dieses Feldes wird dem Mailprogramm signalisiert, daß der Inhalt des Briefes mit dem MIME-Standard konform geht.
Kannte der RFC 822 zwei Teile eines Briefes, nämlich den Kopf (**Header**) und den Text (**Body**), so können Briefe im MIME-Format aus mehreren Teilen bestehen. Die Zeile `MIME-Version: 1.0` muß nur einmal im Kopf des Briefes auftauchen. Die anderen Felder, welche der MIME-Standard definiert, können öfter verwendet werden. Sie beschreiben dann jeweils die Einzelteile, aus denen der Brief besteht. Ein Beispiel:

```
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="8323328-2120168431-
824156555=:325"

--8323328-2120168431-824156555=:325
Content-Type: TEXT/PLAIN; charset=US-ASCII
Textnachricht....

--8323328-2120168431-824156555=:325
Content-Type: IMAGE/JPEG; name="teddy.jpg"
Content-Transfer-Encoding: BASE64
Content-ID: <Pine.LNX.3.91.960212212235.325B@localhost>
Content-Description:
/9j/4AAQSkZJRgABAQAAAQABAAD//gBqICBJbXBvcnRlZCBmcm9tIE1JRkYg
aW1hZ2U6IFh0ZWRkeQoKQ1JFQVRPUjogWFYgVmVyc2lvbiAzLjAwICBSZXY6
[..]
se78SaxeW7Qz3zeW33tqqu7/AHtv3qyaKmOGox96MSeSIUUUVuUFFFFABRRR
RZAFFFFABRRRTAKKKKACiiigAooooA//2Q==
--8323328-2120168431-824156555=:325--
```

Mit dem Feld `Content-Type:` wird der Inhalt eines Briefes beschrieben. Im Kopf des Briefes legt das Feld `Content-Type:` den Aufbau des ganzen Briefes fest. Das Stichwort `Multipart` signalisiert, daß der Brief aus mehreren Teilen besteht. Der Untertyp von `Multipart Mixed` liefert den Hinweis, daß der Brief aus heterogenen Teilen besteht. Der erste Teil dieses Beispiels besteht aus Klartext und der zweite Teil enthält ein Bild. Die einzelnen Teile des Briefes werden durch eine Zahlenkombination eingegrenzt, die im Kopf des Briefes im Feld `Boundary` festgelegt wurde. Diese **Grenze** (**Boundary**) ist nichts weiter als eine eindeutig identifizierbare Zeichenfolge, anhand derer die einzelnen Teile einer E-Mail unterschieden werden. Ein MIME-konformes Mailprogramm sollte anhand dieser Informationen jeden einzelnen Teil adäquat darstellen können. Im Feld `Content-Type:` können sieben verschiedene Typen festgelegt werden, die jeweils bestimmte Untertypen zur genaueren Beschreibung des Inhalts umfassen (Auflistung der Untertypen ohne Anspruch auf Vollständigkeit):

- `text: plain, enriched, html`

- multipart: mixed, alternative, parallel, digest
- message: rfc822, partial
- image: jpeg,gif
- audio: basic
- video: mpeg
- application: octet-stream, PostScript, active

Die Typen `image`, `audio`, `video` sprechen für sich selbst. Der Typ `message` sollte dann verwendet werden, wenn der Brief einen anderen Brief enthält (z.B. einen weitergeleiteten Brief). Der Typ `application` ist für die Beschreibung ausführbarer Programme gedacht. Dem Typ `text` kann noch der Parameter `charset:` beigefügt werden. Die Vorgabe der Programme lautet in der Regel `charset: us-ascii`. Anstelle von `us-ascii` kann hier auch `iso-8859-1` für den deutschen Zeichensatz eingetragen werden. Inzwischen werden auch vielfach E-Mails, markiert durch `text/html`, wie HTML-Seiten codiert (beim Netscape-Browser ist sogar Klartext und HTML-Darstellung voreingestellt, man bekommt den Brief also doppelt).

7.1.1 SMTP - Simple Mail Transfer Protocol

Der urspüngliche Standard für **SMTP** - niedergelegt im RFC 821[40] [Post 82] - stammt aus dem Jahr 1982 und gilt, abgesehen von einigen Erweiterungen, nach wie vor. Dieser RFC 821 legte ein Minimum an Schlüsselworten fest, die jede Implementation von SMTP (d. h. die Verkörperung von SMTP in einem Programm) beherrschen muß. Diese sind in Tabelle 12 aufgelistet.

Die Verbindung eines MTA zu einem anderen läßt sich von Hand nachstellen. Hier die Kommunikation von `mailgw2.muc.meinedomain.de` zu `blackhole1.muc.meinedomain.de`:

```
telnet blackhole1.muc.meinedomain.de 25
Trying 53.122.200.50...
Connected to blackhole1.muc.meinedomain.de.
Escape character is '^]'.
220 Sendmail ESMTP --- T-Systems Munich - Internet Mail Gateway
helo mailgw2.muc.meinedomain.de
250 blackhole1.muc.meinedomain.de
  Hello oberhait@mailgw2.muc.meinedomain.de [53.122.200.49], pleased to meet you
mail from:<oberhaitzinger_b@gmx.de>
```

[40]abgelöst durch RFC 2821 [KlEd 01]

Kommando	**Argument**	**Beschreibung**
`HELO`	Systemname	Beginn, Name des sendenden Systems
`MAIL From:`	Absenderadresse	Beginn der Übermittlung
`RCPT To:`	Empfängeradresse	Adressat der E-Mail
`DATA`		Brieftext, Ende durch eine Zeile mit '.'
`HELP`	Topic	Hilfestellung
`VRFY`	Mailadresse	Mailadresse verifizieren
`EXPN`	Mailadresse	Mailadresse expandieren (z. B. Liste)
`RSET`		Senden abbrechen, Zurücksetzen
`NOOP`		nichts tun
`QUIT`		Verbindung beenden

Tabelle 12: Schüsselwörter für SMTP laut RFC 821 [Post 82]

```
250 2.1.0 <oberhaitzinger_b@gmx.de>... Sender ok
rcpt to:<oberhait@muc.meinedomain.de>
250 2.1.5 <oberhait@muc.meinedomain.de>... Recipient ok
data
354 Enter mail, end with "." on a line by itself
hallo,

diese mail soll zeigen, wie smtp von hand gesprochen wird.
so koennen mailverbindungen getestet werden.
.
250 2.0.0 g3F9CqwC013177 Message accepted for delivery
quit
221 2.0.0 blackhole1.muc.meinedomain.de closing connection
Connection closed by foreign host.
```

