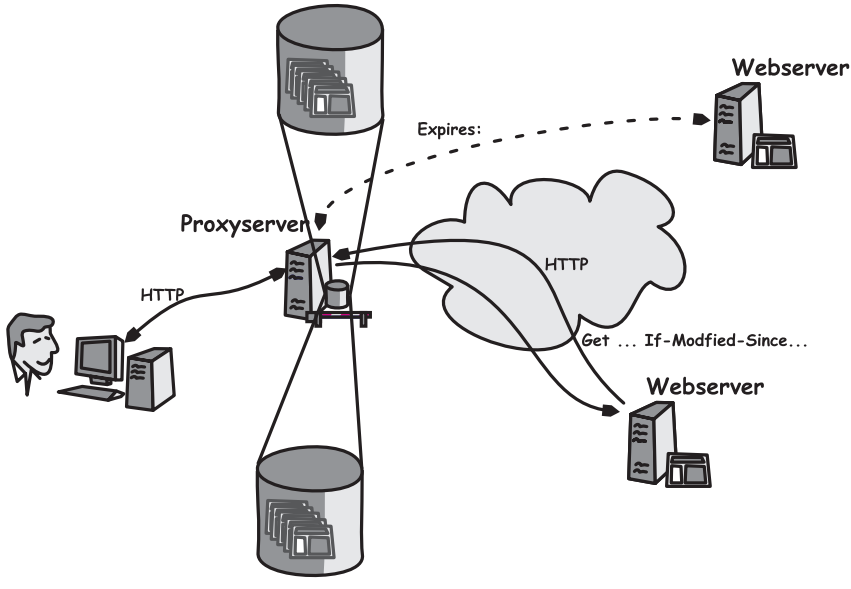


Abbildung 10.2: Verfallsdatum und »If-Modified-Since«: zwei Proxy-Strategien

Noch ein Kontrollattribut innerhalb eines Dokuments ist der »Last-Modified«-Parameter. Dieser gibt an, wann das Dokument zuletzt geändert wurde. Dieser Wert legt nicht fest, wann das Dokument verfällt, sondern gibt dem Proxy einen Hinweis, wie volativ dieses Datum ist. So ist die Wahrscheinlichkeit, dass sich ein Dokument in naher Zukunft verändern wird, ungleich höher bei einem Dokument, das sich vor ein paar Stunden geändert hat, als bei einem, das seit zehn Jahren unverändert geblieben ist. Diese Information kann ein Proxy verwenden, um eine Aktualisierungsstrategie zu entwickeln, indem er sich eine »eigene« Verfallszeit der Kopie im Cache errechnet: Angenommen, der Server hat ein Dokument im Cache, das einen Last-Modified-Wert D1 aufweist. Das Dokument wurde zu einem Zeitpunkt D2 vom Ursprungsrechner

geladen (festgehalten im Attribut »date:«). Der Server ermittelt die Differenz zwischen D1 und D2 und addiert einen konfigurierbaren Faktor von üblicherweise zehn Prozent hinzu. Damit wird ein heuristischer Wert ermittelt, nach dem der Proxy ein Verfallsdatum bestimmt.

Beispiel: Der Proxy lädt ein Dokument von einem Webserver mit dem Wert »last-modified: 31.12.2001«. Das Dokument wird am 30.06.2002 vom Server geladen. Der Proxy »weiß« somit, dass sich innerhalb eines halben Jahres keine Veränderungen am Dokument ergeben haben. Mit einem Aufschlag von zehn Prozent ergibt sich ein Verfallsdatum von 182 Tagen*10%= 18 Tage. Nach Ablauf dieser 18 Tage (am 18.07.2002) verfällt das Dokument im Cache und wird erneut angefordert. Anschließend beginnt ein erneuter Zyklus nach dem gleichen Prinzip.

Zwischen den genannten Werten zur Dokumentenkontrolle, »Expires:« und »Cache-Control:« auf der einen und »Last Modified:« auf der anderen Seite, ist ein Kompromiss zu finden, der es dem Cache ermöglicht, eine sinnvolle Vorhersage des Verfallsdatums durchzuführen.

- Expires: birgt die Gefahr, dass ein Webadministrator das Dokument vor Ablauf des Verfallsdatums ändert, ohne dass die Kopien in den Caches über diese Änderungen informiert werden. Dieses Attribut ist nur sehr selten in den Webdokumenten enthalten, weil es kein notwendiges Attribut ist und von vielen Autoren einfach vergessen wird.
- Cache-Control: liefert zwar ein Verfallsdatum für das Dokument, verhindert aber nicht, dass einzelne Caches eine Weile mit alten Daten arbeiten, bevor sie nach Ablauf der Verfallsperiode erneut vom Ursprungsserver geladen werden. Auch hier ist das Attribut für ein Dokument keinesfalls vorgeschrieben und wird eher selten verwendet.
- Last Modified: weist in Tests mit Proxy-Kaskadierungen Probleme auf, ist aber ein Attribut, das von jedem Dokument geführt wird, da es sich hierbei um den Zeitstempel der Dateierzeugung handelt, der auf Seiten des Filesystems erzeugt wird.

Up-to-date-Check des Servers

Bisher wurde nur beschrieben, wie ein Dokument ein Verfallsdatum erhält, ab dem es nicht mehr gültig ist. Ein Proxyserver kann aber auch direkt am Webserver anfragen, ob sich das Dokument verändert hat oder nicht. In Abhängigkeit von der Antwort bekommt er ein Update des Dokuments oder nicht. In diesem Fall läuft der Cache nicht Gefahr, veraltete Daten an den Client auszuliefern. Diese Konfiguration setzt allerdings voraus, dass der Proxyserver stets mit dem Internet verbunden ist.

Fordert ein Client ein bestimmtes Dokument an, das der Proxy bereits im Cache vorliegen hat, so stellt er gegenüber dem Webserver des Originaldokuments eine direkte Anfrage über das *http*-Protokoll. Dabei verwendet er das »date:«-Attribut aus der Kopie im Cache und formuliert einen Get-

Request, in dem er den Webserver auffordert, ihm nur dann das Dokument zu übersenden, wenn es sich um eine neuere Version handelt:

```
>telnet proxy.meinefirma.com 8080
GET http://www.meinefirma.com/marketing.html HTTP/1.0
If-Modified-Since: Tue, 19 Sep 2001 19:18:02 GMT; length=924
...
HTTP/1.1 304 Not Modified
Date: Wed, 10 Jan 2001 06:30:57 GMT
...
```

Für den Fall, dass das Dokument verändert wurde, übersendet der Webserver die aktualisierte Fassung, andernfalls den Serverstatuscode 304 »Not Modified«.

Up-to-date-Check durch den Client

Ein Anwender möchte sich nicht immer darauf verlassen, dass das Dokument, das aus dem Cache des Proxyservers in der aktuellen Version vorliegt. In diesem Fall erzwingt er über die Kombination »Shift« und »Reload« am Browser den direkten Download des geforderten Dokuments auf dem Original-Server. Über das HTTP-Protokoll wird dies über das so genannte »Pragma«-Feld ausgelöst:

```
>telnet proxy.meinefirma.com 8080
GET http://www.meinefirma.com/marketing.html HTTP/1.0
Pragma: no-cache
...
HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
Date: Fri, 18 Jan 2002 19:11:57 GMT
...
```

Vor- und Nachteile der verschiedenen Architekturen

Mit dem Einsatz von Proxyservern wird der Netzwerk-Traffic um einiges reduziert, da eine Vielzahl von Bildern, Grafiken und Textdokumenten nur noch einmal übertragen werden müssen.

Verlässt man sich bei der Konfiguration eines Proxy auf ein Verfallsdatum, bei dem ein Request nicht automatisch am Originalserver verifiziert wird, kann es dazu kommen, dass der Proxyserver veraltete Daten liefert. Sicherer ist die direkte Abfrage der Version am Originalserver, ohne explizit das Dokument selbst anzufordern.

Für den Betreiber einer Webseite ergibt sich beim Einsatz von Proxy-Diensten der Nachteil, dass die Zugriffsstatistik verfälscht werden kann. Für den Fall, dass ein Cache einen Request auf ein Dokument nicht an den Originalserver berichtet, fehlt dieser Zugriff in der Statistik. Gerade diese Zahlen sind aber für das Marketing des Betreibers einer Webseite wichtig.

10.2.3 Proxy-Routing

Wenn für kleine und mittlere Unternehmen ein einzelner Proxyserver ausreicht, um den Internettraffic zu bewerkstelligen, so ist dies bei größeren Unternehmen bis hin zu Internet Service Providern nicht mehr der Fall. Hier ist man darauf angewiesen, effiziente Proxy-Architekturen aufzubauen, die ausfallsicher und performant den Internetzugriff für die Benutzer bereitstellen. Hierzu gehört die Verteilung der Last auf verschiedene Proxyserver.

Proxy-Autokonfiguration

Ein Browser kann über die manuelle Konfiguration meist nur einen Proxy für den Internetaccess konfigurieren. Fällt dieser Server aus, muss der Benutzer einen anderen Proxyserver eintragen, was viele Anwender bereits vor zu hohe Anforderungen stellt. Eine Alternative hierzu ist das Proxy-Autokonfigurations-File (pac-File). Diese Datei wird vom Proxyserver bereitgestellt und auf Seiten des Browsers gespeichert.

Das pac-File wird mit jedem Start des Browsers neu vom Proxyserver geladen. Fällt der Server aus, so werden die »alten« Daten des gespeicherten pac-Files auf Seiten des Browsers für die Konfiguration herangezogen.

Im pac-File stehen Daten, die den Browser anweisen, was er zu tun hat, wenn bestimmte Konstellationen eintreten. Hier einige Beispiele:

- Der Browser soll die Seiten www.meinefirma.com ohne Zuhilfenahme eines Proxy kontaktieren. Dies wird insbesondere bei internen Seiten verwendet, bei denen die Performance des direkten Zugriffs nicht durch einen Proxy gesteigert werden kann.
- Wenn der Default-Proxy nicht erreichbar ist, wird der Browser angewiesen, einen zweiten Server zu kontaktieren. Damit verhindert man einen Single Point of Failure. Jeder Browser, der das pac-File verwendet, kennt damit einen Failover-Server.
- Der Browser kann so konfiguriert werden, dass er bei bestimmten Domains den Proxy nicht verwendet, sondern direkt den Server des Request kontaktiert.
- Der Browser kontaktiert für bestimmte Domänen spezielle, dafür vorgesehene Proxyserver. Man kann z.B. vorgeben, dass der Client für die Topleveldomänen ».com« den Proxyserver »proxy1.meinefirma.de«, für die Domänen ».de« den Proxyserver »proxy2.meinefirma.de« und für ftp-Zugriffe den Server »proxy3.meinefirma.de« kontaktieren soll.

Über pac-Files lassen sich damit relativ einfach Strategien an die Clients verteilen, ohne dass der Anwender mit deren Konfiguration belastet werden muss.

10.2.4 Verteiltes Cachen von Daten

Der einfachste Ansatz, die Latenzzeit für den Datenzugriff zu reduzieren, ist die Bereitstellung eines lokalen Cache in jedem Subsystem. Die Clients verbinden sich mit dem Proxy, der seinerseits die angeforderten Daten lädt, sie im Cache hinterlegt und anschließend an die Clients weitergibt. Dabei gehen die Proxyserver direkt auf die entfernte Seite, unabhängig davon, ob ein benachbarter Proxy diese Information bereits hat oder nicht. Auf diese Art entsteht mit der Zeit eine Redundanz in den verschiedenen Caches, die nicht untereinander abgeglichen wird.

Der nächste bessere Ansatz ist die dynamische Abgleichung der Caches untereinander. Dadurch wird vermieden, dass im gleichen Netzwerk ein bereits zwischengespeichertes Dokument mehrfach in den Proxycaches auftaucht. Für den Client bleibt der Zugriff nach wie vor auf einem einzigen Proxy. Dieser aber »fragt« die benachbarten Caches ab, ob das angeforderte Dokument vorhanden ist. Erst wenn dies nicht der Fall ist, wird der Originalserver des Dokuments direkt kontaktiert. Mit der Vermeidung von Redundanzen vergrößert sich der zu verwendende Speicherplatz, so dass man insgesamt eine höhere Hitrate für den Datenzugriff erreicht.

Es kann vorkommen, dass bei einer zu hohen Zahl benachbarter Caches die Inter-Cache-Kommunikation so hoch wird, dass sie der Zugriffszeit entgegenwirkt: Ein lokaler Cache fragt zuerst all die vielen benachbarten Proxys ab, bevor er sich auf direktem Weg das Dokument besorgt. Dieses Verhalten addiert sich mit jedem weiteren gleichwertigen Proxy, so dass alle Services damit beschäftigt sind, Anfragen zu stellen und gleichzeitig zu beantworten. Um dem entgegenzuwirken, setzt man hierarchische Proxy-Topologien ein.

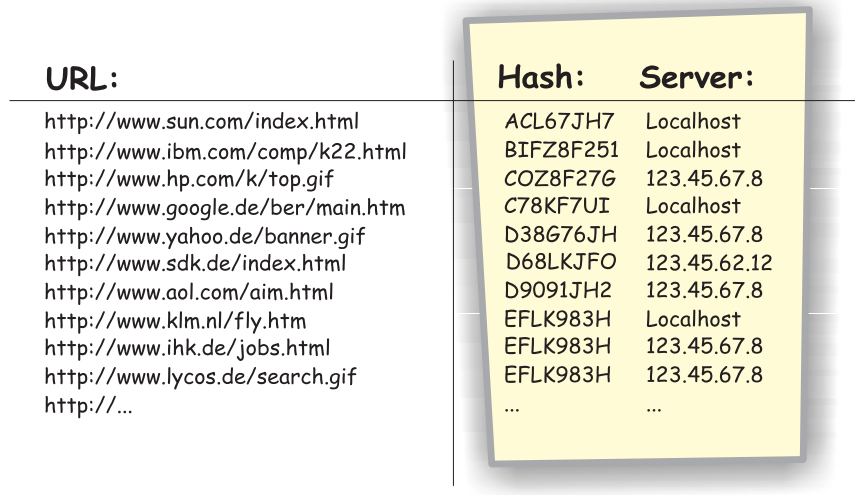
In den nächsten Abschnitten werden verschiedene Strategien und Architekturen für diese Fragestellung diskutiert.

Cache: Verwaltung mit Hash-Tables

Woher »weiß« ein Cache, welches Dokument er hat? Diese Information wird im Speicher gehalten, auf den sowohl der Cache als auch benachbarte Proxys gleichermaßen zugreifen können. Diese Information ist aber nicht in Plaintext-Versionen als einziges Datenfile gespeichert, sondern der Proxy generiert aus einer URL einen Hash und versieht diesen Wert mit der Information, ob das entsprechende Datum im Cache vorliegt oder nicht. (Hashes sind Einwegfunktionen, siehe Kapitel 6.5.1 für weitere Informationen.) Eine URL wie zum Beispiel »www.meinefirma.com/marketing/analysen/1998/statistik.html« kann über eine Hash-Funktion in den Wert »xyz123« abgebildet werden. Diese Hash-Tabelle wird mit den Daten der anderen Caches kombiniert, so dass jeder Proxy über den Datenbestand des anderen Bescheid weiß. Fordert ein Client über den lokalen Cache ein Dokument an, wird dessen URL gehashed und der resultierende Wert mit der Hash-Tabelle verglichen. So kann der Proxy relativ schnell herausfinden, ob er selbst oder ein anderer Server das Dokument bereits im Cache vorliegen hat oder nicht.

Hash-Tabellen sind relativ klein und können leicht komprimiert werden. Updates dieser Tabellen generieren damit keinen allzu großen Aufwand im Netzwerk und können somit regelmäßig zwischen den Servern ausgetauscht werden.

Abbildung 10.3: Prinzip einer Hash-Tabelle, die ein Proxyserver verwaltet



Serverseitiges Proxy-Routing

Die meisten Proxyserver unterstützen die bedingte Routing-Konfiguration. Das bedeutet, dass am Server eingestellt wird, dass er bei einem Client-Request nicht sofort den Zielserver kontaktiert, sondern die Anfrage an einen weiteren Proxy übergibt. Man findet diese so genannte »Proxy-Chaining«-Konfiguration in größeren Unternehmen oder bei Service Providern, bei denen die lokalen Proxys Anfragen direkt abwickeln und bei dann noch nicht bearbeiteten Requests den übergeordneten Proxy kontaktieren. Man versucht mit dieser Architektur, die Netzwerkaktivität weitgehend lokal zu halten und damit Bandbreite und Verbindungskosten zu sparen.

icp

Damit Caches miteinander Daten abgleichen können, ist ein Protokoll notwendig, über das sie die Kommunikation betreiben. Eines dieser Protokolle ist das Internet Cache Protocol (*icp*). Diese Architektur verfährt wie folgt:

1. Der Client stellt einen Request, der beim konfigurierten lokalen Proxy eintrifft.
2. Ist das Dokument im Cache vorhanden ist, wird der Request beantwortet. Wenn nicht, werden benachbarte Proxyserver kontaktiert und überprüft, ob einer dieser Server das angeforderte Dokument im Cache hat. Die Kommunikation geschieht über *icp*. Diese angesprochenen benachbarten Proxys sind alle auf dem gleichen hierarchischen Level.

3. Besitzt einer der benachbarten Caches das Dokument, wird es an den anfragenden Proxy übergeben, der die Daten im eigenen Cache hinterlegt und anschließend an den Client übergibt. Liegt dagegen das Dokument bei den Proxys des gleichen Levels nicht vor, stellt der erste Proxy die Anfrage an seinen übergeordneten Server, seinem »Parent-Proxy«.
4. Der Parent-Proxy kann das Dokument haben und gibt es direkt an den initialen Proxy weiter oder aber er fordert die Daten vom Zielserver an.

Hier wurde ein Szenario beschrieben, das eine zweistufige Proxy-Hierarchie vorsieht. Mit den gleichen Techniken sind auch höherstufige Proxy-Hierarchien denkbar.

carp

Man kann einen verteilten Cache aufbauen, bei dem nicht mehr ein einzelner Cache für das Zwischenspeichern von Internetdaten zuständig ist, sondern ein ganzer Array. Die Verteilung der Aufgaben geschieht über das Cache Array Routing Protocol (*carp*).

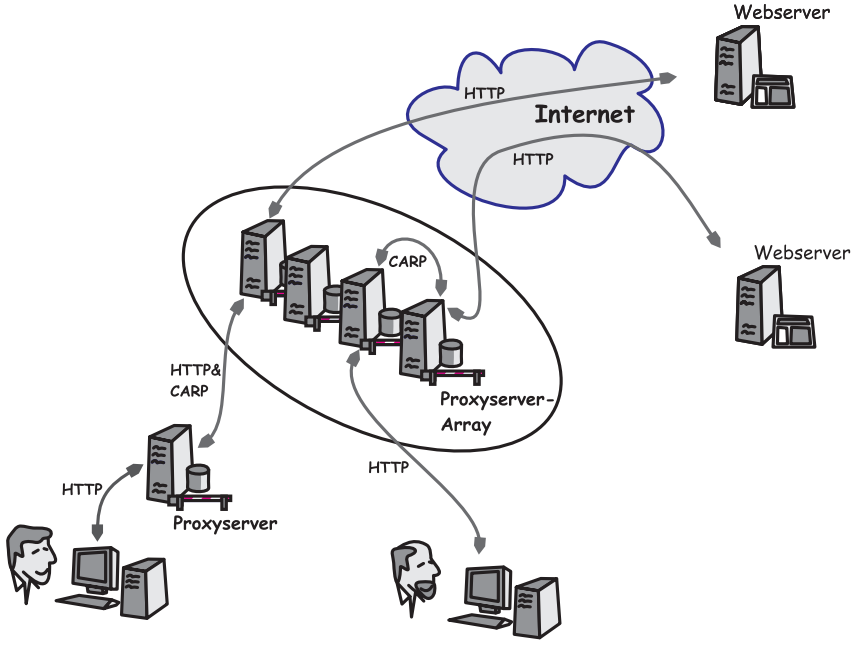


Abbildung 10.4: carp-Architektur

Ein Client adressiert seinen Internetrequest an seinen Default-Proxy mittels einer URL. Der Cache kontaktiert nicht direkt den Ziel-Webserver, sondern ermittelt, welcher der parallelen Proxys in Frage kommt. Dies wird über die Berechnung eines Qualitätsfaktors entschieden:

1. Jeder Proxy in einem *carp*-Array besitzt über seinen Hostnamen einen eindeutigen Hash-Wert. Nehmen wir an, wir hätten in unserer Firma

einen Array von verschiedenen Proxyservern installiert, die von proxy1(.meinefirma.de) bis proxy5 durchnummeriert sind. Entsprechend wird von *carp* jedem Server ein eindeutiger Hash-Wert zugewiesen:

Proxy1: 13

Proxy2: 22

Proxy3: 2

Proxy4: 4

Proxy5: 11

2. Jeder Proxyserver kann nun mit einem Gewichtungsfaktor versehen werden, der es dem Administrator erlaubt, festzulegen, wie viele gleichzeitige Requests der Server bearbeiten kann. Dadurch ergeben sich in der Proxyserver-Hash-Tabelle andere Werte (die Werte sind hier nur exemplarisch):

   Proxy1: 2

   Proxy2: 15

   Proxy3: 19

   Proxy4: 7

   Proxy5: 10

3. Ein Request eines Benutzers wird über eine URL artikuliert. Diese hat ebenfalls einen Hash-Wert, den *carp* ermittelt (die Werte sind hier nur exemplarisch):

   http://www.internet-seite.us: 22

   http://www.uninteressant.nl: 17

   http://www.wichtig.uk: 3

4. Die Kombination der Hashes von Proxys und URLs ergibt einen Qualitätsfaktor, dessen höchster Wert den verwendeten Proxy definiert (die Werte sind hier nur exemplarisch):

Internetseite	Proxy1	Proxy2	Proxy3	Proxy4	Proxy5
www.internet-seite.us	22	18	2	14	14
www.uninteressant.nl	34	16	8	3	31
www.wichtig.uk	52	15	24	7	5

5. Die Proxyserver, die für die betreffende URL den höchsten Hash-Wert liefern, sind primär für den Request verantwortlich. Da die Berechnung der Hashes deterministisch ist und die Performance der einzelnen Proxys berücksichtigt, ist eine ausgewogene Lastverteilung der Client-Zugriffe erzielt.