Die Mitteilung `Sender ok` bzw. `Recipient ok` besagt nicht, daß eine Überprüfung des Usermailaccounts erfolgreich war, sondern daß keine Konfigurationseinschränkungen gegen diese Mailverbindung sprechen. Dies wird noch näher in Abschnitt 7.1.3 erläutert.
Beim Verbindungsaufbau meldet sich der lokale MTA mit einer **Begrüßungszeile**. Der lokal empfangende MTA wird mit `HELO` angesprochen und als sendender MTA der Name des Sendesystems `mailgw2.muc.meinedomain.de` angegeben. Der lokale MTA antwortet mit einem Zahlencode (hier 250), der dem Sender-MTA signalisiert, daß seine geforderte Aktion in Ordnung geht. Die Klarschrift nach dem Zahlencode dient nur der besseren Lesbarkeit für den Menschen (z. B. für den, der Fehler suchen muß). Auf `MAIL FROM:` folgt die Adresse des Absenders, und auf `RCPT TO:` die des Empfängers. Auf das Schlüsselwort `DATA` folgt schließlich der ganze Brief, also sowohl die Kopfzeilen, als auch der Text. Der Empfänger-MTA wird solange Text erwarten, bis ihm der Sender-MTA über eine Zeile, die nur einen Punkt enthält, signalisiert, daß der Brief zu Ende ist. Nach der letzten

Bestätigung des Empfänger-MTAs könnte der Sender den nächsten Brief übermitteln, wiederum beginnend mit `MAIL FROM:`. Nach dem Empfang des Briefes kopiert der lokale MTA den Brief in die Postfach-Datei des Empfängers.
Der RFC 821 [Post 82] legte noch einige weitere Schlüsselworte fest, z. B. `EXPN` für **expand**, welches eine Ausgabe der User einer Mailingliste erlaubt, oder `VRFY` für **verify**, mittels dessen eine Validierung der Mailadresse erfolgt. Eine ganze Reihe von RFCs haben den Standard für SMTP erweitert. Die erweiterte Version heißt nun offiziell **ESMTP** (für **Extended SMTP**). Hinzugekommen sind beispielsweise Schlüsselworte für die Unterstützung von 8bit-Briefen (z.B. solche mit Umlauten), und die Möglichkeit, eine maximale Größe für Briefe, die empfangen werden, festzulegen.

7.1.2 Zusammenspiel DNS und SMTP

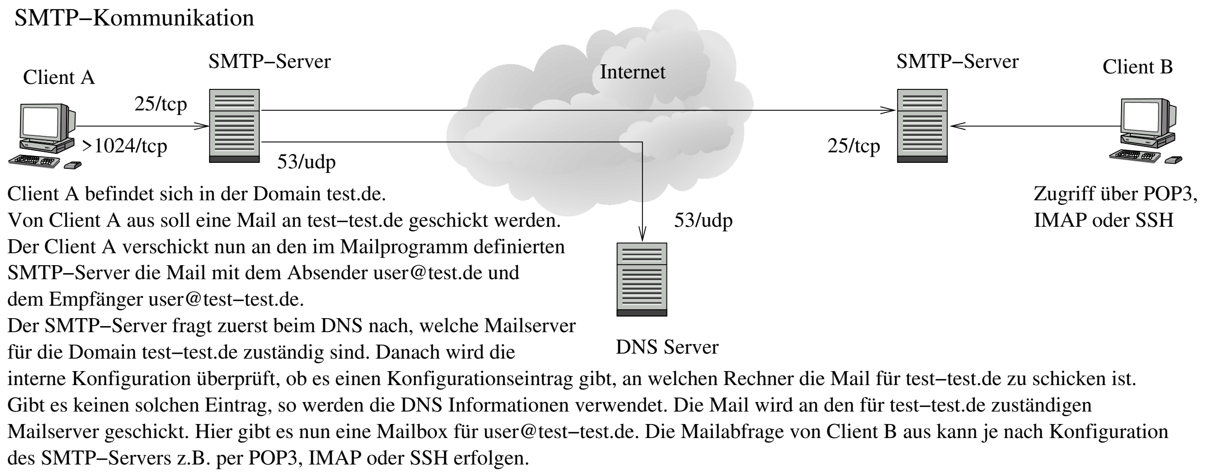

Abbildung 65: Schematische Darstellung einer SMTP-Kommunikation

Wie weiß nun ein Mailprogramm, wohin es seine Mails schicken soll? Diese Frage ist recht einfach zu beantworten: ein Mailclient kennt seinen Mailserver (MTA), an den er alle Mails zustellt und von dem er auch die zu empfangenden Mails erhält. Gehen wir nun davon aus, daß der MTA direkt Verbindung zum Internet herstellen kann. So wird zuerst der DNS abgefragt, welche Mailserver mit welcher Priorität für die Empfängerdomain zuständig sind. Danach wird gecheckt, ob in der internen Konfiguration des MTAs eine spezielle Zustellung für diese Domain verlangt wird. Ist dies nicht der Fall, so werden die Informationen aus dem DNS für die Zustellung verwendet. Der Verbindungsaufbau erfolgt mit dem Mailserver, der die niedrigste Priorität anhand der MX Records aufweist. Scheitert dieser Versuch, wird der Mailserver mit der zweitniedrigsten Priorität verwendet und so weiter. Wurde nun die Mail an einen der zuständigen Mailserver erfolgreich übertragen, so wird die Mailinformation bis auf einen Logfileeintrag vom sendenden Server gelöscht.

Der empfangende Server entscheidet entweder anhand interner Listen oder eines internen DNS, wohin die gerade emfangende Mail zugestellt werden soll. Der letzte Mailserver in der Kette legt die Mail beim Empfänger im Postfach ab. In Abbildung 65 ist eine SMTP-Kommunikation schematisch dargestellt.

7.1.3 Sicherheitsaspekte

Spam

Spam ist unverlangte Massen E-Mail, die über das Internet verschickt wird. Aus Sicht des Absenders ist es eine Art Postwurfsendung, oft an eine Liste, die aus Abonnenten von Usenet Gruppen ausgesucht oder von Firmen bezogen werden, die sich auf E-Mail-Vertrieb spezialisiert haben. Für den Empfänger ist es im Normalfall lediglich Müll. Im Allgemeinen gilt es nicht als guter Ton, Spam zu versenden. Es ist üblicherweise vergleichbar mit unverlangten Werbetelefonaten, nur zahlt der Empfänger einen Teil der Kosten mit, da jeder Benutzer die Aufrechterhaltung des Internets mitträgt.
Einige scheinbar unverlangte E-Mails sind tatsächlich E-Mails, deren Empfang man zugestimmt hat, als man sich bei einer Site registriert hat und dort ein Kreuz in ein Kästchen gesetzt hat, mit dem man die Zustimmung zur Zusendung von Informationen über bestimmte Produkte oder Interessen gab. Dies ist bekannt als **OPT-IN E-Mail** oder **Permission-Based E-Mail** (genehmigungsbasierte E-Mail).
Ein Bericht aus erster Hand gibt an, daß der Ausdruck Spam einem bekannten Monty Python Sketch ('Well, we have Spam, tomato & Spam, egg & Spam, Egg, bacon & Spam...') entnommen wurde, der aktuell war, als Spam das erste mal im Internet auftauchte. Spam ist ein eingetragenes Warenzeichen der Firma Hormel Foods und bezeichnet ein Fleischprodukt, das bei den US-Streitkräften im 2. Weltkrieg sehr bekannt war.
Ein kommerzieller Spammer führt eine Datenbank mit teilweise mehreren Millionen Adressen. Diese kann er, z.B. durch das gezielte (mit einem Programm automatisierte) Absuchen von Newsgroups, Homepages oder E-Mail-Verzeichnissen, aber auch durch Durchprobieren gängiger Adressen, erhalten. Das Versenden der E-Mails funktioniert ebenfalls automatisch. Da der Versand von E-Mails fast nichts kostet, spielt es keine Rolle, wenn viele Adressen ungültig sind.
Programme zum Auslesen von Newsgroups sind sehr einfach zu schreiben und sehr ergiebig.
Um nicht viele Fehlermeldungen wegen ungültiger Adressen oder gehässige Antworten zu erhalten, setzt der Spammer eine falsche Rückantwortadresse ein, die es zusätzlich schwierig macht, seine Identität herauszufinden. Kommuniziert werden kann nur per Post oder Fax. Da so für Beschwerden Kosten anfallen, erhoffen sich Spammer, weniger Negativreaktionen zu erhalten.
Zusätzlich verwendet ein erfahrener Spammer nicht den Mailserver seines Providers, sondern einen anderen, ungeschützten Mailserver. Damit erschwert der Spammer Gegenmaßnahmen, da ungeschützte Mailserver oft keinen Administrator haben oder einen, dem Reklamationen egal sind.

Trotzdem hinterlassen Spammer ihre Spuren. Außerdem müssen Spammer irgendwo innerhalb der Mail auch eine Kontaktadresse angeben, damit ihre Produkte wenigstens theoretisch gekauft werden können.
Im Internet gibt es Listen mit Maildomains und IP-Adressen, die als Spammer bekannt sind. Diese Listen können in Mailsysteme integriert werden, so daß diese Domains dann geblockt werden. Solche Listen werden unter `http://www.spam.org/` [spa 02] und `http://spam.abuse.net/` [abu 02] gepflegt.

Relaying

Wenn nun für den E-Mailversand nicht der eigene Provider, sondern ein fremder Rechner benutzt wird, so spricht man von **Relaying**. Im ersten Schritt ist da noch nichts Unehrenhaftes dabei. Leider werden diese Relays aber, wie oben erwähnt, oft für Spam Mails mißbraucht, um den Mailweg nicht so leicht zurück verfolgen zu können. Es gibt kommerzielle Listen, in denen alle Mailsysteme aufgelistet sind, über die Spam verschickt wird oder die als **Open Relay** verwendet werden können. Wird nun ein System als Relay mißbraucht, so ist die Wahrscheinlichkeit hoch, daß auch dieses System auf eine dieser Listen wandert. Sobald ein System auf einer der Listen vermerkt ist, werden E-Mails, die von diesem System weiterverschickt werden, von den Empfangsmaschinen abgelehnt, wenn diese mit den Relaying-Listen arbeiten. Dadurch können auch alle berechtigten User nicht mehr zu ihren Kommunikationspartnern mailen.
Mailserver können so konfiguriert werden, daß sie anhand der Informationen zu Absender und Empfänger das Relaying erschweren können. Folgende Checks sind dann nötig:

- Der sendende MTA ist ein internes System, das den Mailserver als Relay verwenden darf.
- Der sendende MTA steht nicht auf der Liste der Systeme, die den Mailserver als Relay verwenden dürfen:
  - Absenderdomain ist extern, Emfängerdomain intern.

Wird nun versucht, von einer externen Absenderdomain an eine externe Emfängerdomain Mails zu verschicken, so wird das geblockt. Ebenso ist es nicht erlaubt, von einem MTA aus Mails mit interner Absenderdomain an externe Emfängerdomains zu versenden, wenn der MTA nicht als genehmigter Relay eingetragen ist.

Fälschen von E-Mail-Absenderadressen

Ein weiteres Problem, das in diesem Zusammenhang nicht vergessen werden darf, ist die relativ einfache Möglichkeit des Fälschens von Absenderadressen beim E-Mail-Verkehr. Somit wird es erschwert, Spamlisten zu führen. Außerdem besteht die theoretische

Möglichkeit, daß von extern eine Mail mit lokaler Absenderadresse wieder an extern geschickt wird. Das kann im Mailserver durch die Option 'Relaying' oder 'Relay' unterbunden werden.
Um generell einen Check zu machen, ob die Absenderadresse von dem sendenden Mailserver aus versendet werden darf, kann eine sogenannte **DNS Blackliste** verwendet werden. Hier wird anhand der im Internet bekannten MX Records überprüft, ob der MX für diese Absenderdomain auf den sendenden Mailserver zeigt. Das ist in der Praxis aber nicht empfehlenswert, da es oft für ausgehende Mails einen Mailrelay gibt, der nicht im MX verzeichnet ist.