6. Die Proxys »kennen« sich aufgrund einer Mitgliederliste, in der alle Server eingetragen sind, die am Array teilnehmen. Jeder Server besitzt diese Liste, die regelmäßig aktualisiert wird. Die Liste beinhaltet ferner die Performance-Gewichte der einzelnen Server, Update-Informationen der Liste und »Time-To-Live« Informationen, die den zeitlichen Abstand der Statuschecks der Server untereinander definieren. Diese Liste wird von einem Master-Proxy innerhalb des Arrays gepflegt, von dort beziehen die anderen Server die Mitgliederliste.
7. *carp*-Architekturen besitzen eine einfache Failover-Struktur. Ist ein Proxy, der einen Request bearbeiten soll, nicht verfügbar, springt derjenige in der Hierarchieliste ein, der mit dem zweithöchsten Qualitätsfaktor für eine betreffende URL vorkommt.

Proxyserver, die mittels einer *carp*-Architektur mehrere Server miteinander verbinden, bilden eine intelligente Einheit, die Webdokumente zwischenspeichert. Die Verteilung der Requests und der Cache-Lokationen erlauben es, eine ausfallsichere und performante Proxy-Lösung aufzubauen. Für den Fall, dass einer der Server ausfällt, übernimmt ein anderer den Dienst, die zwischengespeicherten Daten werden schnell und sicher gefunden. Die Zuständigkeiten werden über die Berechnung von Bewertungsfaktoren gesteuert, die sich aus den URLs der Requests ergeben.

10.3 Zugriffskontrolle über Proxy

Wir haben den Proxyserver bisher nur unter dem Aspekt des Zwischenspeicherns von Internetdateien betrachtet. Ein weiteres wichtiges Einsatzgebiet ist das der Zugriffskontrolle.

Ein Mitarbeiter in einem Unternehmen hat einen Internetarbeitsplatz und kann nun auf jedes Dokument im Netz zugreifen, das ihn interessiert. Dies geht nicht immer konform mit der Intention des Arbeitgebers. Dieser hat die Effizienz seines Personals im Sinn, wenn er das Internet als Mittel für die Informationsbeschaffung bereitstellt. Der Arbeitgeber kann im Arbeitsvertrag festhalten, dass das Internet ausschließlich zu Geschäftszwecken und nicht für den privaten Gebrauch verwendet werden darf, aber es gibt kaum einen Arbeitnehmer, der nicht das eine oder andere Mal aus privatem Interesse auf Internetseiten klickt, die nichts mit seiner derzeitigen Beschäftigung zu tun haben. Es liegt im Ermessen des Unternehmens, hier einen Spielraum zu definieren, innerhalb dessen sich der Einzelne legal bewegen kann. Zwei Aspekte spielen hier eine Rolle: die Zeit, die der Mitarbeiter damit verbringt, privat im Internet zu recherchieren, und die Art der Daten, die er sich verschafft.

Surft ein Mitarbeiter häufig privat im Internet, verringert sich seine Produktivität. Hier muss der Arbeitgeber über einen Proxyserver reglementierend eingreifen. Dieser kann so konfiguriert werden, dass er einzelne Mitarbeiter oder ganze Personengruppen nur für einen bestimmten Zeitraum ins Inter-

net lässt. Je nach Produkt ist der entsprechende Konfigurationsaufwand mehr oder weniger hoch. Oft wird ein Directory Server hinzugeschaltet, der den Benutzerdatenstamm bereithält, für die sich dann die zeitliche Limitierung des Internetzugangs definieren lässt.

Der andere Aspekt der Internettätigkeiten der Mitarbeiter betrifft die Art der Dokumente, die geladen werden. Sucht ein Benutzer während seiner Arbeitszeit Informationen zum Urlaubsort, den er im Sommer aufsuchen wird, ordert ein anderer Aktien oder informiert sich über die Tagesnachrichten, so handelt es sich sicher um keine rechtlich bedenklichen Tätigkeiten. Anders liegt der Fall beim Herunterladen von politisch fragwürdigen Dokumenten oder Seiten aus dem Pornobereich. Ein Unternehmen wird sehr wahrscheinlich solche Tätigkeiten seiner Mitarbeiter nicht tolerieren wollen. Darüber hinaus wird es auch nicht akzeptieren wollen, dass der Zugriff aus seinem Netzwerk auf den Logdateien fragwürdiger Server protokolliert wird.

Werden tatsächlich illegale Daten übertragen und auf unternehmenseigenen Datenträgern gesichert, so liegt ein Straftatbestand vor, der zum Teil auch dem Unternehmen angelastet wird. Werden solche Aktivitäten publik, wird das der Reputation der Firma schaden, obwohl diese nicht aktiv beteiligt war. Aus diesem Grund wird für die Mitarbeiter des Unternehmens der Zugriff auf Internetseiten beschränkt. Diese Aufgabe kann ebenfalls von einem Proxyserver übernommen werden.

10.3.1 Topologien eines kontrollierten Internetzugriffs

Das Intranet eines Unternehmens wird beim Übergang zum weltweiten Internet durch eine Firewall geschützt. Haben die unternehmenseigenen Rechner eine legitime IP-Adresse und sieht die Netzwerkinfrastruktur ein funktionierendes Routing ins Internet vor, kann der Anwender über diesen Rechner an der weltweiten Datenkommunikation teilnehmen. Um die Sicherheitspolitik des Unternehmens umzusetzen, kommen Firewalls, Virenscanner und Intrusion-Detection-Systeme zum Einsatz.

Ohne Proxyserver müsste jeder einzelne Rechner, der eine Verbindung ins Internet aufbauen darf, an der Firewall bekanntgegeben werden oder es existiert eine Regel, die besagt, dass jeder ausgehende *http*-Request (und die damit verbundene Datenübertragung aus dem Internet in das unternehmenseigene Netzwerk) generell erlaubt ist. Während der unlimitierte Internetzugriff in den meisten Fällen nicht in Frage kommt, ist die Kontrolle über Firewall-Regeln oft zu komplex. Es ist gerade die Einfachheit von Systemen, die deren fehlerfreie Funktionalität bedingt. Ein System wie die Firewall, die den Angriffen aus dem Internet ausgesetzt ist und ihnen begegnen muss, ist umso sicherer, je einfacher und übersichtlicher die Implementierung ist.

Ein vorgeschalteter Proxy macht es für den Administrator der Firewall (und somit für die Umsetzung der Security Policy) einfacher, den Internetzugang

sicher zu gestalten: Der einzige Client, der ins Internet eine Verbindung aufbauen darf, ist der Proxyserver. Eine solche Topologie hat folgende Vorteile:

- Würde man die internen Internetrequests über die verschiedenen Protokolle (*http*, *https*, *ftp* usw.) für alle User öffnen, wäre diese Masse an Verbindungen nur sehr schwer zu kontrollieren. Würde man umgekehrt jeden Benutzer an der Firewall dediziert freischalten, würde ein umfangreiches Regelwerk entstehen, das nur noch sehr schwer zu überschauen wäre. Beim Einsatz eines Proxy muss nur ein einziger Client eine Internetverbindung haben: der Proxy.
- Die Benutzer sind nicht für den direkten Zugriff ins Internet zugelassen, ein Verbindungsaufbau scheitert an der Firewall. Aus diesem Grund sind alle gezwungen, den Proxyserver zu verwenden. Dadurch wird es für ein Unternehmen einfach, die Internetverbindung zu kontrollieren: Da jeder Benutzer den Proxyserver verwenden muss, kann man an dieser Stelle Filterregeln hinterlegen, die dann für alle gelten.
- Ein Proxy kann so konfiguriert werden, dass er keine Authentifizierungsdaten vom Benutzer verlangt. Damit ist jedem Mitarbeiter der Internetzugriff gestattet, sofern er den Proxy verwendet.
- Ist der Proxyserver in der Lage, über *ldap* auf ein Directory zuzugreifen, so kann der Internetzugriff für die Benutzer individuell gestaltet werden. Dazu werden im Directory Gruppen angelegt, denen man Rechte einräumt oder verwehrt.

  Beispiel: Am Directory Server sind die Benutzerdaten aller Mitarbeiter eingetragen. Es werden Gruppen installiert, denen einzelne Benutzer angehören. Man installiert beispielsweise eine Gruppe »Internetuser« und konfiguriert den Proxyserver so, dass alle Mitglieder dieser Gruppe das Rechte zum Internetzugriff haben. Allen anderen Anwender soll dieses Privileg verwehrt sein.

  Die Kontrolle des Proxyservers geschieht wie folgt: Ein Anwender versucht, aus dem Internet eine Datei über den Proxyserver anzufordern. Gleich zu Beginn des Request überprüft der Proxy die Identität des Benutzers, indem er ihn auffordert, seine Authentisierungsdaten (User-ID/Passwort oder clientseitiges Zertifikat) anzugeben. Diese Daten werden zunächst auf Gültigkeit am Directory überprüft. Ist dies erfolgreich, überprüft der Proxy anschließend über *ldap* die Gruppenzugehörigkeit. Gehört der Mitarbeiter zur Gruppe »Internetuser«, so wird das Dokument aus dem Internet angefordert, andernfalls wird der Request abgewiesen.
- Die Konfiguration am Proxy kann sehr flexibel gestaltet sein. So kann er angewiesen werden, einen anonymen Internetzugriff auf Zeiten nach Dienstschluss zu verlegen oder bestimmten Netzwerkbereichen gesonderte Rechte zuzubilligen.

Ein Proxy kann sehr flexibel eingesetzt werden, wenn die Verbindung ins Internet kontrolliert werden soll. Er unterstützt die Umsetzung der Sicherheitspolitik und verhindert, dass unkontrollierte Zugriffe erfolgen, die den Unternehmensrichtlinien widersprechen.

Um die Zugriffsrechte auf das Internet umzusetzen, werden im Directory Server Gruppen von Mitarbeitern gebildet, die eine Berechtigung zum Internetzugriff haben sollen. Der Proxyserver fordert von einem Client bei dessen Internetaufruf die Authentifizierungsdaten (User-ID/Passwort oder digitales Zertifikat) an, um die Identität der Person festzustellen. Anschließend lässt er den Directory Server prüfen, ob es sich bei der betreffenden Person um ein Mitglied einer Gruppe handelt, der der Internetzugriff gestattet ist.

10.3.2 Browser-Konfigurationen

Clientseitiger Zugriff auf Anwenderebene

Für den Zugriff bieten sich dem Client drei verschiedene Möglichkeiten der Konfiguration des Browsers an. Wir betrachten als Beispiel das Protokoll *http*. Das beschriebene Szenario ist auch für andere gängige Internetprotokolle gültig.

1. Direkte Verbindung mit dem Internet

   In dieser Einstellung baut der Browser eine direkte Verbindung mit der angeforderten Seite auf. Ist in einem Unternehmen eine Firewall eingerichtet, die nur dem Proxyserver als Client-Applikation den Zugriff auf externe Daten erlaubt, wird dieser Versuch abgewehrt und der Benutzer erhält keine Daten aus dem Internet.

2. Manuelle Proxy-Konfiguration

   Bei dieser Einstellung konfiguriert man für den entsprechenden Zugriff einen Proxyserver, der stellvertretend für den Client die Anfrage stellt und die Daten an den Client ausliefert. Es kann auch festgelegt werden, ob für bestimmte Domänen der direkte Weg unter Vernachlässigung des Proxy gewählt wird. (Man kann damit beispielsweise die Zugriffe auf interne Server direkt ansteuern.)

3. Autoproxy-Konfiguration

   In dieser Einstellung wird eine URL angegeben, die der Browser beim Starten aufruft und von der er ein so genanntes pac-File erhält. Diese Datei enthält Konfigurationsdaten, die dem Browser vorgeben, welchen Weg er ins Internet bzw. auf interne Server zu nehmen hat. Hier werden dem Client Failover-Szenarios und Load-Balancing-Strategien mitgeteilt. Ein solches Proxy-Auto-Configuration-File wird vom Administrator des

Proxy angelegt und jedes Mal neu vom Browser beim Starten angefordert. Ist der Proxyserver einmal nicht erreichbar, kann das zuletzt verwendete pac-File verwendet werden, das als lokale Kopie auf Seiten des Clients hinterlegt wird. Mit dieser Datei wird der Client des Anwenders automatisch konfiguriert.

Diese drei Konfigurationen steuern die Datenverbindung des Clients auf interne und externe Ressourcen.

Clientseitiger Proxy-Zugriff auf Protokollebene

Falls für eine Internetverbindung ein Proxyserver konfiguriert ist, wird nicht der Webserver kontaktiert, an den der Request adressiert ist. Es wird vielmehr eine Verbindung zum konfigurierten Proxyserver aufgebaut. Diesem wird der Request übergeben und es wird erwartet, dass sich der Proxy um die Abarbeitung der Anforderung kümmert und das Resultat ausliefert. So wie in einem direkten Webserver-Zugriff eine Verbindung ausgebaut wurde, so wird sie hier über den Proxy ebenfalls über *http* hergestellt. Wir können dies wiederum mit einer Telnet-Session simulieren:

```
telnet proxy.meinefirma.com 8080
GET http://www.sun.com/index.html HTTP/1.1
Host: www.sun.com
...
```

Da die Verbindung zum Proxy und nicht direkt auf die Zieladresse aufgebaut wurde, benötigt dieser die geforderte Adresse in Form einer gültigen URL. Diese wurde hier in Form des Wertes nach dem GET:-Statement übermittelt. Somit hat der Proxy alle erforderlichen Angaben, um seinerseits eine Verbindung zur originären Seite aufzubauen und stellvertretend für den Client das geforderte Dokument anzufordern, nach Erhalt gegebenenfalls im Cache zwischenzuspeichern und dem anfordernden Client zu übergeben.

10.3.3 Reglementierung des Internetzugriffs

Mit den bisherigen Schritten hat ein Unternehmen die Voraussetzung geschaffen, dass sich alle im Netzwerk befindlichen Clients mit der Proxy-Topologie auseinander setzen müssen. Andernfalls kann keine Verbindung ins Internet erfolgen. (Natürlich kann sich ein Mitarbeiter ein Modem auf seinem Rechner installieren, den Telefonapparat zum Verbindungsaufbau zu seinem Provider nutzen und im Internet surfen, ohne dass der unternehmenseigene Proxy kontaktiert werden muss. Allerdings ist dies ein sehr gravierender Eingriff in die Sicherheitspolitik des Unternehmens, der nicht nur den Proxy-Dienst, sondern vielmehr auch die Firewall aushebelt. Ein solcher Schritt eines Mitarbeiters ist ein ernst zu nehmender Verstoß gegen die Unternehmenssicherheit, der mit rechtlichen Mitteln geahndet werden muss.)

Für die Umsetzung von Filterregeln gibt es mehrere Strategien:

- das Sperren des gesamten Datenbestands des Internets und im anschließenden Fall das Freischalten einzelner Ressourcen. Der Proxyserver wird angewiesen, jeden Request auf Daten, die im Internet angefordert werden, abzuweisen. Im nächsten Schritt werden einzelne Seiten explizit freigeschaltet. Die Freigabe erfolgt unter dem Gesichtspunkt der Notwendigkeit der Informationen für den Arbeitsablauf. So sicher wie dieser Ansatz für ein Unternehmen auch ist, so wenig praktikabel ist er in der Umsetzung. Jeder Mitarbeiter muss beim Administrator eine erforderliche Seite freischalten lassen. In absehbarer Zeit wird die Liste der freigegebenen URLs länger und länger, eine Kontrolle der einzelnen Seiten wird immer weniger wahrscheinlich und nach einem endlichen Zeitraum existiert eine lange Liste von Seiten, die nicht mehr administriert werden kann. Darüber hinaus verliert der Mitarbeiter Zeit, indem er jedes einzelne Dokument neu beantragen muss.
- Eine andere Art der Reglementierung des Inhalts von Internetdokumenten ist das totale Freischalten aller Internetressourcen, verbunden mit dem anschließenden Sperren einzelner Seiten. Dieser Ansatz ist ungleich toleranter als der vorhergehende, erspart er doch dem Mitarbeiter das Beantragen von Freischaltung. Der Administrator steht allerdings immer noch vor der Sisyphus-Aufgabe, die Liste der gesperrten Seiten ständig zu aktualisieren. Immerhin gibt es kommerzielle Dienste, die Listen anbieten, die man dem Proxyserver eingeben kann, und das Sperren ganzer Kategorien von Seiten erlauben. So sind diese Listen nach Sex, Verbrechen, Drogen, politischer Fragwürdigkeit und vielem mehr gegliedert und der Verantwortliche für den Proxy kann nun die entsprechenden Kategorien auswählen und im Server umsetzen. Die Dynamik der Entstehung neuer Seiten und des Verfalls alter im Internet ist allerdings sehr hoch, so dass dieser Ansatz zwar immer noch praktikabel ist, aber nie lückenlos alles abdeckt, was ein Unternehmen eigentlich unterbinden möchte.
- Der dritte Weg, den ein Unternehmen gehen kann, ist die aktive Filterung. Daten, die aus dem Internet geladen werden, können vom Proxyserver vom Inhalt her untersucht werden und entsprechend einer Filterregel akzeptiert oder abgewiesen werden. Diese Filter werden im Allgemeinen in der Syntax der Regular Expressions umgesetzt, die im nächsten Absatz detailliert besprochen werden. Für den Augenblick genügt es, zu sagen, dass es sich hier um Filter handelt, die auf bestimmte Schlüsselwörter reagieren und dann entsprechende Aktionen initiieren. Beispielsweise könnte ein Filter so konfiguriert sein, dass er alle Daten, in denen der String »sex« vorkommt, nicht akzeptiert und die Daten nicht überträgt. Damit würde dem Mitarbeiter die Möglichkeit genommen, in seiner Arbeitszeit pornografische Seiten herunterzuladen. Leider ist dieser Filter aber auch kontraproduktiv, da er nicht nur nackte

Tatsachen, sondern auch Berichte über »Rechtsextremismus« oder »Sicherheitsexperten« unterbindet. Filter dieser Art müssen demnach sorgfältig definiert und umgesetzt werden.

- Ein vierter Weg ist zwar praktikabel, aber rechtlich nicht umsetzbar. Der Proxyserver hat normalerweise die Möglichkeit, Zugriffe auf den Datenbestand des Internets zu protokollieren. Diese Protokolldaten kann ein Unternehmen zwar theoretisch auswerten, es darf aber keine Eins-zu-eins-Schlüsse aus dem Zugriff auf eine Seite und demjenigen ziehen, der sie aufgerufen hat. Ein Administrator bewegt sich rechtlich in einer Grauzone, insbesondere dann, wenn sich ein Benutzer gegenüber dem Proxyserver authentifizieren muss, um ins Internet zu gelangen. Dann nämlich findet sich in den Protokoll- oder Logdateien neben dem Namen oder der User-ID des Mitarbeiters auch die Seite, die er aufgerufen hat. Handelt es sich hier um eine vom Unternehmen nicht gewünschte Seite, kann der Administrator diese Information an den Vorstand weitergeben. Wird der betreffende Mitarbeiter nun zur Rechenschaft gezogen, kann der Arbeitnehmer das Unternehmen verklagen. Wir werden in einem eigenen Abschnitt über die rechtlichen Grauzonen beim Betrieb eines Proxyservers noch einmal auf dieses Thema zurückkommen. Der Administrator kann also keine Auswertung im Sinne der Zuordnung aufgerufener Seiten vornehmen. Aber er sieht die Zugriffe und kann nun sein Instrumentarium an Filterregeln neu an die Gegebenheiten anpassen und die so entdeckten Seiten mit Hilfe von URL- oder Regular-Expression-Filtern sperren.

Welchen Weg ein Unternehmen auch geht, um die Internettechnologie am Arbeitsplatz einzuführen, es kann nicht Sinn der Sache sein, dass man die Mitarbeiter maßregelt und überwacht. Sicherlich wird es immer wieder Verstöße gegen die Unternehmensrichtlinien geben und es gibt Mittel, diese Verstöße aufzudecken. Letztendlich ist aber ein Appell an die Benutzer sicherlich hilfreicher, als schwer zu administrierende Regeln und Architekturen. Für die wirklich »schweren« Fälle kann man immer noch einen moderaten Maßnahmenkatalog aufstellen, der ein Instrumentarium verschiedener Standardfilter vorsieht. Für diese Fälle werden wir in den nächsten Abschnitten die Umsetzung von Internetfiltern vorstellen.

10.3.4 Inhaltskontrolle über Regular Expressions, Filter und Datenscanner

Das Instrumentarium der Regular Expressions hat bereits ganze Bücher gefüllt. Hier soll ein kurzer Einblick in dieses Thema das Verständnis für die Mächtigkeit dieser Technik vermitteln, um für Filterregeln in Proxy-Konfigurationen etwas gewappnet zu sein.

- Beginn und Ende: »^« und »$«

  Die beiden ersten wichtigen Ausdrücke innerhalb einer solchen Regel sind »^« und »$«, die stellvertretend für »Anfang« und »Ende« stehen.

Beispiele:
»^http://www.meinefirma.de«, »http://www.meinefirma.de$« und »^http://www.meinefirma.de$« sind alle Regular Expressions, die auf die URL »http://www.meinefirma.de« zutreffen.

- Keines, eines oder viele: »*«

  Um Abkürzungen für sich wiederholende Zeichen einzuführen, wird folgender Operator verwendet: »*«

  Dieser Operator steht für null, eines oder viele der vorangegangenen Zeichen.

  Beispiele:
  http://w*.meinefirma.de« steht für »http:vw.meinefirma.de«,
  »http://ww.meinefirma.de«, »http://www.meinefirma.de« usw.

- Irgendein Zeichen: ».«

  Der Punkt (».«) steht als Stellvertreter für jedes beliebige Zeichen. Da er insbesondere auch für sich selbst steht, kann man ihn über ein »Escape« in Form eines Backslash (»\«) wieder direkt spezifizieren (eine kleine Inkonsistenz innerhalb dieses Instrumentariums, da der Backslash letztendlich ebenfalls nur ein Zeichen ist).

  Beispiele:
  »http://...\ . meinefirma\ .com« steht sowohl für »http://www.meinefirma.de« als auch »http://wap.meinefirma.de«.

  »http://.*« steht für jede URL, die mit *http* beginnt. Ein Filter, der damit gebildet wird und in Verbindung mit einem »deny« gebracht wird, sperrt das gesamte Internet über *http*.

- Buchstaben oder Zahlen: »[a-z]«, »[0-9]«

  Damit man spezifizieren kann, ob die Regular Expression in einem bestimmten Bereich Buchstaben oder Zahlen enthält, verwendet man Klammerausdrücke.

  Beispiele:

  »[ab]«:Der String hat an dieser Stelle entweder ein »a« oder ein »b«, http://www.meinefirm[ab].de

  »[a-d]«: Der String hat an dieser Stelle entweder ein »a«, »b«, »c« oder »d«, http://www.meinefirm[a-d].de

Es gibt eine Vielzahl von weiteren Definitionen von Werkzeugen der Regular Expressions, die in einschlägiger Literatur bestens dokumentiert ist. Leider sind die Implementierungen nicht immer übereinstimmend und es gibt verschiedene unterschiedliche Definitionen. Welche dieser »RegEx«-Versionen die betreffende Proxyserver-Implementierung hat, muss vom Hersteller erfragt werden.

Mit den wenigen bisher vorgestellten Techniken der Regular Expressions lassen sich nun recht einfach eine Vielzahl von Filtern für das Internet zusammenbauen. Hierzu ein paar Beispiele:

Deny: http://www\.hacker-online\.de.*

Dieser Filter verweigert den kompletten Zugriff auf den Webserver »www.hacker-online.de«. Er unterbindet aber nicht den Zugriff auf »phreaks.hacker-online.de«. Dies wird erst durch

Deny: http://.*\.hacker-online\.de.*

verhindert. Damit hat man aber noch nicht den Zugriff über ein anderes Protokoll unterbunden. Um dies zu realisieren, sieht der Filter folgendermaßen aus:

Deny: [a-z]*://.*\.hacker-online\.de.*

Neben den RegEx-basierten Filtern gibt es in vielen Proxyservern die Möglichkeit, den Inhalt von Dokumenten nach bestimmten Mustern zu durchleuchten und entsprechende Maßnahmen zu treffen. Eine dieser Möglichkeiten wäre das Einbinden eines Virenscanners, der den Content der angeforderten Seiten nach Virenmustern untersucht und beim Auffinden das geforderte Dokument verweigert. Andere Filter suchen nach bestimmten Tags in Webseiten, die beispielsweise Applets, JavaScript-Snipplets oder ActiveX-Controls ausfiltern. Manche Filter bieten die Möglichkeit, die Daten nach MIME-Typ zu sieben. Je nach Konfiguration werden entweder diese Funktionen aus dem Dokument herausgeschnitten oder die gesamte Seite wird gesperrt.

Die Filterregeln können mit diesen Mitteln beliebig fein eingestellt werden, um sie den Bedürfnissen eines Unternehmens anzupassen. Wie immer aber die Regeln aufgesetzt werden, sie sollten so umfangreich wie nötig, dann aber so sicher wie möglich definiert werden.

10.3.5 Rechtliche Aspekte der Mitarbeiterkontrolle

Die technischen Möglichkeiten gewähren einem Administrator auch einen Einblick in die Aktivitäten der Mitarbeiter, die Auswertung solcher Daten ist jedoch rechtlich umstritten.

Es existiert eine rechtliche Grauzone, die es einem Administrator unter Umständen nicht einfach macht, eine Entscheidung über die zu ergreifenden Maßnahmen zu treffen. Grund ist der Schutz des Persönlichkeitsrechts, das den Mitarbeiter davor bewahrt, dass seine Aktivitäten protokolliert, ausgewertet und ihm zur Last gelegt werden können.

Ein Proxyserver protokolliert dagegen die Internetzugriffe. Anhand der Logdaten kann ein Administrator nicht nur verfolgen, welche Daten aus dem Internet aufgerufen werden, sondern auch, von welcher IP-Adresse dieser Request kommt. Nicht selten wird bei authentifizierten Zugriffen sogar die User-ID in den Logfiles dokumentiert.

Streng genommen ist das Protokollieren in eine solche Logdatei nicht statthaft, da sich dadurch bereits Rückschlüsse auf die Aktivitäten der jeweiligen User ziehen lassen. Hier stellt sich das Gesetz vor die Einzelperson und schützt deren Privatsphäre. Dies gilt sowohl für den privaten als auch für den beruflichen Bereich. Surft ein Mitarbeiter während seiner Arbeitszeit im Internet und kann ihm dies über die Logfiles nachgewiesen werden, so darf ein Unternehmen diese Daten nicht gegen die Person verwenden. Auch wenn ein Unternehmen den privaten Gebrauch des Internets in der Arbeitszeit per Arbeitsvertrag untersagt, kann es den Mitarbeiter aufgrund seines Verhaltens nicht belangen.

Hieraus ergibt sich viel Konfliktpotenzial. Die entgangene Arbeitszeit stellt dabei noch das geringste Problem dar. Kritischer wird es, wenn sich der Mitarbeiter durch sein Verhalten strafbar macht oder den Ruf des Unternehmens gefährdet. Beispiel Kinderpornografie: Der Besitz derartiger Daten ist strafbar. Da dieses Material im Cache des Proxy vorliegt, ist das Unternehmen im Besitz dieser Dokumente. Das einzige Mittel, das das Unternehmen in einem solchen Fall hat, ist, den Inhalt der Proxys zu überwachen und illegales Material sofort zu entfernen. Als weiteres Gegenmittel können diese Seiten gesperrt werden, was technisch gesehen eine sehr schwierige Aufgabe bedeutet, da täglich neue Server Inhalte dieser Art anbieten.

10.4 Reverse Proxy

In der bisherigen Diskussion um den Einsatz von Proxys wurden die Verbindungen immer stellvertretend für den Client durchgeführt. Bei einer Reverse-Proxy-Konfiguration steht dieser Dienst stellvertretend für einen Server.

Zweck einer solchen Topologie ist es, den wirklichen Ursprung der Daten, die von einem Server kommen, zu verschleiern. Angenommen, der Webserver unserer Beispielfirma »meinefirma.com« steht in einem sicheren Netzwerk hinter einer Firewall und beinhaltet neben der Firmen- und Produktpräsentation auch Preislisten und Supportdaten. Ein Angriff auf diese Daten, bei dem beispielsweise die Preise zu unseren Ungunsten modifiziert würden, dürfte potenzielle Kunden vom Kauf abschrecken. Aus diesem Grund ist der Einsatz eines Reverse Proxy vorgesehen, der vor der Firewall des Netzwerks steht und – aus Sicht des Clients – so reagiert, als wäre er der Webserver.

Beispiel: Ein Reverse Proxy ist vor der Firewall unseres Unternehmens »meinefirma.com« installiert. Er reagiert, als sei er ein vollwertiger Webserver. In Wahrheit nimmt er die Anfrage entgegen und reicht sie auf den internen Webserver weiter. Die Beantwortung des externen Client-Request durch den wahren Webserver wird an den Reverse Proxy weitergeleitet, der seinerseits die Daten an den Client übermittelt. Nach außen ist damit der direkte Zugriff auf den Webserver unterbunden. Wird der Reverse Proxy erfolgreich angegriffen, finden sich hier keine sensiblen Daten.

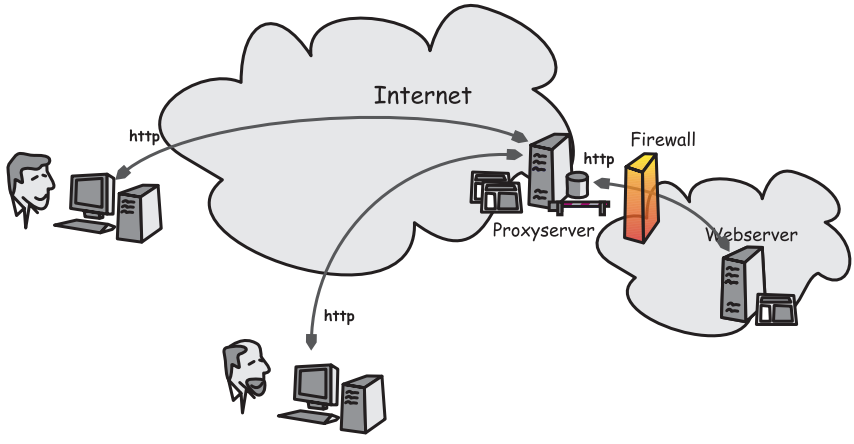


Abbildung 10.5: Reverse-Proxy-Konfiguration

Der wesentliche Unterschied zur bisher diskutierten Topologie ist, dass der Proxy eine aktive Schutzfunktion auf Seiten des Servers übernimmt. Im Gegensatz zur vorher diskutierten Topologie übernimmt dieser Dienst nicht die Stellvertreterfunktionalität für den Client, sondern für den Server. Entsprechend steht dieser Dienst nicht auf Seiten des Anwenders, sondern auf Seiten der Webservers.

Beispiel:

Ein externer Client möchte auf unseren Webserver zugreifen. Er verwendet die URL »www.meinefirma.de/marketing/preisliste.html«. Der Reverse Proxy ist im DNS mit dem Hostname-Alias »www« eingetragen, das heißt, die IP-Auflösung wird die angesprochene URL auf diesen Dienst anstatt auf den eigentlichen Webserver lenken. Der Proxy nimmt den Request entgegen und schreibt ihn um:

Der Client-Request:

»http://www.meinefirma.de/marketing/preisliste.html«

wird vom Proxy in

»http://webserver.meinefirma.de/marketing/preisliste.html«

umgeschrieben und an den internen Webserver hinter der Firewall weitergeleitet.

Damit hat man die Möglichkeit, auf Seiten der Firewall eine Regel umzusetzen, die besagt, dass niemand außer dem Reverse Proxy über das *http*-Protokoll auf das interne Netzwerk zugreifen darf. Hätten wir den Proxy nicht installiert, müssten wir jedem Client aus dem Internet den Zugriff auf die interne Ressource gewähren, was ein Sicherheitsrisiko darstellt.

Wird der Request vom internen Webserver beantwortet und an den Proxy weitergeleitet, so muss dieser noch einmal eine URL umschreiben, da ansonsten der wirkliche Webserver verraten würde:

Die Antwort des internen Webservers mit der URL

»http://webserver.meinefirma.de/marketing/preisliste.html«

wird vom Proxy in

»http://www.meinefirma.de/marketing/preisliste.html«

umgeschrieben, bevor sie an den Client ausgegeben wird.

Wichtig ist, dass dem Client ein wirklicher Webserver simuliert wird. In dessen Browser-Einstellung kann der direkte Zugriff auf die Internetressource eingestellt sein und er wird beim Zugriff auf unsere Firma von einem Proxyserver direkt bedient. Diese Topologie ist (anders als in der vorherigen Diskussion um den Einsatz eines Proxy für den Zugriff auf das Internet) für den Client vollkommen transparent.

11 Portale

11.1 Einführung

Das *html*-basierte Internet ist seit dem Aufkommen der ersten Webseiten um 1994 auf viele Millionen Sites angewachsen. Jedes Unternehmen, jede Gruppierung und Einzelperson kämpft um die Aufmerksamkeit des Benutzers, der durch das Netz »surft«. In Hinblick auf den einzelnen Anwender ist es das ultimative Ziel, die Startseite, also die erste Seite, die beim Starten des Browsers geladen wird, zu erobern. Diese »Heimatadresse« wird vom Benutzer am häufigsten frequentiert und stellt für ihn den Startpunkt ins Internet dar.

Dieses Konzept machte Seiten wie »home.netscape.com« oder »www.microsoft.com« zu den am häufigsten besuchten Seiten. Dies war darauf zurückzuführen, dass Netscape und Microsoft als Hersteller der beiden verbreitetsten Browser ihre Seite als Startseite voreingestellt haben.

Die Betreiber dieser Seiten erkannten sehr schnell die Werbewirksamkeit dieses Effekts: Ein Werbebanner auf dieser Seite wird mit jedem Starten eines Browsers gelesen, die Werbefläche kann teuer verkauft werden und dient als zusätzliche Einnahmequelle.

Die Benutzer besuchen häufig Seiten, die eine Dienstleistung anbieten, insbesondere wenn dieser Dienst kostenlos ist. So kamen die ersten Webmail-Seiten auf, über die man sich kostenlos einen Account verschaffen kann und über den man von überall her sich einloggen, Mails versenden und empfangen kann. Wird ein solcher Dienst angeboten, kommt der Anwender immer wieder auf diese Seite zurück. Damit hat man sich oft bereits einen Platz in der Bookmark-Liste des Anwenders erkämpft. Natürlich wird eine solche Seite mit Werbung versehen, die beim Versenden und Empfangen von Mails stets im Blickfeld des Benutzers ist und teuer verkauft werden kann.

Ein weiterer Schritt in Richtung Startseite ist das Portal. Hierbei handelt es sich um eine Webseite, die mehr oder weniger viele Dienstleistungen anbietet. Neben der Webmail finden sich Nachrichten, Kalender-, Chat- und Suchdienste, man kann seine Bookmark-Liste auf den Seiten eintragen, ein Adressbuch ist vielfach vorhanden und mit Börsenticker und Wetterinformationen ist noch lange nicht alles aufgezählt, was man an Services auf einer solchen Seite anbieten kann. Die wesentlichen Merkmale dieser Portale sind (je nach Portalanbieter kann der eine oder andere Punkt in dieser Liste fehlen):

- Sie sind meist kostenlos.
- Sie bieten eine Vielzahl nützlicher Dienste an.
- Die wichtigsten Services sind auf einer Seite in Form von vielen so genannten »Channels« zusammengefasst.
- Sie sind individuell anzupassen.
- Sie aktualisieren sich selbständig.

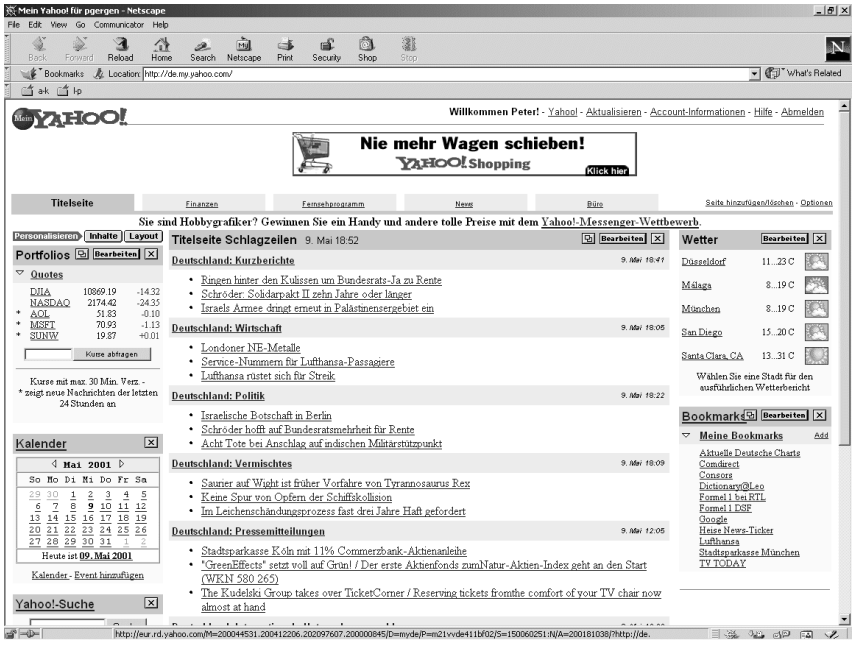


Abbildung 11.1: Beispiel eines personalisierten Portals

Was sind nun die Vorteile für einen Anwender? Was motiviert ihn, diese Seite als Startseite auf seinem Browser einzurichten?

Erstes Argument ist sicher die vertraute Umgebung. Wenn diese Seite immer als erster Anlaufpunkt im Internet erscheint, hat man als Benutzer eine Umgebung, in der man sich automatisch und schnell zurechtfindet. Bei den Bookmarks ist blitzschnell der Verweis auf eine wichtige Seite erreichbar, den Börsen-Ticker zeigen immer an der gleichen Stelle die für den Benutzer relevanten Aktienkurse an, die Nachrichten werden alle paar Minuten aktualisiert. Damit nicht genug, man kann sich den Hintergrund mit Struktur-GIFs tapezieren und die Textfarbe rot gestalten. Kurz: Der Anwender wird in die Lage versetzt, sich seine Heimat im Internet beliebig zu gestalten.

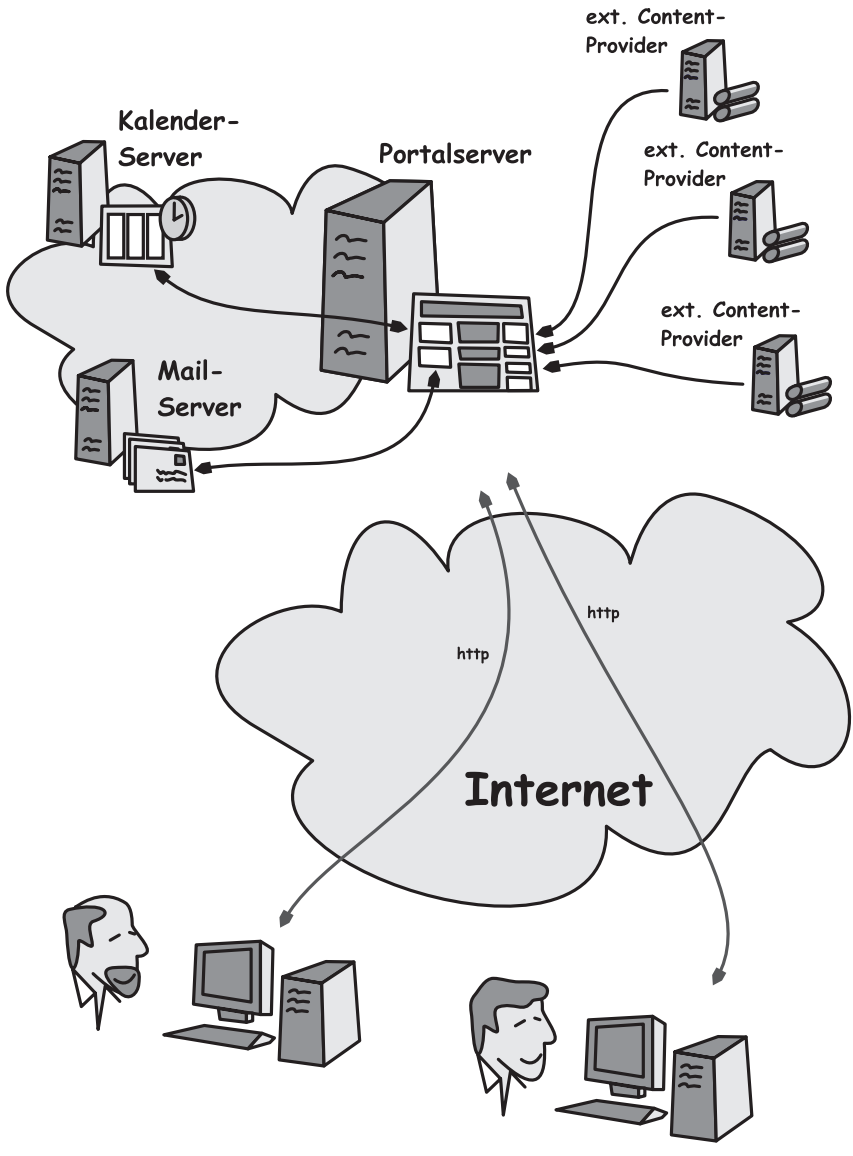


Abbildung 11.2: Portalarchitektur mit externen Content-Providern und Mail/Kalender-Applikationen

Neben der Vertrautheit der Umgebung kommt ein weiterer Faktor hinzu, die direkte Verfügbarkeit. Die Dienste werden überwiegend parallel auf einer Seite zusammengefasst. Das kommt der Trägheit des Anwenders entgegen, der sich ansonsten zu jedem einzelnen Service durchklicken muss. Hier kann man auf einen Blick alle wichtige Informationen erhalten, feinjustiert auf die individuellen Bedürfnisse.