Mailbomb

Mailbomb kann etwa mit Postlawine übersetzt werden.
Als Mailbomb bezeichnet man eine große Menge von E-Mails an ein und dieselbe Person, um ihr zu schaden oder sie in ihrer Arbeit zu behindern. Dies kann z.B. dadurch geschehen, daß der Speicherplatz der Mailbox überfüllt wird und der Angegriffene keine Mails mehr empfangen kann. In Extremfällen kann auch der Mailserver zum Absturz gebracht werden. Solche 'Strafaktionen' werden gegenüber Personen angewandt, die sich irgendwie den Unmut des Absenders der Mailbomb zugezogen haben.
Mailbombs schaden nicht nur dem Ziel, sondern dem gesamten Mailsystem und all seinen Benutzern.
An Mailservern können von Hand auch einzelne Absender gesperrt werden.

Verschlüsselung

E-Mails gehen standardmässig unverschlüsselt über das Netz. D.h. Mails können sowohl während der Übertragung, wie auch auf einem der Mailserver, die sie passieren, mitgelesen werden. Somit sollten sensitive Daten verschlüsselt werden. Eine geeignete Möglichkeit hierfür ist die Verschlüsselung der Mail auf dem Client. Dies kann mit z.B. PGP ([pgp 02]) bzw. GnuPGP ([gnu 02]) erfolgen. Das wird hier aber nicht weiter behandelt, da es sich um Clientsicherheit handelt und keine Konfiguration am Mailrelay erfordert.
Weitere Informationen und Links können unter [boe 02] gefunden werden.

7.1.4 Die Software „Sendmail“

Es gibt nun verschiedene Möglichkeiten, einen Mailserver zu implementieren. Hier soll das Softwarepaket Sendmail ([sen 02]) vorgestellt werden.
Da es nun aber mittlerweile im Netz sehr viele Personen und Firmen gibt, die potenzielle Kunden ungefragt mit Mails überhäufen, und diese Mailsysteme, von denen der Spam ausgeht, oft als Kommunikationspartner gesperrt sind, wird die Funktion des Mailrelays

mißbraucht. Um also freie Mailkommunikation für die eigenen Anwender garantieren zu können, sollte man sein eigenes Mailsystem davor schützen, als Mailrelay von Unbefugten verwendet zu werden.
Sendmail ist ein sehr mächtiger und weit verbreiteter Mailserver mit unzähligen, verschiedenen Konfigurationsmöglichkeiten. Hier sollen nur die wichtigsten Elemente vorgestellt werden, damit Mailzustellung und **Anti-Relaying** funktioniert.
Die Konfigurationsfiles findet man unter `/etc/mail/`, das Startscript heißt `/etc/init.d/sendmail`. Zum automatischen Start des Dienstes werden durch das Konfigurationstool `YaST` Softlinks aus den Runlevel Verzeichnissen `/etc/init.d/rc*.d/` auf das Startfile gelegt. Aus `/etc/mail/sendmail.mc` wird das eigenliche Konfigurationsfile `/etc/sendmail.cf` mittels m4 Macros generiert (Informationen zu m4 sind unter [mac 02a] zu finden). In der `sendmail.mc` werden die verschiedenen Optionen schnell und übersichtlich eingebettet. Die Entsprechungen in der `sendmail.cf` sind um so kryptischer.
Die `sendmail.mc` könnte beispielsweise so aussehen:

```
divert(0)dnl
include(`/usr/share/sendmail/sendmail.cf/m4/cf.m4')dnl
VERSIONID(`$Id: sendmail.mc, v 8.12.0.Beta19 2001/05/29 12:00:00 cowboy Exp $')
OSTYPE(`linux')dnl
LOCAL_CONFIG
Cwsecserver.nm.informatik.uni-muenchen.de
FEATURE(always_add_domain)dnl
FEATURE(accept_unresolvable_domains)dnl
FEATURE(mailertable)dnl
FEATURE(genericstable)dnl
FEATURE(access_db)dnl
GENERICS_DOMAIN_FILE(`/etc/mail/generics-domains')dnl
define(`SMART_HOST', `mail-relay.test.de')
define(`confSMTP_LOGIN_MSG', `Sendmail --- Internet Mail Gateway')dnl
define(`confPRIVACY_FLAGS', `authwarnings,noexpn,noetrn,novrfy,needmailhelo,
       restrictmailq,restrictqrun')dnl
define(`confTO_QUEUEWARN', `4h')dnl
define(`confTO_QUEUERETURN', `12h')dnl
define(`confQUEUE_SORT_ORDER', `Priority')dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

Nun zur Erklärung der hier aufgeführten Optionen:

- `dnl` heißt **delete until next new line** und kann als Kommentarzeichen verstanden werden.

- `divert(0)`: sorgt anhand der Nummerierung dafür, daß der Konfigurationsinput in unterschiedliche Teile aufgeteilt werden kann, um Reihenfolgeprobleme auszuschließen, wobei aber die Optionen übersichtlich angeordnet werden können.
- `include('/usr/share/sendmail/sendmail.cf/m4/cf.m4')` gibt an, wo das m4 Programm zu finden ist.
- `VERSIONID` gibt die Versionsnummer des verwendeten Sendmail an.
- `OSTYPE` setzt fest, welches Betriebssystem dem Server zugrundeliegt.
- Mit `Cw` wird der vollqualifizierte Hostname des Mailservers angegeben, den er bei selbst generierte Mails im Absender stehen haben soll.
- `FEATURE(always_add_domain)` sorgt dafür, daß bei jeder Mail, bei deren Absender keine Domain angegeben ist, die eigene Domain angehängt wird.
- `FEATURE(accept_unresolvable_domains)` wird angegeben, wenn Mails auch dann angenommen werden sollen, obwohl zur Zeit keine DNS Auflösung für die Absenderdomain möglich ist.
- `FEATURE(mailertable)` zeigt Sendmail an, daß es eine Tabelle `mailertable` (siehe auch Seite 193) gibt, in der definiert ist, welche Empfängerdomains zu welchem Mailserver geschickt werden.
- `FEATURE(genericstable)` zeigt Sendmail an, es existiert eine Tabelle `genericstable` mit Mailadressen (Seite 193), die zur Umsetzung speziell definierter User und inoffizieller Domains dient.
- `FEATURE(access_db)` zeigt Sendmail an, daß es eine Datenbank `access_db` gibt, mit deren Hilfe festgelegt wird, welche IP-Adressen den Mailserver als Relay verwenden dürfen, welche User explizit erlaubt sind und was geblockt werden soll. Das Konfigurationsfile dazu heißt `access` (siehe auch Seite 192).
- `GENERICS_DOMAIN_FILE` gibt an, wo das File mit den lokal zu behandelnden Domains zu finden ist, die mit Hilfe der `genericstable` behandelt werden.
- `define('SMART_HOST', '[10.10.10.10]')` wird verwendet, wenn keine direkte Verbindung zum Internet besteht. Damit werden alle Mails, deren Empfängerdomains nicht explizit in der `mailertable` angegeben sind, zu dem hier angegebenen Mailserver geschickt.
- `define('confSMTP_LOGIN_MSG', '... etc ...')` legt die Meldung fest, die bei Verbindungsbeginn angezeigt wird.
- `define('confPRIVACY_FLAGS', '....')` setzt Sicherheitsbedingungen zum Mailaustausch:

  - `authwarnings`: Überprüfung des bei `HELO` angegebenen Servernamens auf Gültigkeit. Bei Problemen wird ein `X-Header` mit der Mail mitgeschickt, in dem die Warnungen bezüglich der entdeckten Unstimmigkeiten aufgelistet sind. Die Mail wird aber ganz normal weiter behandelt.
  - `noexpn`: Es wird keine Expansion von 'Aliases' erlaubt.
  - `noetrn`: `etrn` ist das remote Anstoßen von Queueläufen und wird in RFC 1985 [Wint 96] beschrieben. Damit könnte bewirkt werden, daß die Mailschlange nicht in der normalen Reihenfolge abgearbeitet wird, sondern E-Mails von bestimmten Absendern oder für bestimmte Empfänger sofort zugestellt werden. Dies wird hier unterbunden.
  - `novrfy`: Keine Validierung von lokalen Adressen erlaubt.
  - `needmailhelo`: `HELO` wird unbedingt am Anfang der Mailverbindung verlangt.
  - `restrictmailq`: Diese Option hat nur lokal auf dem Mailserver Bedeutung. Nur User, die in der gleichen Gruppe wie Sendmail sind, können den Befehl `mailq` ausführen.
  - `restrictqrun`: Auch diese Option hat nur lokal auf dem Mailserver Bedeutung. Nur User, die in der gleichen Gruppe wie Sendmail sind, können den Befehl `qrun` ausführen.

  Nimmt man `authwarnings`, `noexpn`, `noetrn` und `novrfy` zusammen, so kann auch die Option `goaway` gesetzt werden.

- `define('confTO_QUEUEWARN', '4h')` legt fest, daß eine Warnmeldung an den Absender geschickt wird, wenn eine Mail vier Stunden lang nicht versandt werden konnte.
- `define('confTO_QUEUERETURN', '12h')` sorgt dafür, daß eine Mail, die für zwölf Stunden nicht verschickt werden konnte, an den Absender zurückgeht.
- `define('confQUEUE_SORT_ORDER', 'Priority')` gibt an, daß die Mails ihrer Priorität nach bearbeitet werden.
- `MAILER` gibt die zur Verfügung stehenden Mailer an. Standardmäßig wird nur `smtp` verwendet.

In der Datei `/etc/aliases`, aus der mit `newaliases` die dazugehörige Datenbank generiert wird, stehen **Aliase**, an die Mails verschickt werden. Um nicht für jeden Namen eine Mailadresse angeben zu müssen, werden alle Systemmails an `root` geschickt, der die Mails dann an ein definiertes Postfach bekommt. Ist das nicht vorhanden, so werden die Mails lokal unter `/var/spool/mail` und dem jeweiligen Benutzernamen abgelegt.