Damit die Dienste angepasst werden können, muss nahezu jeder Service personalisiert werden können. Das heißt, die Börse liefert nur die für den Anwender wirklich interessanten Aktienkurse, Nachrichten werden nach frei wählbaren Sparten angeboten und der Wetterkanal liefert nur die für den Anwender interessanten Daten ausgewählter Wetterstationen.

Vor allem anderen steht der finanzielle Aspekt. Die wenigsten Benutzer würden für eine solche Dienstleistung bezahlen, so praktisch und informativ sie auch sein mag. Warum betreibt ein Unternehmen einen solchen Aufwand kostenlos? Diese Frage wurde eigentlich schon beantwortet: Eine solche Seite ist sehr attraktiv, wird immer wieder angesteuert, ist im Idealfall die Startseite vieler Benutzer und kann als Mittel für Werbung eingesetzt werden. Genau dies ist der Zweck, den ein Betreiber verfolgt. Mit dem Ausliefern der Nachrichten werden die neuesten Produkte aus dem Unternehmensportfolio angepriesen, eine Mail wird mit einem teuer verkauften Werbebanner präsentiert. Ein Portal dient somit als Präsentationsplattform für die eigenen Produkte eines Unternehmens oder als Werbefläche für Drittanbieter.

Die Kehrseite dieser Entwicklung ist, dass basierend auf dem Erfolg einiger weniger Portale in kurzer Zeit viele Nachahmer versuchten, mit einem eigenen Auftritt Kunden auf die Unternehmensseite zu lenken. Plötzlich wurde diese Technik für viele Unternehmen ein Grund, nicht unerhebliche Investitionen zu tätigen und der Websurfer wurde plötzlich überall mit kostenlosen Webaccounts, Chat-Rooms und personalisierten Organizern umworben. Diese Inflation der Dienste zeigt bereits die ersten Abschwächungen. Einige Portale haben bereits wieder geschlossen, weil sich der administrative und finanzielle Aufwand nicht lohnt. Bald werden nur noch die Großen unter den Anbietern überleben.

Von dieser Entwicklung profitieren andere Geschäftsmodelle, die nicht rein werbeorientiert sind, sondern Wert auf Dienstleistung am Kunden legen, beispielsweise Finanzportale, die von Banken betrieben werden und es dem Kunden ermöglichen, sich über ein Portal die benötigten Dienstleistungen zusammenzustellen. Daraus entwickelten sich Bill-Presentment-Lösungen, die portalbasiert dem Kunden eine Übersicht über alle eingegangenen Rechnungen geben und über die er entscheidet, wie er welche Forderungen über welches Konto bezahlen möchte. In einem anderen Channel kann der Anbieter dem Kunden je nach Finanzstatus einen günstigen Autokredit anbieten, ein weiterer Channel zeigt die Übersicht über das Depot mit direktem Link auf die bankinterne Online-Brokerage.

11.2 Portalarchitekturen

Eines der ersten weltweit erfolgreichen Portale war Yahoo!, das aus der anfänglichen Linksammlung einen umfangreichen Service-Katalog zusammengestellt hat. Die ursprüngliche Bestimmung einer Suchmaschine ist um eine Vielzahl zusätzlicher Dienste erweitert worden, was die Attraktivität

dieser Webseite deutlich erhöht. Neben den bereits beschriebenen Diensten werden Auktionen angeboten, Marktplätze betrieben und Kleinanzeigen geschaltet, Spiele, Fernseh- und Kinoprogramme angeboten und Chat-Dienste betrieben.

Einige Funktionalitäten, vorwiegend diejenigen, die vom Benutzer selbständig und aktiv verwendet werden (Kalender- und Adressbücher, Bookmarks und Mailaccounts), werden von solchen Unternehmen selbst betrieben, andere Dienste, insbesondere die Nachrichten-Channels werden von Channel-Providern beliefert. Das heißt, die Nachrichten kommen von Nachrichtenagenturen oder Verlagen, die Informationen werden entsprechend den Vorgaben des Portalbetreibers aufbereitet und in Form von *html*-Daten in die entsprechenden Tabellen- oder Frame-Einsätze der Seite eingespeist.

Es gibt eine Anzahl von Herstellern, die fertige Portalprodukte anbieten, aber viele Betreiber entwickeln immer noch selbst die Plattformen und die dazugehörigen Dienste.

Benutzerprofile

Um sich auf einer solchen Seite einzurichten, muss sich der Benutzer erst einmal anmelden. Diese Registrierung verlangt mehr oder weniger viele Informationen von Benutzer, je nachdem, in welchem Ausmaß der Betreiber Daten über seine Interessenten sammeln möchte.

Damit ein Portal personalisiert werden kann, werden Benutzerdaten in entsprechenden Datenbanken hinterlegt. Diese Daten werden in Benutzerprofilen zusammengefasst, in denen die individuellen Einstellungen des einzelnen Anwenders, der sich an dieser Seite angemeldet hat, hinterlegt werden. Dabei kann ein Anwender je nach Portal entscheiden, welche Channels er abonniert, in welcher Reihenfolge sie angeordnet werden und in welcher Farbe sie angezeigt werden.

Profildatenbanken

An die Datenbanken, die die Profile bereitstellen, werden hohe Anforderungen gestellt. Relationale Datenbanken sind in der Regel nicht schnell genug, um eine genügend große Zahl von »Concurrent Usern« zu unterstützen. Da ein Portal oft als Startseite für den Internetzugang der Benutzer verwendet wird, werden deren Profildaten relativ häufig gelesen. Hier liegt die Last auf Seiten des lesenden Zugriffs. Aber auch wenn der einzelne Anwender sein Benutzerprofil eher selten ändert, so ist es doch die Menge an Anwendern, die eine große Zahl von schreibenden Zugriffen auf den Profilserver ausmacht. Betreibt man die Bereitstellung der Profile über eine relationale Datenbank, so kann die große Zahl der parallelen Zugriffe dazu führen, dass die Datenbank nicht mehr schnell genug ist, um eine ausreichende Performance zu bieten. Wählt man andererseits eine Lösung über einen Directory-Dienst, werden aufgrund der Fokussierung dieses Services auf den lesenden Zugriff Änderungen möglicherweise nur relativ langsam umgesetzt. In beiden Fällen beruht die Lösung auf der Unterstützung durch Caching-Mecha-

nismen, die die Datenbanken überwiegend im Memory der Hardware des Portals hinterlegen. Dadurch ist der lesende Zugriff performant gehalten, aber auch Änderungen werden, da sie im Memory geschehen, zügig umgesetzt. Der Nachteil ist, dass die Daten relativ selten auf die Festplatte geschrieben werden, was den Zugriff auf diese Daten für andere Applikationen erschwert.

Single Sign On

Je nach Portal werden unterschiedliche Erkennungsmechanismen für den Anwender implementiert. Die verbreitetste Form ist die, dass im Browser des Benutzers ein persistentes Cookie hinterlegt wird, das vom Server ausgelesen wird. Basierend auf diesen Informationen »weiß« das Portal, um welchen Anwender es sich hier handelt, es sucht sich das richtige Profil aus der Datenbank und erzeugt die personalisierte Seite.

Andere Portale verlangen die Authentifizierung über ein digitales Zertifikat oder über eine Kombination von User-ID und Passwort. Egal wie der Anwender in das Portal findet, er erwartet, dass er sich nach der Anmeldung nicht mehr weiter gegenüber dem Server ausweisen muss. Sowohl der Zugriff auf den Mailserver wie auf den Kalender oder das persönliche Adressbuch lassen sich direkt und ohne weitere Überprüfung der Identität des Benutzers nutzen. Diese Funktionalität wird als »Single Sign On« bezeichnet.

Aber weil das *http*-Protokoll nicht persistent ist, behält das Portal die Verbindungen der Anwender nicht im Gedächtnis, vielmehr sind nach der Übermittlung der Daten über *http* die Transaktionen für das Portal abgeschlossen. Cookies unterstützen bereits die Wiedererkennung des Anwenders für die erste Portalseite. Wechselt der User nun in die Mailsicht, um sich einen Überblick über seinen Mailaccount zu verschaffen, so könnten die Zugangsdaten wie User-ID und Passwort ebenfalls clientseitig in einem Cookie hinterlegt werden. Allerdings ist dies nicht unbedingt die sicherste Art der Verwaltung solch wichtiger Account-Informationen, insbesondere, wenn mit solchen Daten der Zugang auf weitere Applikationen gesteuert wird.

Alternativ kann das Portal stellvertretend für den Anwender eine Verbindung zum Mailserver erzeugen und dabei die Accounting-Daten des Benutzers verwenden, die es aus dem Directory-Server ausliest. Für diesen Zugriff benötigt das Portal User-ID und Passwort, beides lässt sich aus dem Directory auslesen. Dieser Ansatz bedingt allerdings, dass das Passwort im Directory lesbar (und nicht wie üblich gehashed) ist.

Ein dritter Ansatz ist, dass das Portal über einen Masteraccount die Verbindung zum Mailserver aufbaut und sich einen Zugang zu den Maildaten des Benutzers verschafft.

Der übliche Weg ist der, dass das Portal über einen speziellen Account im Directory Zugang auf den Mailserver erlangt und sich eine so genannte SessionID generiert, mit deren Hilfe sich der Mailserver an eine frühere erfolg-

reiche Authentifizierung »erinnert«. Diese SessionID hat eine bestimmte »Lebensdauer« und verfällt nach einer gewissen Zeit, in der über die Verbindung zum Mailserver keine Daten mehr ausgetauscht werden. Damit hat man eine Art Ticketing-System eingerichtet, das vermeidet, dass zu viele Male User-ID und Passwort ausgetauscht werden müssen. Diese SessionID wird bei jeder Transaktion mit dem Mailserver übermittelt und gestattet den Zugriff auf die Maildaten. Die Übermittlung dieser Daten kann auf drei verschiedene Wege geschehen:

- Cookies: Cookies wurden bereits in Kapitel 7.8 beschrieben. Ergänzend hierzu sei hier noch die Möglichkeit der Server-Side-Cookies erwähnt, die über eine Datenbank auf Seiten des Servers gespeichert werden.
- URL-basiert: Hier werden die Verweise auf eine Applikation mit einer Variablen versehen, die die SessionID beinhaltet. Dieses Verfahren wird als »URL-Rewriting« bezeichnet.

  Beispiel:

  ```
  http://portal.meinefirma.com/Desktopservlet?SessionID=u173UKls11
  ```

- »hidden Fields« innerhalb einer Webseite. Die Sessiondaten werden in versteckten Variablen übermittelt.

  Beispiel:

  ```
  <FORM NAME="foo" METHOD="GET" ACTION="/cgi-bin/Portal>
  <INPUT TYPE="hidden" NAME="SessionID" VALUE=" u173UKls11">
  </FORM>
  ```

Der Mailserver behält die Identitäten der gültigen Sessions bis zum Erreichen des Verfallsdatums im Memory.

Single Sign On basiert auf der clientseitigen Verwaltung der SessionIDs zu den verschiedenen Applikationen. Es speichert die »Tickets« für die Zugänge zu den einzelnen Diensten auf unterschiedliche Weise: in Cookies, in URLs oder in »hidden Fields« auf den Webseiten.

11.3 Portalversionen

11.3.1 Informations-/Service-Portale

Die erste und einfachste Form eines Portals ist eine Informationsseite. Diese wird von Content-Providern mit Nachrichten versorgt, die nach Sparten geordnet sind. So gibt es Sportkanäle, andere liefern Informationen über Politik, Kultur oder Wirtschaft. Der Anwender kann sich die für ihn interessanten Channels zusammenstellen und so positionieren, wie es für ihn am sinnvollsten erscheint. Die Parameter der Personalisierung werden in den Profildaten des Anwenders hinterlegt.

Werden diesen Informationskanälen zusätzlich Dienstleistungen wie Mail und Calendar hinzugefügt, erweitert sich diese Seite zu einem allgemeinen Service-Portal. Eine Vielzahl von verschiedenen Angeboten werden darüber hinaus mittlerweile angeboten: Adressbuch, Notiz- und Merkzettel, Fotoalben und vieles mehr. Die Daten werden nicht mehr ausschließlich über das Web verteilt, sondern auch über WAP oder PDA.

11.3.2 Enterprise Information Portale (EIP)

Im internen Netzwerk vieler Unternehmen haben sich mittlerweile Informationsportale herausgebildet, die den Mitarbeiter mit ständig aktualisierten Informationen informieren. Sie fassen die wichtigsten Kommunikations- und Nachrichtenkanäle zusammen und können vom Mitarbeiter individuell angepasst werden.

Vordefinierte Rollen legen fest, welche Art von Informationen für die jeweiligen Gruppierungen, Abteilungen und Positionen bereitgestellt werden.

Ziel ist es, dem Mitarbeiter ein Hilfsmittel an die Hand zu geben, mit dem er schnell auf alle für seinen Arbeitsablauf relevanten Daten zugreifen kann. Davon verspricht sich das Unternehmen eine Zeitersparnis und damit eine Steigerung der Produktivität.

11.3.3 Einkaufsportale, eCommerce und eProcurement

Bietet ein Unternehmen eine breite Fülle von Produkten an, so kann es ein Einkaufsportal einrichten, das für den einzelnen Anwender individuell angepasst werden kann. Bei der Anmeldung gibt dieser seine Vorlieben an und bekommt entsprechende Angebote. So werden auf einem Buchportal dem Anwender automatisch die aktuellen Angebote aus seinem Interessensgebiet angezeigt.

Solche Einkaufsportale entstehen sowohl im Business-To-Consumer- (B2C) als auch im Business-To-Business-Markt (B2B). Ziel ist es, die Interessenten mit personenbezogenen Angeboten effizient zu bewerben und damit Umsätze zu steigern. Portale dieser Art sind die Aushängeschilder der Unternehmen, die »eBusiness« im Endkundenbereich betreiben.

Eine besondere Ausprägung von Einkaufsportalen sind die elektronischen Marktplätze. Hier treffen sich im virtuellen Geschäftsfeld Anbieter und Interessenten, um sich auszutauschen, Geschäftsbeziehungen zu knüpfen und Handel zu treiben. Die Beteiligten haben auf einem solchen Portal ein individuelles Anwenderprofil mit vordefinierten Bestellabläufen, Rabatten, Lieferadressen usw. Anbieter und Interessenten können in diesem vorkonfigurierten Umfeld wesentlich effizienter Geschäfte abschließen, als dies im klassischen Umfeld möglich wäre.

Man unterscheidet zwischen horizontalen und vertikalen Marktplätzen, wie sie im Folgenden beschrieben werden.

Horizontale Marktplätze

Horizontale Marktplätze zeichnen sich dadurch aus, dass ohne Fokussierung auf eine Branche Güter und Dienstleistungen angeboten werden. Hier treffen sich viele Anbieter mit vielen Interessenten und handeln Geschäftsbeziehungen aus.

Die horizontalen Marktplätze orientieren sich im Güterbereich vornehmlich an so genannten C-Artikeln. Dabei handelt es sich um geringwertige Güter wie Kugelschreiber, Papier oder Druckerpatronen, die für das Funktionieren eines Unternehmens unerlässlich sind, aber einen geringen Anteil am Gesamtumsatz ausmachen. Gerade diese Güter stellen einen hohen Aufwand an Prozesskosten bei der Beschaffung dar (Auswahl der erforderlichen Waren, Aushandeln eines Preises usw.). Durch automatisierte Bestellabläufe werden über diese Marktplätze Kosteneinsparungen von teilweise über 50% erzielt.

Vertikale Marktplätze

Vertikale Marktplätze konzentrieren sich auf eine bestimmte Branche. Sie sind nicht für den Endkunden gedacht, sondern sind bevorzugt im Bereich Business-to-Business zu finden. So existieren beispielsweise Marktplätze für die Aluminiumbeschaffung für den Fahrzeugbau. Auf einem solchen Marktplatz werden großvolumige Einkäufe getätigt. Dabei hat der Einkäufer die Möglichkeit, über Ausschreibungen und Versteigerungen Lieferbedingungen, Qualität und Rabatte bei den Anbietern auszuhandeln, um sich anschließend für das günstigste Angebot zu entscheiden.

Portallösungen sind Grundbausteine einer funktionierenden eCommerce-Lösung.

11.3.4 Finanzportale/Electronic Bill Presentment And Payment

Finanzdienstleister haben die Möglichkeiten von Portalen entdeckt und bieten über diese Architektur in zunehmendem Maße den Kunden Konto- und Rechnungsverwaltungen an. Die Idee ist, dass der komplette Finanzhaushalt eines Kunden über das Portal gesteuert wird. Dies schließt Konten der betreibenden Bank sowie anderer Geldinstitute ein. Der Kunde hat auf »seiner« Finanz-Webseite einen Überblick über all seine Konten, Sparbücher und Kreditkartenabrechnungen.

Eine weitere Entwicklung ist die so genannte »Bill Consolidation«. Grundidee ist es, die Menge an Rechnungen, die im klassischen Umfeld immer noch per Post dem Endkunden zugesandt werden, auf elektronischem Weg an eine zentrale Stelle zu senden. Dies kann bei einer Bank oder bei einem Unternehmen realisiert sein, das sich auf dieses Geschäftsmodell spezialisiert hat. Der Kunde hat nun die Möglichkeit, auf der Portalseite dieses Unternehmens seine gesamten Rechnungen zu verwalten.

Die Weiterentwicklung einer solchen Lösung bezieht alle Konten des Anwenders ein. Auf einer solchen personalisierten Portalseite hat ein Kunde all seine Finanzen im Überblick und bekommt eine Übersicht über alle eingegangenen, bereits bezahlten oder noch offenen Rechnungen präsentiert. Über Auswahlfenster kann er entscheiden, von welchem Konto er welche Rechnung begleichen möchte.

Lösungen dieser Art werden unter dem Begriff »ePayment« oder »ebpp« (»electronic bill presentment and payment«) zusammengefasst.

11.3.5 Dienstleistungsportale

Ämter und Behörden stellen immer mehr so genannte Dienstleistungsportale zur Verfügung. Dadurch soll die Bürgernähe unterstützt und der Gang zu den Behörden erspart werden. Solche Portale bieten es dem Anwender an, entsprechend seines Wohnorts ohne langes Suchen die richtigen Ämter und Behörden zu finden. Sämtliche Angaben sind geografisch geordnet, entsprechend den Profildaten des Anwenders werden alle Dienstleistungen, regionale Nachrichten und Informationen aufgelistet. Telefonnummern und Ansprechpartner der jeweiligen Behörden werden mühelos gefunden.

Dienstleistungen dieser Art werden im Zuge der Internetdurchdringung zunehmend als Portale des »eGovernment« beschrieben.

11.4 Zusammenfassung

Portale werden in zunehmendem Maße im Internet eingesetzt, um dem Interessenten eine Personalisierung anzubieten. Das Ziel ist eine enge Kundenbindung, was durch das Anbieten von Dienstleistungen angestrebt wird, die auf die individuellen Bedürfnisse des Einzelnen angepasst sind.

12 Wireless

12.1 Entwicklung der Mobilkommunikation

12.1.1 Mobiltelefone

Die Entwicklung der Informationstechnologie hat sehr schnell den Bedarf an mobilen Endgeräten für die Benutzer geweckt. Die Kommunikation der Menschen untereinander sowie die Bereitstellung von Informationen jeder Art müssen nicht nur zu jeder Tages- und Nachtzeit erfolgen, sondern auch an jedem Ort. Dieser Anspruch wurde zunächst von den mobilen Telefonen befriedigt, so genannte »Handys«, die aus dem heutigen Leben fast nicht mehr wegzudenken sind. Innerhalb kürzester Zeit hat sich hier ein rasant wachsender Markt entwickelt, der sehr hohe Umsätze verbucht. Grund ist der jedem Menschen eigene, sehr intensive Drang nach Kommunikation, kombiniert mit dem Trend nach modischen Neuentwicklungen mit immer neueren, kleineren und funktionaleren Geräten.

Das Hauptargument für Mobiltelefone ist die direkte Kommunikation. Man ist immer und überall erreichbar und man kann schneller und flexibler auf Ereignisse und Situationen reagieren. Obwohl die physikalische Distanz fast beliebig groß sein kann, schaffen sie virtuelle Nähe zu den Gesprächspartnern.

Eine der ersten Innovationen in diesem Bereich, die Voicebox, wurde aus dem Internet übernommen. Wo man dort über E-Mail-Verkehr asynchrone Informationen austauschen kann, bietet der Mobilmarkt die Funktionalität des Anrufbeantworters. Damit ist man, wenn auch zeitverzögert, immer noch erreichbar, selbst wenn man gerade nicht online ist. Mehr noch, man kann entscheiden, ob man gerade mit einem Anrufer sprechen möchte oder nicht, ohne Gefahr zu laufen, eine wichtige Nachricht zu verpassen.

Eine weitere Innovation ist der Short Message Service (SMS). Dieser Dienst gestattet eine Kommunikation auf einem abstrakteren Level. Um eine Nachricht zu übermitteln, ist man nicht mehr auf die eigene analoge Stimme angewiesen, die neben der Nachricht auch Emotionen überträgt und die sich bei einer Tonaufnahme auf der Voicebox des Adressaten nicht korrigieren lässt. Man kann vielmehr die Nachricht verfassen, ändern, Korrekturen einfügen und erst wenn die Nachricht in der gewünschten Form verfasst ist, sendet man sie ab. Daneben hat sie, wie E-Mail, einen dokumentarischen Charakter,

weil eine kleinere Menge von SMS-Nachrichten auf dem Mobiltelefon gespeichert werden können.

Über ein Adressbuch sind die häufig frequentierten Gesprächspartner maximal vier bis fünf Tasten weit entfernt. Damit übernimmt es die erste Funktion eines Organizers. Freunde und Bekannte, Kollegen und Verwandte sind alphabetisch auf dem Gerät gespeichert und lassen sich jeweils mit einem eigenen Logo bzw. Klingelton versehen. Daraus entstand innerhalb von nur wenigen Jahren eine vollkommen neue Form der Dialogbereitschaft und der Kommunikationsform.

Schließlich entdeckten die Hersteller von Mobiltelefonen eine sehr wichtige Eigenschaft des Menschen, den Hang zur Individualität. Man möchte die Geräte personalisieren, um ihnen Einzigartigkeit zu verleihen. So lässt sich das Aussehen von Handys in Farbe und Form verändern, die Klingeltöne und das Logo des Providers können verändert und angepasst werden. Während die Form der Personalisierung bei Internetportalen eher funktioneller Art ist, stehen hier mehr emotionale Gesichtspunkte im Vordergrund: Ein »Handy« muss »cool« aussehen und möglichst individuelle Töne und Grafiken haben.

12.1.2 Laptops

Parallel zum Mobiltelefon entwickelte sich der tragbare Computer, der so genannten Laptop. Eines der ersten Geräte dieser Art war das Osborne-1, das 1981 auf den Markt kam. Es hatte die Größe eines Koffers und ein Gewicht von mehr als 10 kg. Die Entwicklung ergab sich aus der Notwendigkeit eines immer dezentraleren Arbeitsplatzes, der die Portabilität der Rechner zwangsläufig mit sich brachte. Diese Rechner sind von der Funktionalität her weitgehend mit dem stationären PCs identisch, machen aber Abstriche beim Display und der Dauer der Verwendung: Der Bildschirm ist kleiner und die Arbeitszeit richtet sich nach dem Ladevolumen der Batterie, bietet aber (bei voller Batterie oder Netzspannung) zu jeder Zeit und an jedem Ort die darin gespeicherten Daten und Funktionalitäten.

12.1.3 PDA

Anfang der neunziger Jahre entstand aus dem Bereich der mobilen Computer ein interessanter Zweig, die so genannten »Handhelds«. Diese Geräte erfreuen sich immer größerer Beliebtheit, obwohl der Umfang ihrer Funktionalität gegenüber Computern drastisch reduziert ist. Die wichtigsten Applikationen sind Adressbuch, Kalender, Notizen und Merkzettel. In jüngerer Zeit sind Internetbrowser und E-Mail-Reader hinzugekommen, ferner Dokumenten-Viewer für Adobe- und Word-Dateien und Reader für eBooks, die elektronische Variante von Büchern. Der Funktionsumfang lässt sich erweitern, Applikationen werden auf einschlägigen Seiten im Internet für den Download angeboten.

Die Vorteile sind:

- Reduktion: Die Handhelds sind vom Funktionsumfang der Applikationen her gegenüber einem PC drastisch reduziert und auf die Bedürfnisse eines mobilen Users optimiert. Dadurch sind die Anforderungen eines solchen Geräts an das Betriebssystem, Speicher und die CPU wesentlich geringer.
- Geringe Größe: Ein PDA ist so gestaltet, dass er mehr oder weniger bequem in die Westentasche passt.
- Schnelle Betriebsbereitschaft: Während ein Laptop oder PC minutenlang bootet, ist ein Handheld wegen des relativ kleinen Betriebssystems nach bereits maximal drei bis vier Sekunden betriebsbereit.
- Lange Betriebsdauer: Die Geräte schalten nach einer gewissen Dauer der Inaktivität automatisch ab. Dadurch sind die Anforderungen an die Batterie wesentlich geringer, was ein leichteres Gerät mit sich bringt.
- Einfache Bedienung: Die meisten PDAs werden über einen Touch Screen und einen Stift bedient. Damit macht man (bei tastaturlosen Geräten) den Nachteil einer fehlenden direkten Dateneingabe wett.

Das erste Gerät dieser Art erschien 1992 auf dem Markt, der »Newton« von Apple. Das Gerät war seiner Zeit weit voraus, hatte allerdings Probleme bei der Handschrifterkennung und war langsam.

Aus heutiger Sicht werden sich die beiden Strömungen der Mobiltelefone und Handhelds mittelfristig in einem einzigen Gerät vereinen. Spätestens dann hat man einen universelles Gerät, das alle wichtigen und persönlichen Daten bereithält, Verbindungen zu anderen Geräten und Menschen aufbaut und das den Anwender an wichtige Termine erinnert. Dieser mobile Helfer wird vielleicht nicht alle Funktionalitäten haben, die man eventuell benötigt, es wird aber die Möglichkeit bieten, sich mit einem anderen Dienst zu verbinden, der diese Funktion besitzt. Mit diesen Geräten wird man somit seinen eigenen persönlichen Wirkungskreis um ein Vielfaches erweitern können und hätte zudem die ständige Datenbereitschaft direkt vor Ort.

Dem Weg dieser Geräte ins Internet stehen einige Hindernisse im Weg. Der Wireless-Markt ist noch recht neu und die Techniken, die eine optimale Übermittlung von Daten auf mobile Endgeräte unterstützen, werden erst entwickelt. Aufgrund der Einschränkungen in der Darstellung, Bedienung und Datenübertragung müssen neue Wege beschritten werden, um Endgeräte dieser Art internettauglich zu machen.

- Datenübertragung

  Bei der mobilen Kommunikation sind die verwendeten Internetstandards wie *html*, *http*, SSL nicht brauchbar, weil das Internet für die Festnetzanbindung geschaffen wurde, die auf einer Datenübertragung in Textform basiert. Als in späteren Jahren multimediale Daten Einzug hielten, wurden die Übertragungsraten ausgebaut, um die wachsende Da-

tenflut zu übermitteln. Im Wireless-Umfeld kommt jedoch der Ausbau der Bandbreiten sehr viel langsamer voran, so dass die im Internet angebotenen Webseiten ein viel zu großes Datenvolumen haben, um noch mit sinnvollem und bezahlbarem Aufwand übertragen zu werden.

- Darstellung

  Selbst wenn die Datenübertragung von Webseiten einigermaßen effizient wäre, könnten sie kaum sinnvoll dargestellt werden. Dies liegt an der Größe, Auflösung und Farbtiefe heutiger Displays. Die meisten Mobiltelefone könnten gerade ein bis zwei Prozent einer kleinen Internetseite (800x600 Pixel) direkt darstellen. Die wenigsten haben ein Farbdisplay, sie unterstützen lediglich Schwarzweißgrafiken. Bei den PDA/Handhelds sind Geräte mit Farbtiefen von bis zu 256 auf dem Markt, die Auflösung ist etwas größer als beim *wap*-Telefon, die meisten kommen aber über 15% der Darstellung einer Webpräsentation nicht hinaus.

  Aber selbst wenn die volle Auflösung und Farbtiefe eines Computermonitors erreicht würde, wäre es schwierig, auf einem wenige Quadratzentimeter großen Fenster sinnvoll Internetseiten lesen zu können. Eine Vergrößerung des Displays ist schwierig, da das Gerät immer noch in die Westentasche passen soll.

  Ein weiteres Handicap ist die Benutzerführung. Internetseiten sind darauf ausgerichtet, dass man über Mausbewegungen und -klicks die entsprechenden Aktionen auslöst. Bei einem Mobiltelefon ist weder eine bequeme Tastatur zu erwarten noch eine der Maus vergleichbare Eingabemöglichkeit.

- Funktionsumfang

  Nicht unerheblich ist die Tatsache, dass kleine tragbare Informationsgeräte andere Hardware-Voraussetzungen als stationäre Desktop-PCs haben. Typischerweise unterliegt man hier Beschränkungen von mehr als 90% der üblichen PC-Ausstattungen. Dadurch ist es fast unmöglich, Software zu entwickeln, die den Browserumfang von PC-Betriebssystemen bereitstellen.

Sowohl für wireline- als auch für wireless-User ist das Internet das einzig relevante Datennetz für Informationen, das heißt, es wird kein spezielles *wap*- oder PDA-Netz aufgebaut werden. Eine der wesentlichen Herausforderung ist die Anpassung bzw. Erweiterung des bestehenden Internets an die Anforderungen einer drahtlosen Kommunikation. In einem Festnetz werden die Daten über TCP/IP übertragen, ein Protokoll, das aus der Taufe gehoben wurde, als man noch nicht die drahtlose Datenübertragung in Betracht zog. Entsprechend wenig sind diese Protokolle auf die gegenwärtigen Funkübertragungswege vorbereitet. Wireless-Geräte tauschen die Daten über Funkwellen im Wellenlängenbereich von Radiowellen aus. Diese Übertragung kann durch Interferenzen von anderen Geräten gestört werden, Bäume, Gebäude oder Gebirge können »im Weg« stehen und die Kommunikation verhindern, selbst die Sonneneinstrahlung kann zu Beeinträchtigungen im

Datenaustausch führen. Ein weiteres Problem ist die geringe Bandbreite der Übertragungskanäle für alle Mobiluser. Die Übertragungsfrequenzen müssen bei längeren Kommunikationspausen anderen Benutzern zur Verfügung gestellt werden, damit eine einigermaßen sinnvolle Verwendung der ohnehin begrenzten Übertragungsbandbreite gewährleistet ist. Protokolle wie TCP/IP und *http*, die die Daten unkomprimiert und im Klartext über das Netz austauschen, benötigen eine weit höhere Bandbreite, so dass man für die Übertragung der Daten neue Entwicklungen in Richtung komprimierten Datenaustauschs beschreiten muss.

Ein weiteres Hindernis ist die bestehende Form der Daten im Internet. Diese sind vorwiegend in *html* geschrieben, die Daten werden über *http* über das Netz verteilt. Beide Techniken sind für große Displays und Bandbreiten für die Datenübertragung optimiert und in keiner Weise für die gegenwärtigen mobilen Endgeräte geeignet. Da man das Internet nicht neu gestalten kann, müssen die zu übertragenden Daten übersetzt werden. Dies machen so genannte Gateways, die je nach Technologie die Daten auf die notwendigsten Informationen und/oder in eine komprimierte Form übersetzen, um die zur Verfügung stehende Bandbreite für die Datenübertragung optimal zu nutzen.

Auf der anderen Seite fragt man sich, ob die volle Internetbereitschaft solcher Geräte nötig ist. Der durchschnittliche mobile User hat ein komplett anderes Bedürfnisprofil, als es ein Anwender hat, der am heimischen PC sitzt und von dort aus das Internet durchforstet. Er möchte unterwegs telefonieren können und mittels kurzer Nachrichten informiert werden. Er möchte Börsendaten erhalten und eine sichere Verbindung zu seiner Bank aufbauen können, um Kontostände oder Rechnungseingänge zu überprüfen. Lokalisierte Informationen wie beispielsweise Restauranttipps oder Parkplatz-Finder helfen ihm, sich in einer fremden Umgebung zurechtzufinden. Dagegen ist das Bedürfnis, im Internet zu surfen, relativ gering. Daraus ergibt sich, dass E-Mails und SMS, Adressbücher und Kalender, Nachrichten und Finanzinformationen weit wichtiger sind, als hochgestylte Webseiten mit aufwändigem grafischen Aufbau.

Um diese Daten zu liefern, werden immer mehr so genannte Wireless-Portale installiert. Diese stellen einen grafischen Desktop für einen Internetbrowser bereit, übernehmen aber auch die Aufgabe, die vom Anwender geforderten Daten zu sammeln und sie auf die Wireless-Geräte zu übermitteln.

12.2 Wireless-Portale

In Kapitel 11 wurden Internetportale beschrieben. Mit einer solchen Webseite kann sich der Benutzer einen persönlichen Bereich im Internet schaffen, den er als Ausgangspunkt für alle weiteren Netzaktivitäten verwendet.

Portale dieser Art haben ein grafisches Frontend – das die *html*-Version für den Browser erzeugt – und ein Backend, in dem die Daten und Dienste hinterlegt sind.

Die meisten Wireless-Portale bieten ebenfalls einen *html*-Desktop für den Festnetzzugriff, sie öffnen aber die gleichen Backend-Daten und -Dienste auch für die mobilen Geräte. Dadurch greift der User über sein Mobilgerät auf die gleichen Mails, Adressen, Nachrichten und Dienste zu, wie aus der Festnetzverbindung.

Der Vorteil einer solchen Architektur ist, dass der Benutzer nicht mehr von dem verwendeten Gerät abhängig ist. Er kann an einem öffentlichen Internetterminal genauso seine Mail lesen wie mit jedem beliebigen Mobiltelefon. Seine E-Mail-Adresse, die er auf seinem PC konfiguriert hat, ist die gleiche, die er beim Verfassen einer Nachricht über seinen PDA verwendet. Die Börsendaten aller für ihn wichtigen Unternehmen werden einmal konfiguriert und sind von überall her erreichbar.

Die ersten Portale, die im Internet aufgebaut wurden, stellten die Architektur des User-Interfaces weitgehend in den Vordergrund und die Informationen und Services wurden als »Zulieferer« der Daten verstanden.

Heutige Portale entwickeln sich immer mehr zu einer umfangreichen und ausfallsicheren Backend-Infrastruktur, die über eine geeignete Präsentationslogik die verschiedenen Darstellungsgeräte unterstützt. Eine solche Architektur gibt dem Betreiber die Möglichkeit, zukünftige Endgeräte zu berücksichtigen, ohne den Portalaufbau grundlegend ändern zu müssen.

Die unterschiedlichen Bedienungselemente der jeweiligen Endgeräte erfordern eine individuelle Anpassung der bereitstellenden Dienste:

- Die Anwender im Festnetz verwenden für die Darstellung von Web-Content einen Web-Browser. Die Bedienung erfolgt über Tastatur- und Mauseingaben. Die Darstellung der Informationen geschieht auf hochauflösenden Monitoren mit hoher Farbtiefe. Trotzdem müssen die Eigenheiten der verschiedenen Browser berücksichtigt werden. So unterscheiden sich Internet Explorer, Netscape Navigator, Opera und viele anderen Browser in der Interpretation der *html*-, Java- und JavaScript-Syntax. Gleiche gilt für die verschiedenen Betriebssysteme. So reagieren die Browser unter Linux, Windows oder Macintosh unterschiedlich, der Aufwand für diese Anpassungen ist aber in der Regel überschaubar.
- Der Markt der Handhelds bietet eine breite Palette verschiedener Betriebssysteme und Funktionalitäten. Die Bedienung erfolgt über einen Stift, der auf einem Touchscreen von der Größe einer Handfläche verwendet wird. Die Verbindung ins Internet erfolgt entweder über eine Docking-Station, die am PC angeschlossen ist, oder über die Infrarot-Schnittstelle zum Mobiltelefon. Die Größe der Displays, die Auflösung, Farbunterstützung und -tiefe sind vollkommen unterschiedlich. Diese Vielfalt macht es für die Anbieter von Wireless-Portalen schwer, einen

überschaubaren Aufwand für die Unterstützung dieser Geräte zu finden. Während einige Geräte umfangreiche Browser-Unterstützung enthalten, bei der selbst Cookies und JavaScript weitgehend implementiert sind, bieten andere Geräte nicht einmal eine volle *html*-Implementierung. Die Marktdurchdringung der verschiedenen Geräte bestimmt die Entscheidung für die Unterstützung durch das Portal. Damit haben die Entwickler dieser Applikationen die Aufgabe, entweder einen kleinsten gemeinsamen Nenner für alle Geräte zu finden oder über eine Geräteerkennung beim Aufbau der Verbindung zum Portal individuell angepasste Seiten bereitzustellen.

- Smart-Phones verbinden die Möglichkeiten eines Mobiltelefons mit den Vorteilen eines PDA. Sie werden entweder über Stifte auf einem Touch Screen bedient oder bieten eine vollwertige Tastatur an, andere haben eine Kombination aus beidem. Für die Datenübertragung wird je nach Gerätetyp sowohl *http* als auch *wap* verwendet. Die Internetverbindung geschieht direkt über das interne Modem. Bei diesen Geräten ist es am schwierigsten, eine umfassende Unterstützung anzubieten. Die Bedienung und die grafische Darstellung ist neben dem Funktionsumfang, Protokollunterstützung und *html*/wml-Support vollkommen unterschiedlich. Daraus ergibt sich für diese Kategorie der größte Implementierungsaufwand für einen Portalbetreiber bei der Unterstützung der marktüblichen Geräte.
- *wap*-Telefone bieten den geringsten Darstellungs- und Funktionsumfang. Da sich aber der *wap*-Standard weitgehend etabliert hat, ist diese Unterstützung mit mehr oder weniger geringen Unterschieden gleich. Eine Präsentation der Daten ist weitgehend frei von grafischen Hilfsmitteln, man beschränkt sich auf Texte und Links.

Für jede Klasse der vorgestellten Endgeräte muss eine angepasste Applikation die Backend-Daten aufbereiten und über die entsprechenden Protokolle übertragen. Auch wenn die *html*-Oberfläche nach wie vor einen wichtigen Stellenwert einnimmt und für den Anwender die erste Anlaufstelle für seinen Informationsbedarf ist, erhalten Zugriffe über PDA und *wap* auf die gleichen Daten einen immer höheren Stellenwert.

12.3 WAP-Architektur

Im Folgenden werden die Komponenten des »Wireless Application Environment (WAE)« beschrieben, die die Gesamtheit der *wap*-Technologie ausmachen. Hierzu gehört die Implementierungssprache WML und WMLScript, das Übertragungsprotokoll *wap* und die WAE User Agents, die mobilen Endgeräte.

12.3.1 Einführung

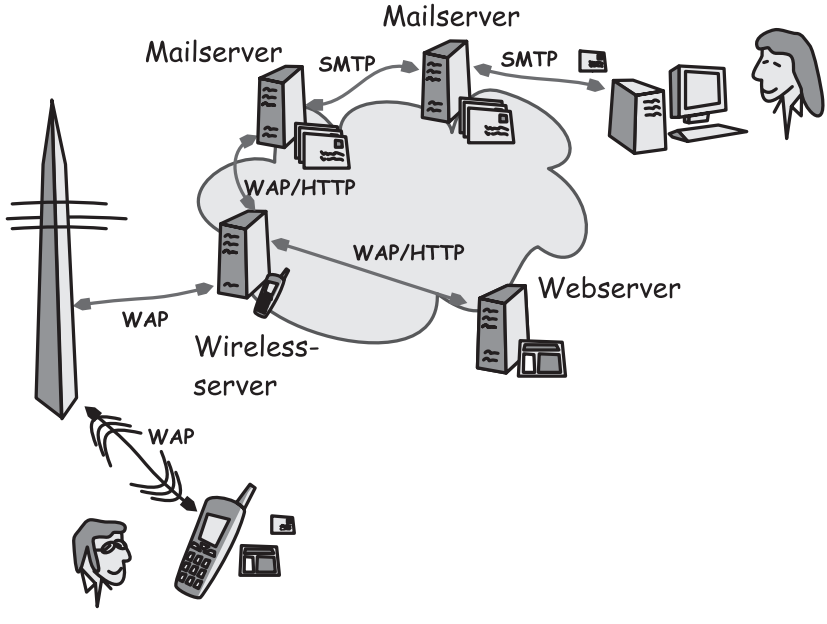
wap ist ein Protokoll, mit dem man über drahtlose Kommunikation mittels eines »Handheld Wireless Device« Informationen aus dem Internet übertragen kann. Diese Geräte sind für die mobile Kommunikation entwickelt und reichen vom einfachen Mobiltelefon bis zum Mini-Computer für die Westentasche.