```
#
# Mail aliases for sendmail
#
# You must run newaliases(1) after making changes to this file.
#

# Required aliases
postmaster:     root
MAILER-DAEMON:  postmaster

# Common aliases
abuse:          postmaster
spam:           postmaster

# Other aliases

root: administrator@[192.168.5.5]
webmaster: root
```

Das `access` File kann nun folgendermaßen aussehen:

```
192.168.1.1                     RELAY
192.168.2                       RELAY
From:netteruser@testdomain.com  OK
From:testdomain.com             550 Relaying denied for your account
```

D.h. die `192.168.1.1` und das Netz `192.168.2.0/255.255.255.0` sind erlaubt, den Mailserver als Relay zu verwenden. Mails von `netteruser@testdomain.com` sind ebenfalls erlaubt, die restliche Domain `testdomain.com` darf allerdings keine Mails über diesen Mailserver schicken.
In der `generics-domains` werden alle Domains aufgelistet, deren Usermails nicht mit dem ursprünglichen Namen und der ursprünglichen Domain weitergegeben werden sollen.

```
rechner.inoffizielle-domain.de
rechner
```

Die Umsetzung erfolgt dann in der `genericstable`. Hier ein Beispiel für die `genericstable`. Die erste Spalte beinhaltet die umzusetzenden Informationen, in der zweiten sind die offiziellen Informationen enthalten.

```
testuser@rechner.inoffizielle-domain.de       userid@lokale.domain1.de
testuser@rechner                              userid@lokale.domain1.de
```

In der `mailertable` wird festgelegt, wenn für bestimmte Domains nicht der Mailserver, der durch den MX Records im DNS festgelegt ist, verwendet werden soll, sondern es andere MTAs gibt, die die Mail erhalten sollen:

```
partner1.de              smtp:[10.10.10.10]
partner2.com             smtp:[172.16.2.2]:[192.168.100.1]
partner3.de              smtp:mailserver.partner3.de
```

`relay-domains` beinhaltet die Liste aller Domains, die als lokale Domains betrachtet werden sollen. Diese Datei, zusammen mit der Datei `access`, sind für die **Anti-Relayingkonfiguration** unerlässlich.