Das Protokoll ist das Äquivalent zum *http*-Protokoll im wireline-basierten Netzwerk. Die Übertragung der Daten geschieht in komprimierter binärer Form. Darüber hinaus ist der Protokoll-Overhead auf ein paar Byte pro Request reduziert, um die Übermittlung der Daten performant zu gestalten.

Die Daten werden von der Quelle über das wireline-Netzwerk zum *wap*-Gateway übertragen und von dort über Funkwellen an den Endbenutzer verschickt. Das Gateway übersetzt die Verbindung des *http*-Protokolls in Richtung des Endanwenders in *wap*, um mit den wireline-Services zu kommunizieren.

Obwohl *wap html* und XML unterstützt, werden Informationsseiten für dieses Protokoll in der »Wireless Markup Language« WML [42] geschrieben. Dabei handelt es sich um eine Sprache, die aus XML abgeleitet wurde und für die Darstellung auf kleinen Displays optimiert ist.

Abbildung 12.1: Wireless-Topologie



Reine WML-Seiten sind mehr oder weniger statisch. Um zusätzliche Funktionalität einzubringen, steht WML-Script zur Verfügung. Hierbei handelt es sich um eine simple Programmiersprache, die auf ECMAScript (dem früheren JavaScript) basiert.

12.3.2 Die Implementierungssprache

Mittels der Wireless Markup Language (WML) sollen Daten auf kleinen und eingeschränkten Displays dargestellt werden. Die Daten werden in einer Art »Kartenstapel« an die Endgeräte übermittelt. Eine Webseite, die in WML übersetzt wird, wird in mehrere Kartenstapel verteilt, sie enthält Texte, Hyperlinks und kleine Grafiken. Eine komplette Webseite passt in einen oder mehrere dieser Kartensammlungen, je nachdem, wie viele Daten das jeweilige Endgerät in einem solchen Stapel verarbeiten kann.

Auf einem Endgeräte-Display wird ein kompletter Stapel geladen, auf dem Display wird allerdings nur eine Karte dargestellt. Der Benutzer kann dadurch rasch von einer Karte zur nächsten wechseln, bis der Stapel abgearbeitet ist. Erst dann wird die Verbindung zur *wap*-Seite wieder aufgebaut und der nächste Stapel geladen.

Der dynamische Inhalt einer *wap*-Seite, der über WMLScript [42] erzeugt wurde, wird nicht wie bei einem Webclient interpretiert, sondern entweder vom *wap*-Server oder vom *wap*-Gateway kompiliert, an das Endgerät übermittelt und dort unmittelbar ausgeführt. Ähnlich wie bei Java wird der Bytecode vom *wap*-Client in einer Virtual Machine interpretiert.

12.3.3 Weitere Datenunterstützungen

Im *wap*-Umfeld werden über die WML-basierten Internetseiten hinaus zusätzlich standardisierte Kalender- und Adressbucheinträge unterstützt. Diese Datenformate sind weitläufig akzeptierte Standardformate, deren Weiterentwicklung vom Internet Mail Consortium (IMC) vorangetrieben wird.

Kalenderformat vCalendar

Dieses Format basiert auf RFC 2445 [38]. Es handelt sich hierbei um ein Textformat auf Basis von Attribute/Value-Paaren, die einem eigenen Schema folgen (siehe Kapitel 9.4.1). Weitere Informationen zur Implementierung und zur Beschreibung des Standards sind unter [36] zu finden.

Adressbuchformat vCard

Das Datenformat vCard wird in [44] beschrieben und definiert eine einfache Syntax zum Formatieren von Benutzerdaten im digitalen Visitenkartenformat.

12.3.4 Der Protokoll-Stack

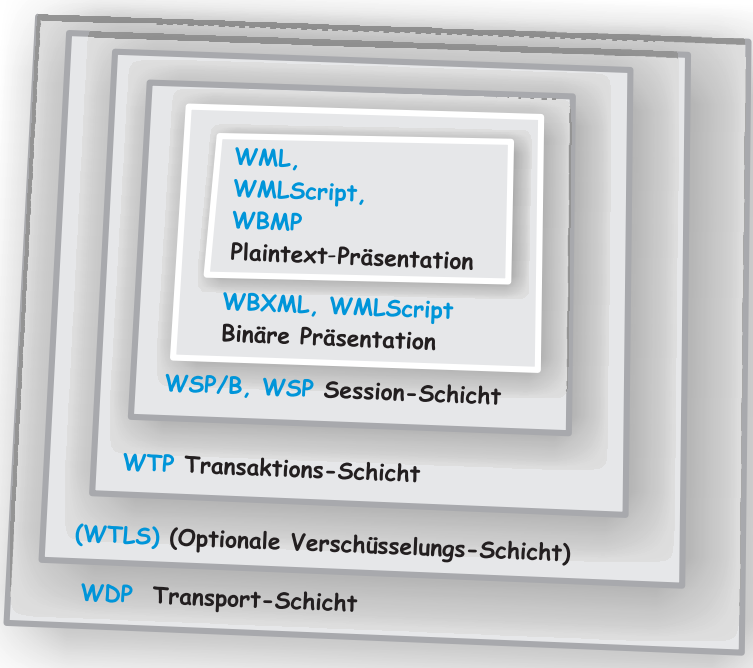
Die Informationsübertragung ist bei *wap* wie in anderen netzwerkbasierten Diensten über verschiedene Stapel, so genannte »Stacks« verteilt.

WML, WMLScript, WBMP

Obwohl die Präsentationsschicht nicht während der Datenübertragung, sondern davor bzw. danach eine Rolle spielt, ist sie im Protokoll als eigene Schicht ausgewiesen.

Die Präsentationsebene untergliedert sich in zwei Bereiche, der reinen Klartextimplementierung und der binär-komprimierten Ebene.

Abbildung 12.2: Transport- und Präsentationsschichten



Die oberste Klartextimplementierung bildet zusammen mit den Steuer-Tags die Display- bzw. Implementierungsversion der Daten. Diese sind in WML implementiert, weitere Funktionalität wird durch WMLScript umgesetzt. Das dritte Präsentationswerkzeug ist WBMP zur Darstellung von einfachen Grafiken.

WBXML, WMLScriptc

Eine Ebene unter der eigentlichen Präsentationsschicht befindet sich die komprimierte Datenversion, die sowohl server- als auch clientseitig in bzw. aus der Klartextschicht hervorgeht. Mit WBXML (für WML) und WMLScriptc (für WML-Script) liegen die Daten in Binärform vor. Grafiken liegen bereits in komprimierter Form vor und brauchen nicht nachträglich umgewandelt zu werden.

wsp/b, wsp

Die dritte Schicht im *wap* ist die oberste Transportschicht. Hier sind die beiden Protokolle *wsp/b* und *wsp* vertreten: Das *wap*-Session-Protokoll */b* (*wsp/b*) ist dem *http* nachempfunden und ist ein zustandsloses binäres Protokoll. Wie im World Wide Web besteht die Kommunikation aus Request/Response-Übertragungen, die ausgetauschten Informationen beinhalten serverseitig neben den reinen Daten Attribute wie Ursprung, Art und Größe der Daten, clientseitig Informationen wie Art des Browsers und Version des Protokolls, das er versteht.

Wie im *http*-Protokoll werden keine Verbindungs- und Sessiondaten ausgetauscht, ein Request wird von einem Client gestellt, vom Server beantwortet und die Verbindung wird abgebaut, das Protokoll ist nicht persistent.

Das *wap*-Session-Protokoll *wsp* ist im Gegensatz zum *wsp/b* verbindungsorientiert. Das heißt, die Session wird erst nach Beenden der gesamten Kommunikation abgebaut. Genau genommen ist *wsp* eine Erweiterung des *wsp/b*, in dem zur Session-Verwaltung weitere Befehle hinzugefügt wurden. Es werden die gleichen Informationen wie bei *wsp/b* ausgetauscht, also Art, Herkunft und Größe der Daten. Weitere Felder verwalten die Session-Informationen der Verbindung.

Die Session wird von einem Client über *wsp* initiiert und wird entweder vom Server oder vom Client nach Austausch der Daten über mehrere Request/Response-Aktionen explizit beendet.

Weitere Details der *wsp*-Spezifikation finden sich unter [40].

wtp

Das *wap*-Transaction-Protokoll steuert den Datenaustausch. Es gibt drei unterschiedliche Klassen von Transaktionen:

- Class 0: Hier kann man fast nicht von einer Transaktion reden, es handelt sich um einen simplen Push von Daten ohne Rückmeldung oder Bestätigung.
- Class 1: Diese Transaktion beinhaltet sowohl den Push von Daten als auch eine Bestätigung, dass die Daten übermittelt wurden (handshake).
- Class 2: Bei dieser Transaktion werden Daten gesendet, bestätigt und gegenbestätigt.

Die meisten *wap*-Transaktionen sind von der Art her Class 2 und 3. Das Protokoll unterstützt die Segmentierung von Daten, deren Übertragung und das nachträgliche Requesten von verlorenen Paketen. Darüber hinaus werden verlorenen Paketen Timeout-Zeiten gegeben, die nach Ablauf einer Frist neu angefordert werden.

WTLS

Der *wap* Transaction Layer Security dient der Sicherheit der übertragenen Daten und stellt einen Verschlüsselungskanal bereit, der dem von SSL bzw. TLS im Internet vergleichbar ist. Wie bei den Internetvarianten ist diese Krypto-Schicht optional, sie kann für die sichere Verbindung verwendet werden, ist aber nicht dafür notwendig.

Es existieren drei verschiedene Krypto-Protokolle:

- Asynchrone Verschlüsselung ohne beiderseitige Authentifizierungsmechanismen (Client/Server)
- Serverseitige Authentifizierung über ein geeignetes digitales Zertifikat
- Clientseitige Authentifizierung über ein geeignetes digitales Zertifikat

Es werden drei verschiedene Typen von digitalen Zertifikaten unterstützt:

- X.509-Zertifikate, der übliche Internetstandard zur Authentifizierung
- WTLS-eigenes Zertifikat, das eine gesonderte Form eines digitalen Zertifikats darstellt und auf der Optimierung des Datenaustauschs beruht, indem nur die notwendigsten Daten für die Authentifizierung übermittelt werden
- X.968, das derzeit noch in der Spezifizierungsphase ist, wird noch nicht zu 100% von allen Geräten unterstützt

wdp

Der Data Transport Layer ist dem internetbasierten User Datagram Protocol (*udp*) nachempfunden, einem einfachen datagrammorientierten Protokoll zur Datenübermittlung auf Netzwerkebene. *wdp* übernimmt nicht die Aufgabe der Übertragungskontrolle, das heißt, es werden keine Pakete quittiert, verlorene Daten werden nicht neu angefordert und Fehler in der Übertragung werden nicht korrigiert. Dies ist Aufgabe der übergeordneten Protokolle.

Weitere Informationen zur Spezifikation finden sich in [41].

12.3.5 Architektur des Kommunikationsprozesses

Webseiten, die in WML, WMLScript und WBMP geschrieben wurden, werden auf Webservern im Internet hinterlegt. Ein mobiles Endgerät sendet einen Request auf eine Internetadresse an das *wap*-Gateway. Es baut eine Session über *wsp* mit dem Gateway auf, handelt Übertragungsdetails aus und übermittelt den Request auf die Internetadresse. Das Gateway empfängt diese Aufforderung über das binäre *wap*-Protokoll, übersetzt sie in das textbasierte *http*-Protokoll und übersendet sie an den Webserver. Die Beantwortung des Request geschieht in umgekehrter Reihenfolge, das Gateway übersetzt auch hier den textbasierten Inhalt des Webservers in binäres *wap*-

Format und übermittelt die Daten an das Endgerät. Das bedeutet, dass jeder übliche Webserver als Content-Plattform für *wap*-Inhalte verwendet werden kann, und es wird kein grundlegend neuer Service implementiert.

Leistungsfähigkeit der Endgeräte

Bei der Einführung der *http*-basierten Internetdienste ging man von hinreichend großen und leistungsfähigen Clients aus. In dem Maße, in dem die Hardware-Entwicklung voranschritt, wurden die Internetseiten aufwändiger und größer und obwohl man das eine oder andere Mal bezüglich des Designs der Webseiten Abstriche machen musste, war die Darstellbarkeit der Seiten selten ein unlösbares Problem.

Mit dem Aufkommen des mobilen Browsers sieht man sich aber plötzlich mit Einschränkungen in der Handhabung und in der Darstellbarkeit konfrontiert. Die aufwändigen Seiten sind in kleinen, meist einfarbigen Displays der CPU-armen Endgeräte nicht darstellbar und es bedarf einer Übersetzung der Seiten in ein Format, das die wesentlichen Informationen aus den Internetseiten extrahiert und sie dem Low-End-Gerät übersendet. Nun hat aber jeder Hersteller dieser User Agents eine andere Vorstellung von Displaygröße, Farbtiefe und Grafikauflösung, so dass es keine einheitliche Umwandlung dieser Seiten gibt, sondern diese individuell für jedes Endgerät neu definiert werden müssen.

Beim Aufbau einer Verbindung wird zwischen dem Server bzw. dem Gateway und dem Client ausgehandelt, welche Daten mit welchen Character-Sets und Sprach-Settings für die Darstellung auf dem User Agent geeignet sind. Dies wird im *wsp*-Layer verhandelt. Sobald der Server die Charakteristik des Endgeräts kennt, kann er die ursprünglichen Daten vom Server modifizieren und sie dem User Agent zur Verfügung stellen. Der Austausch der Charakteristiken des Endgeräts geschieht über vordefinierte Profile, die auf Seiten des Gateways oder eines Servers hinterlegt sind. Diese Informationen werden als »Capability and Preference Information (CPI)« bezeichnet. Ein solches Profil wird in der Verhandlungsphase im *wsp*-Layer beim Verbindungsaufbau über das Attribut »uaprof« im Accept-Header benannt und für die Client-Charakteristik hergenommen. Bei der Verbindung wird üblicherweise die Profilinformation für die Dauer der Verbindung vom *wap*-Gateway gecached. Ein solches Profil ist in XML realisiert und definiert sich aus einem Schema, das von dem W3-Konsortium erarbeitet wurde. Ähnlich wie beim Directory-Schema ist bei der Implementierung des Profils ein Vokabular und eine Syntax vorgegeben.

Datenübertragung von formatierten Kalender- und Adressbucheinträgen

Wie schon beschrieben, werden die Standards vCalendar und vCard von *wap* nativ unterstützt. Für die Übertragung der Daten stehen bei *wap* zwei Methoden zur Verfügung:

1. *wdp*-Datenpaket-Übertragung

   Für die Übertragung von Kalender- oder Adressbuchdaten sind bestimmte Ports vorgesehen, die vom untersten Layer des Protokolls, der Transportschicht *wdp* direkt angesteuert werden. Daten, die dort ankommen, werden von der entsprechenden Applikation entgegengenommen und richtig interpretiert. Es gilt:

   - Port 9204: vCard-Datenpakete
   - Port 9205: vCalendar-Datenpakete
   - Port 9206: vCard-Datenpakete verschlüsselt
   - Port 9207: vCalendar-Datenpakete verschlüsselt

2. *wsp*-Datenaustausch

   Hier werden die Daten als MIME-Types über das Session-Protokoll *wsp* übertragen. Die Spezifikation sieht vor, dass

   - vCards den MIME-Type »text/x-vCard« und die Extension ».vcf«,
   - vCalendar den MIME -Type »text/x-vCalendar« und die Extension ».vcs«

   haben.

12.4 Zusammenfassung

Die Entwicklung im Wireless-Markt macht rasante Fortschritte. Es herrscht keine Klarheit darüber, wie die Endgeräte und ihre Handhabung aussehen werden, man erkennt aber den Wunsch der Kunden nach dieser Art von Geräten. Es ist abzusehen, dass PDA und Mobiltelefon zu einem Gerät zusammenwachsen werden.

Vor der erfolgreichen Markteinführung eines solchen Produkts müssen noch einige Probleme bewältigt werden. So ist z.B. die Datenübertragungsrate noch zu gering, um damit Videos zu übertragen. Die Display-Größen, deren Auflösungen und Farbtiefen sind noch nicht befriedigend. Die Ergonomie bei der Handhabung lässt noch zu wünschen übrig.

13 Internetadressierung über URL

13.1 Struktur einer URL

Um Daten oder Objekte im Internet zu finden, benötigt man Verweise, die sowohl den Namen des jeweiligen Objekts als auch den Ort und die zu verwendende Applikation für die Darstellung benennen. Die am häufigste verwendeten Verweise folgen der *http*-basierten Adressierung:

```
http://www.meinefirma.com/Marketing/index.html
```

In diesem Ausdruck wird das Übertragungsprotokoll *http* benannt, weiterhin der Name des Rechners (www) und die Domain (meinefirma.com), in der sich dieser Rechner befindet. Weiterhin ist die Spezifizierung eines Directory auf dem Filesystem des Webservers und der Name der angeforderten Datei selbst angegeben. Diese Form der Referenzierung wurde für die Übertragung von *html*-Seiten eingeführt und hat sich mittlerweile auf Übertragungen mit anderen Protokollen erweitert. Der allgemeine Oberbegriff dieser Verweise ist »Uniform Resource Locator« (URL) und wurde von Tim Berners-Lee [3] erstmalig in dieser Form dargestellt.

Eine URL hat eine standardisierte Struktur der folgenden Form:

```
Schema://(/)Unified Resource Identifier
```

Das Schema legt über das Applikationsprotokoll den Übertragungsweg fest. Es werden eine Reihe von Protokollen unterstützt, die gebräuchlichsten sind:

http	Hypertext Transfer Protocol
ftp	File Transfer Protocol
gopher	Gopher-Protokoll
ldap	Directory-Acces-Protokoll
news *(nntp)*	Usenet News

(Manche Protokolle werden zusätzlich über einen Krypto-Layer, beispielsweise *http* über *ssl* hin zu *https*, zusammengefasst).

Andere Zugriffsschemen sind:

mailto	E-Mail-Adresse
file	Zugriff auf ein lokales File

Die Applikationsprotokolle werden von der URI über »://« bzw. »:///« getrennt. Hier sieht der Adressierungsstandard Raum für Erweiterungen vor. So markiert die Verwendung des doppelten Slash (»/«) die Spezifikation eines Rechners, während der dreifache Slash auf ein lokales File zeigt. So sind folgende Schreibweisen äquivalent:

```
file://localhost/Verzeichnis/Datei
```

und

```
file:///Verzeichnis/Datei
```

Die Spezifikation des Ziels über die URI gliedert sich in die Teile Rechneradressierung und Datenselektion. Für die Rechneradressierung gilt:

```
schema://()<Host>.<Domain>:<Port>/<Datenselektion>
```

Bei manchen Protokollen kann man hier auch noch Authentifizierungsinformationen übergeben:

```
schema://()<userID>:<password>@<Host>.<Domain>:
<Port>/<Datenselektion>
```

Damit ist folgendes Beispiel eine gültige URL:

```
http://juser:iamtheboss@www.meinefirma.com
```

Die Angabe des Rechners innerhalb einer Domain ist die DNS-kompatible Darstellung. Die Namensauflösung eines Servers wird vom DNS übernommen. Mit der Angabe einer URL parst der Browser die URL und isoliert den Hostname, dessen IP-Auflösung er dem lokalen DNS übergibt. Hier ist es auch möglich, direkt eine IP-Adresse zu benennen und den Name Service zu übergehen:

```
http://123.45.67.8
```

Die Angabe des Ports ist im Allgemeinen nur dann notwendig, wenn der adressierte Service auf einem nicht standardisierten Serverport läuft:

```
http://www.meinefirma.com:89
```

Die Darstellung der Datenselektion hängt von der Art des Datenzugriffs ab. So ist der Zugriff über *http* vollkommen unterschiedlich von dem über *ldap*.

13.1.1 http

Die Syntax der Datenselektion sieht vor, dass zunächst bei vollkommener Abwesenheit ein so genanntes Indexfile vom Webserver angefordert wird:

»http://www.meinefirma.com« veranlasst den Webserver, ein so genanntes Indexfile auszuliefern. Diese *html*-Files haben in der Regel Namen wie »index.html« oder »home.html«. Diese Form des Zugriffs unterstützt insbesondere den (nicht verlinkten) Direktzugriff auf eine Webseite, die von einem User angefordert wird. Dieser kann nicht wissen, wie die Startseite

des Webservers heißt, und erwartet, dass der Server ihm »irgendetwas« zurückliefert.