```
lokale.domain.de
lokale.domain.com
lokale.domain.org
```

Nun müssen aber aus den hier aufgezeigten Files das Konfigurationsfile `/etc/sendmail.cf` und alle Datenbankfiles erzeugt werden. Die Aliasdatenbank wird mit `newaliases` geschrieben. Bei der `sendmail.cf` begibt man sich zuerst nach `/etc/mail/` und ruft dann die m4 Macros so auf:
`/usr/bin/m4 sendmail.mc > /etc/sendmail.cf`
Die Datenbanken werden mittels `makemap` erzeugt, z.B.:
`makemap hash mailertable < mailertable`

Bei SuSE wird aber im Verzeichnis `/etc/mail` bereits ein `Makefile` mitgeliefert, daß diese Generierungen übernimmt.

Informationen zu m4 Macros finden Sie unter [mac 02b]. Generelle Sendmailinformationen sind unter [sen 02] erhältlich. Ein Sendmail HowTo ist unter [www 02] zu finden. Viele der hier dargestellten Informationen sind in [Cost 97] nachzulesen.

7.2 World Wide Web

WWW wurde 1989 im **CERN** (dem **Europäischen Kernforschungszentrum in Genf, Conseil Européen pur la Récherche Nucléaire**) entwickelt, basierend auf einem System namens Hypertext.
WWW ist der Versuch, die gesamte Information im Internet zusammenzufassen und über ein einziges Benutzerinterface zugänglich zu machen. Für den Benutzer existieren Programme verschiedener Hersteller, **Browser** genannt, die das WWW verfügbar machen. Die ausgewählten Wörter sind durch Farbe oder Unterstreichung hervorgehoben und können per

Mausklick expandiert werden. Damit beginnt die Reise durch das WWW.
Damit Seiten, die öfter aufgerufen werden, nicht immer über das Netz transportiert werden müssen, können die meisten Browser WWW-Seiten lokal zwischenspeichern (**Cache-Speicherung**). Es erfolgt dann nur eine kurze Anfrage an den Server, ob sich die entsprechende Information seit dem letzten Zugriff geändert hat. Ist dies nicht der Fall, werden die Daten lokal von der Platte geholt. Die größeren Provider und Uni-Rechenzentren unterhalten ebenfalls ein Cache-System. Wenn ein Benutzer eine WWW-Seite anfordert, wird die Info auf der Platte des Providers zwischengespeichert. Bei der Anfrage eines weiteren Benutzers nach derselben Seite innerhalb eines bestimmten Zeitraums wird die lokale Kopie zur Verfügung gestellt (**Proxy-Cache**, **Proxy-Server**, vgl. Kap. 8). Die Proxy-Software überprüft regelmäßig, ob sich die lokal gespeicherten Infos eventuell geändert haben und aktualisiert sie gegebenenfalls. Nicht mehr gefragte Seiten werden nach einiger Zeit gelöscht.
Die Darstellung einer Webseite erfolgt erst im Browser. Da werden die verschiedenen Teile, die übertragen wurden, zu einem Ganzen zusammengesetzt. Grundlage für eine Webseite ist eine Textdatei, die die Strukturierung der Seite vorgibt. Die verwendete Stuktursprache heißt **HTML** (**HyerText MarkUp Language**).
URL ist die Abkürzung für **Uniform Resource Locator** und wird im Netz verwendet, um Informationen vollständig zu bezeichnen. Mit einer URL wird nicht nur eine Datei und das zugehörige Verzeichnis, sondern auch der Rechner festgehalten, auf dem sie zu finden ist.

Protokoll://Rechneradresse:Port/Dateipfad/Dateiname

Eine URL besteht also aus vier Teilen, wobei nicht immer alle Teile aufgeführt werden müssen (meist ist z. B. keine Portangabe nötig). Hier nun zwei Beispiele:

- ftp - Dateitransfer mittels (anonymen) FTP: z. B. `ftp://ftp.microsoft.com/pub/drivers/mouse/mouse.sys`
- http - Lesen von WWW-Seiten (http HyperText Transport Protocol): z. B. `http://www.uni-muenchen.de/`

Solche Standard-Ports müssen nicht angegeben werden. Man kann aber auch unbelegte Portnummern verwenden, beispielsweise um einen modifizierten WWW-Dienst anzubieten. In diesem Fall muß dann die Portnummer angegeben werden, z.B. `http://www.blablub.de:8000/index.html`.

7.2.1 HTTP - Hypertext Transfer Protokoll

HTTP ist ein Protokoll der Applikationsschicht, das alle Möglichkeiten der Übertragung von Hypermedia-Informationen bietet. Seit 1990 ist dieses Protokoll im Einsatz und wird

derzeit meist in der Version `HTTP/1.0` verwendet.
Heutige Informationssysteme benötigen weit mehr Funktionen als das einfache Senden und Empfangen von Nachrichten. Die Entwicklung von `HTTP/1.0` ist nicht abgeschlossen. Es bietet die Möglichkeit, weitere Funktionalität zu entwickeln. Die Adressierung von Ressourcen erfolgt dabei mittels **URIs** (**Uniform Resource Identifier**), die Orte (**URL**) oder Bezeichner (**URN**) sein können. Diese zeigen gleichzeitig den gewünschten Übertragungsmechanismus an. Nachrichten werden in der gleichen Form übertragen, wie sie auch bei normalem Mail-Transport verwendet werden. Dabei kommt oft MIME zum Einsatz. HTTP/1.0 ist auch für den Zugriff auf Server mit anderen Protokollen geeignet. So ist es WWW-Clients möglich, mit Servern und Gateways per SMTP, NNTP, FTP, Gopher und WAIS zu kommunizieren.

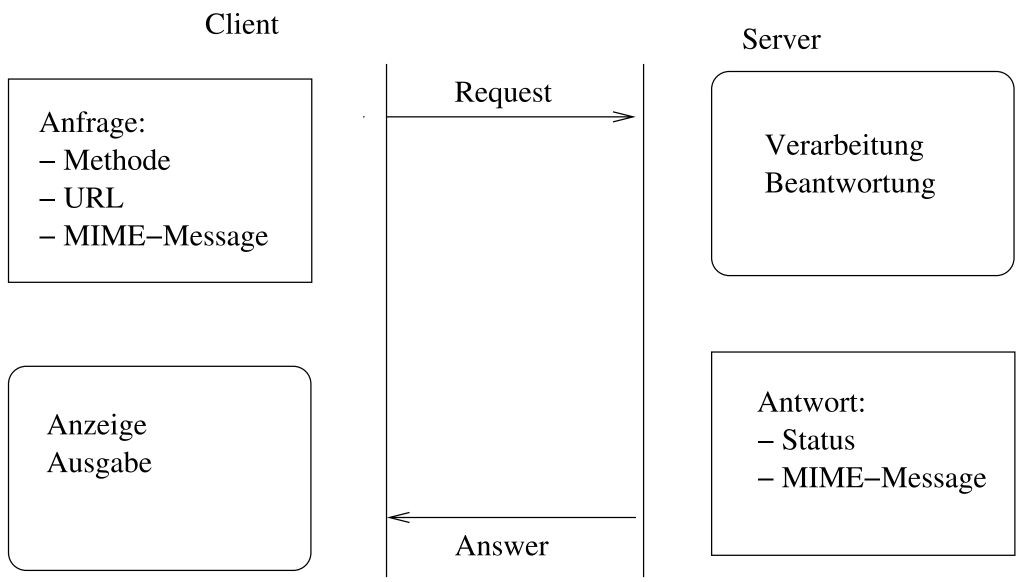

Abbildung 66: Prinzipielle Client-Server Funktionsweise einer HTTP Anfrage

Direkt nach Beantwortung der Frage wird die Verbindung wieder abgebaut, d.h. HTTP ist ein zustandsloses Protokoll. Um aber z.B. das Einkaufen auf Shoppingseiten zu ermöglichen (Warenkorb), muß nun künstlich ein **Verbindungszustand** hergestellt werden. Das kann mit Hilfe von sogenannten **Cookies** geschehen. Da in diesen Cookies nicht nur das Benutzerverhalten wiedergespiegelt wird, sondern auch personenbezogene Daten gespeichert werden können, sind sie mit Vorsicht zu genießen. Informationen zur Funktionsweise von Cookies und Einstellungsmöglichkeiten der Browser finden Sie im Anhang A.5.3.
Derzeit finden HTTP-Verbindungen meist per TCP/IP statt. Das soll aber nicht heißen, daß HTTP nicht auch auf anderen Netzwerkprotokollen aufsetzen kann. Beide Seiten müssen auch dazu in der Lage sein, auf den vorzeitigen Abbruch der Kommunikation

durch die andere Seite zu reagieren. Vorzeitiger Abbruch kann durch Aktionen von Benutzern, Programmfehler oder Überschreiten der Antwortzeiten ausgelöst werden. Durch den Abbruch der Verbindung durch eine der beiden Seiten wird der gesamte Vorgang abgebrochen.