Die Datenselektion für *html*-Seiten baut sich über die Angabe eines Verzeichnisses und eines Files auf:

```
http://<Authentifizierung><host><domain>:<Port>/
<Verzeichnis>/<Datei>
```

Das Verzeichnis muss nicht unbedingt mit einem wirklichen Verzeichnis auf dem Filesystem des Webservers übereinstimmen, vielmehr kann es sich auch um ein virtuelles Mapping auf ein Ziel in einem anderen Verzeichnis, einer Partition oder sogar eines Servers handeln.

In Verbindung mit der Angabe des Dateinamens können weitere Spezifikationen der gewünschten Daten gegeben werden. Insbesondere die Marke (#) zielt auf eine bestimmte Stelle innerhalb einer Webseite:

```
http://.../<Verzeichnis>/<Datei>#Marke
```

Beispiel:

```
http://www.meinefirma.com/Marketing/
Praeso2.html#Absatz3
```

bedeutet einen direkten Sprung auf eine bestimmte Stelle in einem *html*-Dokument.

Bei dynamischen Seiten werden keine Dateien referenziert, sondern Servlets oder Scripte. Diesen Funktionen können Daten über die URL übergeben werden:

```
http://.../<Verzeichnis>/
<Funktion>?<Attribut>=<Wert>(&<Attribut>=<Wert>)...
```

Beispiel:

```
http://www.meinefirma.com/cgi-bin/
Formular?name=Joe&tel=4711
```

13.1.2 ftp

Eine ftp-URL ist ein Zeiger auf ein Ziel innerhalb eines ftp-Servers. Der Login kann wie bereits beschrieben in der URL untergebracht werden. Der Unterschied zwischen einem Zugriff auf eine Datei über einen URL gegenüber einer ftp-Session ist die einmalige Übertragung einer Datei mit dem anschließenden Abbau der Verbindung.

Bei einem URL-basierten Zugriff auf eine Datei, die auf einem ftp-Server hinterlegt ist, wird vom Server der übergebene Request geparst und gemäß den Vorgaben des ftp-Protokolls abgearbeitet.

Beispiel:

Bei einer URL der Form

```
ftp://ftp.meinefirma.com/shared/bin/browsers/communicator.exe
```

wird zunächst eine Verbindung zum ftp-Server aufgebaut. Im vorliegenden Fall erfolgt der Verbindungsaufbau anonym, für den Fall einer notwendigen Authentifizierung folgt entweder eine Eingabemaske zur Angabe von User-ID und Passwort oder beide Daten werden bereits in der URL direkt übergeben.

Anschließend wird über cdw (change working directory) der Pfad auf »/shared/bin/browsers/« gesetzt. Anhand der Dateiendung .exe wird die binäre Datenübertragung initiiert und die Datei »communicator.exe« übertragen. Sobald die Übertragung beendet ist, wird die Verbindung abgebaut.

13.1.3 ldap

Eine URL der Form:

```
ldap://ldap.meinefirma.com/dc=meinefirma,dc=com?cn,mail?sub?(cn=joe*)
```

liefert im Browser eine Ergebnismenge, die alle Mitarbeiter, deren Name mit Joe beginnt, namentlich samt E-Mail-Adresse auflistet. Die Syntax dieser *ldap*-URL ist Standard und sieht wie folgt aus:

```
ldap://<hostname:port#>/<baseDN>?<search-specification>
```

wobei die Suchspezifikation mittels einer durch Fragezeichen separierten Parameterübergabe umgesetzt wird. Die Parameter selbst haben einen festen Platz in der URL:

Erster Parameter: BaseDN der Suche

Zweiter Parameter: Attribute, die bei einem referenzierten Eintrag geliefert werden sollen (wird kein Attribut angegeben, werden alle freigegebenen Attribute dargestellt)

Dritter Parameter: Scope der Suche

Es existieren drei Scopes: base, subtree und one level:

base: es wird auf diesem BaseDN gesucht und nur dort

subtree: es wird beginnend bei der BaseDN abwärts im Baum gesucht

one: es wird genau ein Level unterhalb des BaseDN nach Informationen gesucht

Vierter Parameter: Suchfilter. Dessen Syntax lautet:

(<&/|>(attribute=searchvalue)... (attribute=searchvalue)),
wobei

»&« eine UND-Verknüpfung
»|« eine ODER-Verknüpfung darstellt.

Eine *ldap*-URL wird nicht nur in Browsern verwendet, sondern findet sich vielfach auch in Services, die auf diese Art Suchanfragen über einen String direkt am Verzeichnis platzieren können.

13.1.4 News (nntp)

Um eine Newsgroup auf einem bestimmten Server zu kontaktieren, wird im Allgemeinen zunächst der Server kontaktiert, die Grouplist geladen und anschließend die jeweils interessante Diskussionsrunde ausgewählt. Ein wesentlich schnellerer Zugriff auf bestimmte Newsgruppen ist der direkte Verweis über eine news-URL der Form:

```
news://()<host><domain>/<gruppe>.<untergruppe>...../<MessageID>
```

So ist beispielsweise

```
news://news.meinefirma.com/marketing.technik.autos
```

ein direkter Verweis in eine Diskussionsgruppe.

Dem Host in dieser Domain können wie bei den vorangegangenen Schemata User-ID und Passwort innerhalb der URL direkt mitgegeben werden.

Damit eine solche URL ausgewertet werden kann, muss der Browser eine Verbindung mit dem jeweils verwendeten Newsreader haben, an den er diesen Verweis übergibt. Der Reader seinerseits muss diese URL interpretieren können.

13.2 Verwendbares Characterset für URLs

Für die Verwendung von Sonderzeichen in URLs sind bestimmte Zeichen reserviert, die nur in dem dafür vorgesehenen Kontext verwendet werden dürfen. Diese sind:

: ? = @ ; / &

Nicht empfohlen ist die Verwendung der folgenden Zeichen, die teilweise funktionalen Charakter haben können:

% { } [ ] < > \ ^ ~ ` " #

Neben den rein alphanumerischen Zeichen sollten schließlich die folgenden Zeichen keinen Einfluss auf die Gültigkeit einer URL haben:

$ - , ! * ' _ . + ( )

Für die Verwendung der verbleibenden Zeichen, insbesondere von Umlauten ist die Verwendung der hexadezimalen Darstellungsweise »%hex« vorgesehen. Beispielsweise wird das Leerzeichen mit »%20«, das Gleichheitszeichen mit »%3D« dargestellt.

A Anhang A

Zensur oder Meinungsfreiheit im Internet?

Einer der wichtigsten Faktoren, die zum enormen Wachstum des World Wide Web und des Internets beigetragen haben, ist die Tatsache, dass jede Person sich und seine Meinung darstellen kann. Nahezu jeder Internet Service Provider bietet seinen Kunden eine Möglichkeit für die Installation einer Homepage an, die man beliebig gestalten kann und die von jedem Punkt im Internet aus erreichbar ist.

Mehr noch: Jeder hat die gleichen Möglichkeiten für den Aufbau einer Seite, der eigenen Kreativität sind keine Grenzen gesetzt (wenn man vom Speicherplatz sowie der Veröffentlichung indizierter Inhalte absieht).

Gerade durch diese Voraussetzungen ergibt sich eine ungeheure Meinungsvielfalt, in der sich Gruppierungen untereinander austauschen, Interessenten informieren und Konkurrenten bzw. Gegner gegenseitig analysieren können.

Diese Vielfalt an Informationen, Meinungen und Daten bringt Menschen aus den unterschiedlichsten Regionen der Welt zusammen, unterstützt Kinder und Jugendliche beim Lernen und bietet jedem die Möglichkeit für den Zugriff auf ein ungeheures Spektrum an Wissen, Nachrichten und Daten. Es werden bereits die ersten Stimmen laut, die den Zugang zum Internet als Menschenrecht fordern.

Es gibt aber auf der anderen Seite auch Gruppen, die dieses Medium und seine Möglichkeiten missbrauchen. Das Spektrum ist sehr weit gefächert und reicht von Kinderpornografie bis hin zu politischen Extremen. Computerviren und Hackerangriffe zielen auf die Ressourcen der Anwender, sei es aus Zerstörungswut oder zur persönlichen Bereicherung. Terroristen nutzen Verschlüsselungsverfahren zur geheimen Abstimmung ihrer Angriffe. Kraftwerke, Flughäfen, Krankenhäuser oder militärische Raketenabschussbasen können Ziele für Angreifer aus dem Internet sein.

Der Erfolg des Internets resultiert aus der Vielfalt der Möglichkeiten. Einschränkungen behindern dessen Weiterentwicklung. Man kann Richtlinien entwerfen und versuchen, deren Umsetzung voranzutreiben. So dürfen indizierte Seiten in Deutschland nicht veröffentlicht werden. Dies verhindert jedoch nicht, dass ein anderer Server in einem anderen Land die gleichen Daten anbietet. Manche Staaten verbieten die Verwendung der starken Verschlüsselung für den Datentransfer, sie können aber nicht verhindern, dass außerhalb (und im Verborgenen auch innerhalb) des Landes diese Technik angewandt wird.

In diesem Spannungsfeld zwischen Meinungsfreiheit und Zensur muss man sich vergegenwärtigen, welche Ziele für Einschränkungen in Frage kommen. Hier einige Szenarios und deren Wirkung in absteigende Stufen der Toleranz:

- Absolute Freiheit in Publikation und Aktionen

  Jede Art von Information kann veröffentlicht werden. Ob Webseiten politische Meinungen anbieten, Raubkopien von Musikstücken zum Download bereitstellen oder indiziertes Material anbieten, ist unerheblich. Angriffe auf Ressourcen oder Institutionen bleiben straffrei.

  In einer solchen Umgebung würde sich das Internet sehr bald selbst zerstören, da es keine rechtliche Handhabe gibt, um dem aktiven Missbrauch dieses Mediums entgegenzutreten.

  Es gibt so gut wie kein Land, in dem solche Bedingungen herrschen.

- Publikationsfreiheit und Schutz vor Angriffen

  Aktiv schädliches Verhalten wie Angriffe auf Ressourcen und Institutionen werden unter Strafe gestellt. Ein Maßnahmenkatalog bestimmt die Bestrafung bei Verstößen gegen diese Vorgabe. Publikationen jeder Art sind erlaubt, unabhängig von Form und Inhalt.

  Unter solchen Bedingungen würden viele Hersteller massiven wirtschaftlichen Schaden nehmen, da ihre Produkte ohne Schutz vervielfältigt würden. Indizierte Dokumente würden ohne Kontrolle verbreitet werden, ohne dass es Mittel der Regulierung gibt.

  Das Internet bietet punktuell solche Bedingungen, die Anbieter von solchen Diensten werden aber isoliert, deren Erreichbarkeit wird verhindert.

- Verwendung von Publikationsrichtlinien

  Grundsätzlich dürfen alle Meinungen und Dokumente veröffentlicht werden, Ausnahmen bilden Inhalte, die den Menschenrechten widersprechen oder die einen offensichtlichen Verstoß gegen die Besitzverhältnisse von Personen oder Institutionen darstellen.

  Diese Bedingungen spiegeln weite Teile des heutigen Internets wieder. Es kommt immer wieder einmal zu Verstößen gegen die Einschränkungen, die Betreiber solcher Seiten werden aber in der Regel recht schnell aufgefordert, das Angebot zu entfernen.

- Jugendschutz

  Die Publikation von Daten aller Art wird eingeschränkt. So sind jugendgefährdende Seiten durch einen Schutzmechanismus verborgen, den man nur entsperren kann, wenn man die Voraussetzungen erwirbt.

Beispiele:

Ein Internet Service Provider stellt ein Jugendschutzprogramm auf, das Eltern für ihre Kinder aktivieren können. Damit wird verhindert, dass Minderjährige in Kontakt mit für sie schädlichen Seiten kommen, ohne ihnen dabei die Möglichkeiten der Informationsvielfalt des Internets zu nehmen.

Webseiten mit jugendgefährdendem Inhalt müssen über Authentifizierungsmechanismen verborgen werden, die Freischaltung wird nach Überprüfung an Einzelpersonen vergeben.

Grundsätzlich herrschen diese Bedingungen in Deutschland vor. Die gesetzlichen Regulierungsmöglichkeiten reichen allerdings nur bis zur Landesgrenze und verhindern nicht, dass internationale Seiten diese Anforderungen missachten.

- Moralische, ideologische oder religiöse Grundsätze

  Die Publikation von bestimmten Themen ist aus moralischen, ideologischen oder religiösen Gründen nicht gestattet.

  Unter solchen Bedingungen spricht man von Zensur. Länder, Regierungen oder Institutionen gestatten es nicht, dass Dokumente veröffentlicht oder empfangen werden, die nicht ihren Grundsätzen entsprechen.

  Solche Richtlinien umzusetzen, ist schwierig. Entweder koppeln sich solche Gruppierungen vom Internet ab oder sie versuchen, unerwünschte Inhalte zu filtern oder Netzwerke zu sperren.

Im Internet ist es schwierig, inhaltliche Filter umzusetzen. Kapitel 10 hat gezeigt, wie kompliziert es sein kann, wenn ein Unternehmen seinen Mitarbeitern den Zugriff auf das Internet mit Hilfe von Filtern beschränken will. Man kann Webseiten blockieren, die bestimmten Kriterien genügen. Diese beruhen weitgehend auf Worterkennung, die über Regular Expressions festgelegt wird. Es ist schwierig, einen Filtersatz so zu optimieren, dass die unerwünschten Seiten abgewiesen werden, ohne dass unkritische Seiten außen vor bleiben.

Werden in Newsgruppen Artikel gepostet, die in vordefinierten Filtern hängen bleiben, so kann dies auch Postings betreffen, die das Thema des blockierten Begriffs ernsthaft diskutieren.

Bilder, Videos oder Musikstücke sind noch schwerer zu filtern. Bei Dateien, die mit der Erweiterung ».mp3« übertragen werden, ist eine Erkennung noch relativ einfach. Vielfach werden aber diese Daten komprimiert oder verschlüsselt, mit unverfänglichen Namen versehen und als Datei einer vollkommen anderen Anwendung getarnt.

Beispiel: Der User lädt sich von dieser Seite eine Datei mit der unverfänglichen Bezeichnung »Jump.xls«. Die anbietende Webseite gibt ihm die Anweisung, diese in »Jump.zip« umzutaufen und zu entpacken. Und so wird aus einem vermeintlichen Excel-Spreadsheet ein Musikstück von Van Halen.

Noch viel schwerer wird es, wenn Bilder gefiltert werden sollen. Keine bekannte Software ist in der Lage zu entscheiden, ob es sich bei einer Darstellung auf einem Bild um ein illegales Dokument handelt oder nicht, es sei denn, das Bild wurde schon einmal indiziert. Damit wird zwar nicht die Veröffentlichung von solchem Material verhindert, aber die Verbreitung stark eingeschränkt.

Selbst wenn Filter wie die hier beschriebenen im Einsatz sind, so bringen sie doch einen sehr gravierenden Nachteil mit sich: Der Dienst wird langsam. So wie der Einsatz von Virenscannern auf dem lokalen Rechner die Performance meist merklich senkt, so bremst das Verwenden von Filtern bei einem Diensteanbieter die Geschwindigkeit der Datenbereitstellung aus, was wiederum die Kunden nicht akzeptieren.

Eine Alternative zu Filtern ist die periodische Kontrolle von Internetseiten und die Beschränkung von News-Diensten auf Inhalte, die nicht aus dem Bereich der Giftküche stammen. So gehen immer mehr Internet Service Provider dazu über, den Diskussionsbereich »alt.binaries.*« komplett aus dem Angebot zu streichen, da hier die meisten Copyright-Verletzungen zu finden sind und die meisten pornografischen Dateien ausgetauscht werden. Diese Strategie hat zwei Seiten: Zum einen erspart sich der Betreiber ca. 90% des täglichen Newsfeeds, auf der anderen Seite leidet aber die Attraktivität des Anbieters, was sich in der Anzahl der Kunden niederschlägt.

Die Diskussion um die Art und den Umfang einer Kontrolle des Internets und seiner Inhalte wird sehr kontrovers geführt. Es wird immer Bereiche geben, in denen die Meinungsfreiheit über dem Schutz des Einzelnen steht. Genauso wird es auf der anderen Seite immer wieder Bemühungen um eine Zensur geben. Der vermutlich beste Weg liegt zwischen diesen beiden Extremen. So wie im täglichen Umfeld der gegenseitige Respekt voreinander mit einem Maximum an persönlicher Entfaltung einhergeht, sollten die Verhaltensweisen im Internet aufeinander abgestimmt sein. Mit einem Mindestmaß an Selbstdisziplin und notwendiger Reglementierung reguliert sich auch ein solches Medium wie das Internet selbständig.

B Anhang B

Verwendete Directory-Objektklassen

Die folgenden Objektklassen wurden verwendet und sind hier der Vollständigkeit noch einmal aufgelistet.

top

Unterstützt: ldap

Definition: Objektklasse, die als Superklasse für alle anderen Objektklassen des Directory verwendet wird. Interne Verwendung des Directory-Servers.

Superior Class: top

OID: 2.5.6.0

Erforderliche Attribute	Beschreibung
objectClass	Definiert die Objektklasse des Eintrags
Optionale Attribute	**Beschreibung**
aci	Beschreibt die Informationen der Zugriffskontrolle auf diesen Eintrag

person

Unterstützt: ldap

Definition: definiert Einträge, die im Allgemeinen natürliche Personen beschreiben. Diese Objektklasse ist die Basisklasse für die Objektklasse »organizationalPerson«.

Superior Class: top

OID: 2.5.6.6

Erforderliche Attribute	Beschreibung
objectClass	Definiert die Objektklasse des Eintrags
cn (commonName)	Familien- und Vorname einer Person
sn (surName)	Familienname einer Person

Optionale Attribute	Beschreibung
description	Textbeschreibung der Person
seeAlso	URL, die auf weitere Informationen zu dieser Person zielt
telephoneNumber	Telefonnummer der Person
userPassword	Passwort der Person. Mit diesem Wert kann sie sich gegenüber dem Server authentifizieren

organizationalPerson

Unterstützt: ldap

Definition: Diese Klasse definiert Personen, die mit einem Unternehmen oder einer Organisation assoziiert sind. Die Attribute cn und sn werden von der übergeordneten Klasse »person« vererbt und brauchen hier nicht neu definiert zu werden.

Superior Class: person

OID: 2.5.6.7

Erforderliche Attribute	Beschreibung
ObjectClass	Definiert die Objektklasse des Eintrags
Optionale Attribute	**Beschreibung**
DestinationIndicator	Land- und Stadtinformationen einer Person, die für den öffentlichen Telegramm-Service notwendig sind
fax (facsimileTelephoneNumber)	Faxnummer der Person
internationalIsdnNumber	ISDN-Nummer der Person
l (localityName)	Geografischer Ort, an dem die Person lebt
ou (organizationUnitName)	Organisationseinheit, zu der die Person gehört
physicalDeliveryOfficeName	Lieferadresse der Person
postalAddress	Postadresse der Person
postalCode	Die zur Postadresse zugehörige Postleitzahl
postOfficeBox	Informationen über das Postfach der Person
preferredDeliveryMethod	Informationen über die bevorzugte Art des Kontakts oder der Zulieferung
registeredAddress	Postalische Adresse, unter der die Person bei wichtigen Lieferungen erreichbar ist
st	Bundesstaat, in dem die Person lebt

Optionale Attribute	Beschreibung
street	Straße, in der die Person wohnt
teletexTerminalIdentifier	Identifier des Telex-Terminals der Organisation
telexNumber	Telex-Nummer der Organisation
title	Titel der Person
x121Address	X.121-Adresse der Organisation

inetorgPerson

Unterstützt: von einzelnen Herstellern von Directory-Servern, noch kein Standard

Definition: Definiert Einträge einer Person, die zu einer Organisation gehört. Die Attribute cn und sn werden von der übergeordneten Klasse »person« vererbt und brauchen hier nicht neu definiert zu werden.

Superior Class: person

OID: 2.16.840.1.113730.3.2.2

Erforderliche Attribute	Beschreibung
objectClass	Definiert die Objektklasse des Eintrags
Optionale Attribute	**Beschreibung**
audio	Nimmt eine Sound-Datei in Binärformat auf
businessCategory	Geschäftsfeld, in dem die Person involviert ist
carLicense	Autokennzeichen des Wagens der Person
departmentNumber	Abteilung, in der die Person arbeitet
displayName	Bevorzugter Name, den die Person bei einer Anzeige verwenden möchte
employeeNumber	Personalnummer der Person
employeeType	Art der Beschäftigung der Person
givenName	Vorname der Person
homePhone	Private Telefonnummer
homePostalAddress	Wohnadresse der Person
houseIdentifier	Hausnummer der Wohnung der Person
initials	Initialen der Person
jpegPhoto	Ein Bild in JPEG-Format
labeledURL	URL für eine Webadresse der Person
mail	Mailadresse der Person
manager	Name des verantwortlichen Vorgesetzten der Person

Optionale Attribute	Beschreibung
mobile	Mobilnummer der Person
o (OrganizationName)	Name des Unternehmens oder der Organisation, in der die Person beschäftigt ist
pager	Pager-Nummer der Person
photo	Ein Foto in binärer Form
preferredLanguage	Legt fest, welches die bevorzugte Sprache der Person ist
roomNumber	Nummer des Zimmers, in dem die Person anzutreffen ist
secretary	Der Name der/s Sektretär/in der Person
uid (userID)	User-Kennung der Person
userCertificate	Digitales Zertifikat der Person in Klartext
userClass	Digitales Zertifikat der Person in Binärform
userSMIMECertificate	Digitales Zertifikat der Person in Binärform. Bei der Verwendung von S/MIME wird dieses Attribut abgefragt
x500UniqueIdentifier	Reserviert

referral

Unterstützt: von einzelnen Herstellern von Directory Servern, noch kein Standard

Definition: Diese Objektklasse erlaubt das Verlinken von Directorys. Das Referral-Objekt ist eine Unterklasse der Superklasse top und nimmt den Verweis auf einen andere ldap-Datenquelle auf. Wenn ein Directory-Eintrag das »ref«-Attribut enthält, übergibt der Server einen Verweis auf eine andere ldap-Datenquelle (intern oder auf einem anderen Server), anstatt den angeforderten Wert zu liefern. Es liegt in der Verantwortung des Clients, diesen Verweis auszuwerten und die benötigten Daten von der anderen Datenquelle zu erfragen.

Superior Class: top

OID: 2.16.840.1.113730.3.2.6

Erforderliche Attribute	Beschreibung
objectClass	Definiert die Objektklasse des Eintrags
Optionale Attribute	**Beschreibung**
ref	LDAP URL in the format: »ldap://<server>:<portnumber>/<dn>"

C Anhang C

Protokolle und Ports

Übersicht über die im Internet verwendeten Standard-Ports. Die Liste erhebt keinen Anspruch auf Vollständigkeit.

Port	Protokoll	Dienstname	Bedeutung	RFC
7	tcp/udp	Echo	dupliziert empfangene Dateien	862
9	tcp/udp	discard	vernichtet empfangene Dateien	863
11	tcp/udp	active	Users	
13	tcp/udp	daytime	Datum und Uhrzeit	867
17	tcp/udp	qotd	Quote of the Day	
18	tcp/udo	msp	Message Send Protocol	
19	tcp/udo	chargen	Character Generator	864
20	tcp/udo	ftp-data	Datentransfer (Datenkanal)	959
21	tcp/udo	ftp	Datentransfer (Kommandokanal)	959
22	tcp/udo	ssh	SSH Remote Login Protocol	
23	tcp/udo	telnet	Remote Login – Telnet	1416
25	tcp/udo	smtp	Simple Mail Transfer Protocol	821, 822
37	tcp/udo	time	Time	
38	tcp/udo	rap	Route Access Protocol	

Port	Protokoll	Dienstname	Bedeutung	RFC
42	tcp/udo	nameserver, wins	Name Server, WINS	
43	tcp/udo	nicname	Who is	
49	tcp/udo	tacacs	Authentisierungsprotokoll	
53	tcp/udo	dns	Domain Name System	1034, 1035
65	tcp/udo	tacacs-ds	TACACS-Database Service	
66	tcp/udo	sql*net	Oracle SQL*NET	
67	tcp/udo	bootps	Bootstrap Protocol Server	
68	tcp/udo	bootpc	Bootstrap Protocol Client	
69	tcp/udo	tftp	Trivial File Transfer Protocol	1350
70	tcp/udo	gopher	Gopher	
79	tcp/udo	finger	Informationen über Benutzer	1288
80	tcp/udo	http	Hypertext Transfer Protocol	2068
88	tcp/udo	kerberos	Kerberos	1510
92	tcp/udo	npp	Network Printing Protocol	
107	tcp/udo	rtelnet	Remote Telnet Service	
109	tcp/udo	pop2	Post Office Protocol Version 2	937
110	tcp/udo	pop3	Post Office Protocol Version 3	1939
111	tcp/udo	sun-rpc	Sun Remote Procedure Call	1057
113	tcp/udo	auth	Authentication Service	1413
115	tcp/udo	sftp	Simple File Transfer Protocol	

Port	Protokoll	Dienstname	Bedeutung	RFC
117	tcp/udo	uucp-path	UUCP Path Service	
118	tcp/udo	sqlserv	SQL Services	
119	tcp/udo	nntp	Network News TransferProtocol	977
123	tcp/udo	ntp	Network Time Protocol	
130	tcp/udo	cisco-fna	cisco FNATIVE	
131	tcp/udo	cisco-tna	cisco TNATIVE	
132	tcp/udo	cisco-sys	cisco SYSMAINT	
135	tcp/udo	RPC	Remote Procedure Call	
137	tcp/udo	netbios-ns	NetBios Name Service	1001,1002
138	tcp/udo	netbios-dgm	NetBios Datagram Service	1001,1002
139	tcp/udo	netbios-ssn	NetBios Session Service	1001,1002
143	tcp/udo	imap	Internet Message Access Protocol v2	1064
150	tcp/udo	sql-net	SQL-Net	
156	tcp/udo	sqlsrv	SQL Service	
161	tcp/udo	snmp	Simple Network Management Protocol	1901,1907
162	tcp/udo	snmp-trap	Simple Network Management Protocol	1901,1907
170	tcp/udo	print-srv	Network PostScript	
194	tcp/udo	irc	Internet Relay Chat Protocol	1459
213	tcp/udo	ipx	IPX	
220	tcp/udo	imap3	Interactive Mail Access Protocol v3	1203

Port	Protokoll	Dienstname	Bedeutung	RFC
389	tcp/udo	ldap	Lightweight Directory Access Protocol	
396	tcp/udo	netware-ip	Novell Netware über IP	
401	tcp/udo	ups	Uninterruptible Power Supply	
443	tcp/udo	https	http Protokoll über TLS/SSL	
458	tcp/udo	appleqtc	apple quick time	
500	tcp/udo	isakmp	Internet Security Association and Key Management Protocol	2408
512	tcp	exec	Remote process execution	
513	tcp	login	Remote login	1282
513	udo	who	maintains data bases showing who's	
514	tcp	shell	cmd, wie exec, nur mit automatischer Authentifizierung	
514	udo	syslog	Protokollierung	
515	tcp/udo	printer	Spooler	
517	tcp/udo	talk	Realtime Punkt-zu-Punkt Nachrichtenaustausch	
525	tcp/udo	timed	Timeserver	
540	tcp	uucp	uucp file transfer service	
635	udo	mountd	Linux mountd	
636	tcp/udo	*ldaps*	*ldap*-Protokoll über TLS/SSL	
666	tcp/udo	Doom	IDs Doom	
749	tcp/udo	kerberos-adm	kerberos administration	
989	tcp/udo	ftps-data	ftp-Protokoll, Daten über TLS/SSL	
990	tcp/udo	ftps	ftp-Protokoll, control über TLS/SSL	

Port	Protokoll	Dienstname	Bedeutung	RFC
991	tcp/udo	nas	Netnews Administration System	
992	tcp/udo	telnets	telnet-Protokoll über TLS/SSL	
993	tcp/udo	imaps	imap4-Protokoll über TLS/SSL	
994	tcp/udo	ircs	Irc-Protokoll über TLS/SSL	
995	tcp/udo	pop3s	pop3-Protokoll über TLS/SSL	
1080	tcp	socks	Applikations-Proxyservice	
1214	tcp/udo	KAZAA	KAZAA	
1243	tcp/udo	sub7	Trojaner sub7	
1366	tcp/udo	netware-csp	Novell NetWare Comm Service	
1433	tcp/udo	ms-sql-s	Microsoft-SQL-Server	
1434	tcp/udo	ms-sql-m	Microsoft-SQL-Monitor	
1512	tcp/udo	Wins	Microsoft's Windows Internet Name Service	
1525	tcp/udo	Orasrv	Oracle	
1527	tcp/udo	Tlisrv	Oracle	
1529	tcp/udo	coauthor	Oracle	
1720	tcp/udo	NetMeeting	NetMeeting	
1761	tcp/udo	Microsoft	Systems Management Server (SMS) v.1.2	
1762	tcp/udo	SMS v.1.2	Remote Control	
1763	tcp/udo	SMS v.1.2	Remote Chat	
1764	tcp/udo	SMS v.1.2	File Transfer	

Port	Protokoll	Dienstname	Bedeutung	RFC
4000	udo	Icq	ICQ	
5190	tcp/udo	AOL	AOL	
5631	tcp/udo	pcanywhere-data	pcAnywhere data	
5632	tcp/udo	pcanywheres-tat	pcAnywhere stat	
6000-6063	tcp/udo	x11	X Window System	
6346	tcp	Gnutella	Gnutella	
6665-6669	tcp/udo	IRCU	Irc	1459
6699	tcp/udo	Napster	Napster Peer-to-Peer	
7004	tcp/udo	afs3-kaserver	AFS/Kerberos Authentifizierungs-Service	
7070	udo	RealAudio	RealAudio	
12345, 12346, 20034	tcp	NetBus	Trojaner NetBus	
27000	tcp/udo	QuakeWorld	QuakeWorld	
27010	tcp/udo	Half-Life	Half-Life	
27910	tcp/udo	Quake 2	Quake 2	
27950	tcp/udo	Quake 3	Quake 3	
27990	tcp/udo	Kingpin	Kingpin	
28000	tcp/udo	Tribes	Tribes	
31337	udo	BO	Trojaner BackOrifice	
31790, 31789	udo	Hack'a'Tack	Trojaner Hack'a'Tack	
33434-33523	udo	traceroute	Traceroute	1393

D Anhang D

Verwendete Abkürzungen und ihre Bedeutung

ACI	Access Control Item
ACL	Access Control List
ADMD	Administration Management Domain
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
ASP	Application Service Provider
ASP	Active Server Page
AU	Access Unit
B2B	Business To Business
B2C	Business To Consumer
bcc	blind carbon copy
CA	Certificate Authority
cap	Calendar Access Protocol
carp	Cache Array Routing Protocol
CAST	Carlisle Adams, Stafford Tavares (ein 64-Bit Verschlüsselungsverfahren)
cc	carbon copy
CCITT	Consultative Committee for International Telegraph and Telephone
CIS	Case Ignore String
CES	Case Exact String
CGI	Common Gateway Interface
CPI	Capability and Preference Information
CPU	Central Processing Unit
dap	Directory Access Protocol
dc	Domain Component
DENIC	Deutsches Network Information Center
DES	Data Encryption Standard
DIB	Directory Information Base
disp	Directory Information Shadowing Protocol

DIT	Directory Information Tree
DMZ	Demilitarisierte Zone
dn	Distinguished Name
DNS	Domain Name System
dop	Directory Operations Protocol
DSA	Directory System Agent
dsp	Directory System Protocol
DSS	Digital Signature Standard
DUA	Directory User Agent
ebpp	electronic bill presentment and payment
EIT	Encoded Information Type
ftp	File Transfer Protocol
GCHQ	Government Communications Headquarters
gif	Graphics Interchange Format
html	Hypertext Markup Language
http	Hypertext Transfer Protocol
https	Hypertext Transfer Protocol Secure
icp	Internet Cache Protocol
IDEA	International Data Encryption Algorithm
imap4	Internet Message Access Protocol Version 4
IMC	Internet Mail Consortium
imip	iCalendar Message-Based Interoperability Protocol
ip	Internet Protocol
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITEF	Internet Engineering Task Force
ITU	International Telecommunications Union
jsp	Java Server Page
LAN	Local Area Network
ldap	Lightweight Directory Access Protocol
ldaps	Lightweight Directory Access Protocol Secure
LDIF	ldap Directory Interchange Format
ldup	ldap Replication Update Protocol
MD5	Message Digest Version 5
MHS	Message Handling System
MIME	Multipurpose Internet Mail Extentions

MS	Message Store
MTA	Message Transfer Agent
MTS	Message Transfer System
MX	Mail eXchange
NIST	National Institute for Standards and Technology
nntp	Network News Transfer Protocol
NSA	National Security Agency
pac	Proxy-Autoconfiguration
PDA	Personal Digital Assistant
PES	Proposed Encryption Standard
PGP	Pretty Good Privacy
PHP	Hypertext Preprocessor
PKI	Public Key Infrastructure
pop3	Post Office Protocol Version 3
PMD	Private Management Domain
rdn	Relative Distinguished Name
RFC	Request for Comments
RSA	Rivest, Shamir and Adleman (Bezeichnung für ein Verschlüsselungsverfahren)
SHA	Secure Hash Algorithm
S/MIME	Secure Multipurpose Internet Mail Extentions
smtp	Simple Mail Transfer Protocol
snmp	Simple Network Management Protocol
SSI	Server Side Include
Ssl	Secure Socket Layer
tcp	Transmission Control Protocol
tcp/ip	Transmission Control Protocol/Internet Protocol
UA	User Agent
UBE	Unsolicited Bulk E-mail
udp	User Datagram Protocol
URI	Unified Resource Identifier
URL	Uniform Resource Locator
uuencode	Unix to Unix encode
uudecode	Unix to Unix decode
WAE	Wireless Application Environment
WAN	Wide Area Network

wap	Wireless Access Protocol
WBMP	Wireless Bitmap
WBXML	Wireless Binary Extended Markup Language
wcap	Web-Calendar Access Protocol
wdp	Wireless Datagram Protocol
wml	Wireless Markup Language
wsp	Wireless Session Protocol
wtls	Wireless Transaction Layer Security
wtp	Wireless Transaction Protocol
www	World Wide Web
wysiwyg	What You See Is What You Get
Xml	Extended Markup Language

Literatur

Bei keinem Link im Internet kann heute garantiert werden, dass morgen noch alle URLs Gültigkeit besitzen. Wenn demnach der eine oder andere Verweis im Laufe der Zeit nicht mehr zum Ziel führt, so bitte ich dies zu entschuldigen.

[1] T. Berners-Lee, R. Cailliau: »World Wide Web: Proposal For A Hypertext Project« http://www.w3.org/pub/WWW/Proposal.html

[2] RFC 2252 »Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions« http://www.ietf.org/rfc/rfc2252.txt

[3] RFC 1738, T. Berners-Lee, L. Masinter, M. McCahill: Uniform Resource Locators (URL),
URL: »http://www.ietf.org/rfc/rfc1738.txt«

[4] RFC 2256 »A Summary of the X.500(96) User Schema for use with LDAPv3« http://www.ietf.org/rfc/rfc2256.txt

[5] »LDAP Replication Update Protocol (ldup)«
http://www.ietf.org/html.charters/ldup-charter.html

[6] RFC 821: »Simple Mail Transfer Protocol«, Jonathan B. Postel, August 1982, URL: »http://www.ietf.org/rfc/rfc821.txt «

[7] RFC 822: »Standard for Format of ARPA Internet Text Messages«, STD 11, RFC 822, D. Crocker, August 1982.
URL: http://www.ietf.org/rfc/rfc822.txt

[8] RFC 882: »Domain Names – Concepts and Facilities«, P. Mockapetris, November 1983, URL: »http://www.ietf.org/rfc/rfc882.txt«

[9] RFC 2045: »Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies«, N. Freed et al., November 1996, URL: » http://www.ietf.org/rfc/rfc2045.txt«

[10] RFC 2046: »Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types«, N. Freed, N. Borenstein, November 1996,
URL: »http://www.ietf.org/rfc/rfc2046.txt«

[11] International Telecommunication Union:
http://www.itu.int/home/index.html

[12] RFC 1939: »Post Office Protocol – Version 3«, J. Myers, M. Rose, Mai 1996, URL: »http://www.ietf.org/rfc/rfc1939.txt«

[13] RFC 2060: »Internet Message Access Protocol – Version 4rev1«, M. Crispin, University of Washington, Dezember 1996,
URL: »http://www.ietf.org/rfc/rfc2060.txt«

[14] RFC 2487: »SMTP Service Extention for Secure SMTP over TLS«, P.Hoffman, Internet Mail Consortium, Januar 1999,
URL: »http://www.ietf.org/rfc/rfc2487.txt«

[15] RFC 2554: »SMTP Service Extension for Authentication«, J.Myers, Nestcape Communictions, März 1999,
URL: »http://www.ietf.org/rfc/rfc2554.txt«

[16] RFC 1870: »SMTP Service Extension for Message Size Declaration«, J. Klensin, N. Freed, K. Moore, November 1995,
URL: »http://www.ietf.org/rfc/rfc1870.txt«

[17] RFC 1985: »SMTP Service Extension for Remote Message Queue Starting«, J. De Winter, August 1996,
URL: »http://www.ietf.org/rfc/rfc1985.txt«

[18] RFC 1869:« SMTP Service Extensions«, J. Klensin, WG Chair, N. Freed, M. Rose, E. Stefferud, D. Crocker, November 1995,
URL: »http://www.ietf.org/rfc/rfc1869.txt«

[19] RFC977: »Network News Transfer Protocol«, Brian Kantor, Phil Lapsley, Februar 1986, URL: »http://www.ietf.org/rfc/rfc977.txt«

[20] RFC1036: »Standard for Interchange of USENET Messages«, M. Horton, R. Adams, Dezember 1987,
URL: »http://www.ietf.org/rfc/rfc1036.txt«

[21] RFC 2068: R. Fielding, UC Irvine, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee: »Hypertext Transfer Protocol -- HTTP/1.1« Januar 1997,
URL: »http://www.ietf.org/rfc/rfc2068.txt«

[22] »Nordic WAIS/World Wide Web Project«, Anders Ardö, Sigfrid Lundberg, URL: http://www.lub.lu.se/W4.html

[23] »A Method for Web Robots Control«, M. Koster, 1997,
URL: http://www.robotstxt.org/wc/norobots-rfc.html

[24] »The Anatomy of a Large-Scale Hypertextual Web Search Engine«, Sergey Brin, Lawrence Page, URL: http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm

[25] »The PageRank Citation Ranking: Bringing Order to the Web«, Larry Page, Sergey Brin, R. Motwani, T. Winograd,
URL: http://citeseer.nj.nec.com/368196.html

[26] »Angewandte Kryptographie«, Bruce Schneier, 1996, Addison-Wesley

[27] »On The Design And Security Of Block Ciphers«, X. Lai, ETH Series in Information Processing, v. 1, Konstanz, 1992, Hartung-Gorre Verlag

[28] RFC 2612: »The CAST-256 Encryption Algorithm«, C. Adams, J. Gilchrist, Juni 1999, »http://www.ietf.org/rfc/rfc 2612.txt«

[29] »Geheime Botschaften«, Simon Singh, 1999, Hanser

[30] »Abenteuer Kryptologie«, Reinhard Wobst, 2001, Addison-Wesley

[31] »Verschlüsselte Botschaften«, Rudolf Kippenhahn, 1997, Rowohlt

[32] RFC 1186: »The MD4 Message Digest Algorithm«, R.L. Rivest, Oktober 1990, »http://www.ietf.org/rfc/rfc1186.txt«

[33] RFC 1321: »The MD5 Message Digest Algorithm«, R.L. Rivest, April 1992, »http://www.ietf.org/rfc/rfc1321.txt«

[34] »Proposed Federal Information Processing Standard For Secure Hash Standard«, Federal Register, v. 57, n. 21, 31. Januar 1992, pp.3747-3749

[35] »PGP – Pretty Good Privacy«, Simson Garfinkel, O'Reilly & Associates, Inc.

[36] »IETF Working Group for calendaring and scheduling (calsched)«, RFCs 2445,2446,2447, 2739

[37] RFC 2445: »Internet Calendaring and Scheduling Core Object Specification«, F. Dawson, D. Stenerson,
URL: »http://www.ietf.org/rfc/rfc2445.txt«

[38] RFC 2446: »iCalendar Transport-Independent Interoperability Protocol (iTIP) Scheduling Events, BusyTime, To-dos and Journal Entries«, S. Silverberg, S. Mansour, F. Dawson, R. Hopson,
URL: »http://www.ietf.org/rfc/rfc2446.txt«

[39] RFC 2447: »iCalendar Message-Based Interoperability Protocol (iMIP)«, F. Dawson, S. Mansour, S. Silverberg,
URL: »http://www.ietf.org/rfc/rfc2447.txt«

[40] »Compact html for Small Information Appliances« von Tomihisa Kamada, ACCESS Co.,Ltd., zu finden unter:
http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/

[41] »Wireless Session Protocol«, WAP Forum, Januar 1998.
URL: http://www.wapforum.org

[42] »Wireless Datagram Protocol«, WAP Forum, April 1998.
URL: http://www.wapforum.org

[43] »Wireless Markup Language Specification«, WAP Forum, April 1998.
URL: http://www.wapforum.org

[44] »WMLScript Specification«, WAP Forum, April 1998.
URL: http://www.wapforum.org

[45] RFC 2426: »vCard MIME Directory Profile«, F. Dawson, T. Howes,
URL: http://www.imc.org/rfc2426

Stichwortverzeichnis