Jede Kommunikation zwischen zwei WWW-Programmen besteht aus HTTP-Botschaften, die in Form von Anfragen und Antworten zwischen Client und Server ausgetauscht werden. Jedes der Felder eines HTTP-Header weist die gleiche Struktur auf. Im RFC 822 [Croc 82] [41] wurde definiert, daß jedes Feld mit einem Feldnamen und dem Feldinhalt erscheint. Auf den Feldnamen muß unbedingt ein Doppelpunkt folgen. Der Feldname kann alle Zeichen außer dem Doppelpunkt und der Escape-Sequenzen enthalten. Allgemeinfelder enthalten Informationen wie das Datum, die Message-ID, die verwendete MIME-Version und ein `forwarded`-Feld, das angibt, ob das Dokument eigentlich von einer anderen Adresse stammt.
Bei Anfragen wird zwischen einfachen und komplexen Anfragen unterschieden. Eine einfache Anfrage besteht nur aus einer Zeile, die angibt, welche Information gewünscht wird. Ein Beispiel:
`GET /index.html`
Dabei wird nur die Methode (`GET`) und die URL des Dokumentes angegeben. Es werden keine weiteren Felder erwartet und vom adressierten Server wird auch nur ein ganz einfacher Antwortkopf zurückgesendet. Es kann aber auch eine komplexere Anfrage erzeugt werden. Dabei muß die Zeile aus dem obigen Beispiel noch die Version des HTT-Protokolls angehängt werden. In einem Beispiel würde das folgendermaßen aussehen:
`GET /index.html HTTP/1.0`
Die Anfügung der HTTP-Version ist also der ganze Unterschied zwischen einer einfachen und einer komplexen HTTP-Anfrage. Der Unterschied zwischen einfacher und komplexer Anfrage wird aus Gründen der Kompatibilität gemacht. Ein Browser, der noch das alte `HTTP/0.9` implementiert hat, wird nur eine einfache Anfrage losschicken können. Ein neuer Server muß dann eine Antwort, auch im Format des `HTTP/0.9` zurücksenden.
Um die Anfrage näher zu spezifizieren, wurden weitere Felder eingeführt. In den Anfragefeldern stehen z.B. Informationen über den Server und den benutzten Browser. Weiterhin kann man dort Informationen über den Gegenstand der Übertragung bekommen. In der folgenden kurzen Übersicht sind alle möglichen Felder einer Anfrage aufgeführt.

- **Anfragezeile (Request-Line):** Informationsanfrage wie oben geschildert. Die zugehörigen **Methoden** folgen im nächsten Abschnitt.
- **Allgemeiner Kopf (General-Header):** Im allgemeinen Kopf werden allgemeine Informationen über die Nachricht übermittelt.
- **Anfragekopf (Request-Header):** In diesen Feldern kann der Browser weitere Informationen über die Anfrage und über den Browser selbst absetzen. Diese Felder

[41] neu RFC 2822 [ReEd 01]

sind optional und müssen nicht erscheinen.

- **Gegenstandskopf (Entity-Header):** In diesem Feld werden Einträge übermittelt, welche den Inhalt der Nachricht näher beschreiben.
- **Gegenstand der Nachricht (Entity-Body):** Vor dem eigentlichen Inhalt muß definitionsgemäß eine Leerzeile stehen. Der Inhalt ist dann in dem Format codiert, das in den Gegenstandsfeldern definiert wurde (meist HTML).

Das an erster Stelle in einer Anfragezeile (Request-Line) stehende Wort beschreibt die Methode, die mit der nachfolgenden URL angewendet werden soll. Die Methodennamen müssen dabei immer groß geschrieben werden. Der Entwurf des HTTP-Standards erlaubt leicht eine Erweiterung. Kommen wir nun zur Bedeutung der einzelnen Methoden.

- `GET`: Diese Methode gibt an, daß alle Informationen, die mit der nachfolgenden URL beschrieben werden, zum anfragenden Client geholt werden sollen. Zeigt die URL auf ein Programm (z.B. CGI-Script), dann soll dieses Programm gestartet werden und die produzierten Daten liefern. Handelt es sich bei dem referenzierten Pfad um eine Datei, dann soll diese übertragen werden. Beispiel: `GET http://www.netzmafia.de/index.html`
- `HEAD`: Diese Methode ist identisch zur Methode `GET`. Die Antworten unterscheiden sich nur darin, daß bei der Methode `GET` ein komplettes Dokument übertragen wird und bei `HEAD` nur die Meta-Informationen gesendet werden. Dies ist nützlich, um Links auszuprobieren oder um die Erreichbarkeit von Dokumenten zu testen. Bei Anwendung der Methode `HEAD` wird der Kopf des referenzierten HTML-Dokuments nach `link` und `meta` Elementen durchsucht.
- `POST`: Diese Methode wird zur Übertragung von Daten vom Client zum Server verwendet. Man stelle sich vor, ein HTML-Dokument enthält ein komplexes Formular. Per `POST` wird dem Server angezeigt, daß er auch die Daten im Körper der Botschaft bearbeiten soll. Die wirkliche Funktion, die durch `POST` auf dem adressierten Rechner angestoßen wird, wird durch die URL bestimmt. Meist sind es CGI-Scripte, die den Inhalt der Nachricht verarbeiten.
- `PUT`: Die mit der Methode `PUT` übertragenen Daten sollen unter der angegeben URL gespeichert werden. Das soll ermöglichen, daß WWW-Seiten auch ohne direkten Zugriff auf den anbietenden Rechner erstellt werden können. Wird ein Dokument mit der Methode `PUT` übertragen, so wird unter dieser Adresse ein Dokument mit dem übertragenen Inhalt angelegt. War die Aktion erfolgreich, wird die Meldung `200 created` zurückgegeben. Existiert unter dieser Adresse schon ein Dokument, wird dieses überschrieben. War auch diese Aktion erfolgreich, wird nur `200 OK` zurückgemeldet. Der Hauptunterschied zwischen `POST` und `PUT` besteht darin, daß bei `POST` die URL eine Adresse eines Programmes referenziert, das mit den Daten umgehen kann. Bei `PUT`

hingegen wird die URL als neue Adresse des Dokumentes gesehen, das gerade übertragen wurde. Meist jedoch ist die Methode `PUT` ausgeschaltet, weil Server-Betreiber befürchten, daß die Sicherheit des Systems dadurch nicht mehr gewährleistet ist.

- `DELETE`: Mit dieser Methode kann der Inhalt einer URI gelöscht werden. Diese Methode ist neben der Methode `PUT` eine der Gefährlichsten. Wenn Server nicht richtig konfiguriert wurden, kann es mitunter vorkommen, daß alle Welt die Berechtigung zum Löschen von Ressourcen hat.

- `LINK`: Mit dieser Methode können eine oder mehrere Verbindungen zwischen verschiedenen Dokumenten erzeugt werden. Es werden dabei keine Dokumente erstellt, sondern nur schon bestehende miteinander verbunden.

- `UNLINK`: entfernt Verbindungen zwischen verschieden Ressourcen. Dabei wird nur die Verbindung gelöscht. Die Dokumente existieren trotzdem weiter. Mit diesen Methoden kann man alle möglichen Ressourcen erreichen, welche die verschiedenen Server zur Verfügung stellen.

`DELETE` und `PUT` sollten aus Sicherheitsgründen defaultmäßig deaktiviert sein (ist bei SuSE Grundkonfiguration).
Möchte man zu Testzwecken HTTP von Hand sprechen, so sieht das ungefähr so aus:

```
telnet www.netzmafia.de 80
Trying 141.39.253.210...
Connected to www.netzmafia.de.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 18 Sep 2000 13:59:58 GMT
Server: Apache/1.3.6 (Unix)  (SuSE/Linux)
Last-Modified: Tue, 29 Aug 2000 08:08:58 GMT
ETag: "134015-8e8-39ab6f9a"
Accept-Ranges: bytes
Content-Length: 2280
Connection: close
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Netzmafia</TITLE>
</HEAD>

<body bgcolor="#000000" text="#FFFFCC" link="#FFCC00"
```

```
        alink="#FF0000" vlink="#FF9900">

        ...


</BODY>
</HTML>
Connection closed by foreign host.
```

Da auch hier die Übertragung der Daten, wie auch bei Telnet und FTP, unverschlüsselt erfolgt, sollte man sich bei sensitiven Daten Gedanken zur Verschlüsselung machen.

7.2.2 SSL - Secure Socket Layer

Dieses Protokoll mit kommerziellem Hintergrund erlangte seine Bedeutung durch die große Verbreitung des Web-Browsers Netscape Navigator. Dieser Client beherrscht ein Protokoll namens **Secure Socket Laver** (**SSL**). Wie der Name andeutet, ist es nicht nur für HTTP vorgesehen, sondern soll jedes zuverlässige Transportprotokoll um ein Konzept für einen sicheren Kanal (Vertraulichkeit, Authentifikation, Datenintegrität) erweitern.
Eine gesicherte Verbindung über dieses Protokoll erkennt man im Browser daran, daß die dargestellte URL mit dem Kürzel `https://` beginnt; zusätzlich erscheint in der Statusleiste des Browsers ein eingerastetes Vorhängeschloß. SSL entstand ursprünglich 1994 als proprietäre Lösung von Netscape für ein verbindungsorientiertes Sicherungsprotokoll auf der Datenschicht. Schon ein Jahr später wurde es bei der **IETF** (**Internet Engineering Task Force**) zur Normung eingereicht, heute dient es in der Version 3.0 als Arbeitsbasis der **Transport Layer Security (TLS) Working Group** der IETF. Der ursprüngliche Zweck von SSL bestand lediglich in der Sicherung der Kommunikation zwischen Web-Server und Browser.
Im OSI-Schichtenmodell der Datenkommunikation residiert SSL zwischen der Transportschicht (TCP oder UDP) und der Anwendung. Dabei stellt der SSL-Layer für die Applikation statt der normalen Socket-Funktionen wie `read` oder `write` spezielle Methoden zur Eröffnung und Nutzung einer sicheren Tansportverbindung zur Verfügung. Die Anwendung reicht die Daten, anstatt sie direkt an die Transportschicht zu übergeben, also zunächst an SSL weiter. Dieses schützt Verbindung und Daten durch die Bearbeitung mit kryptographischen Verfahren und übermittelt erst anschließend die Daten an die Transportschicht.

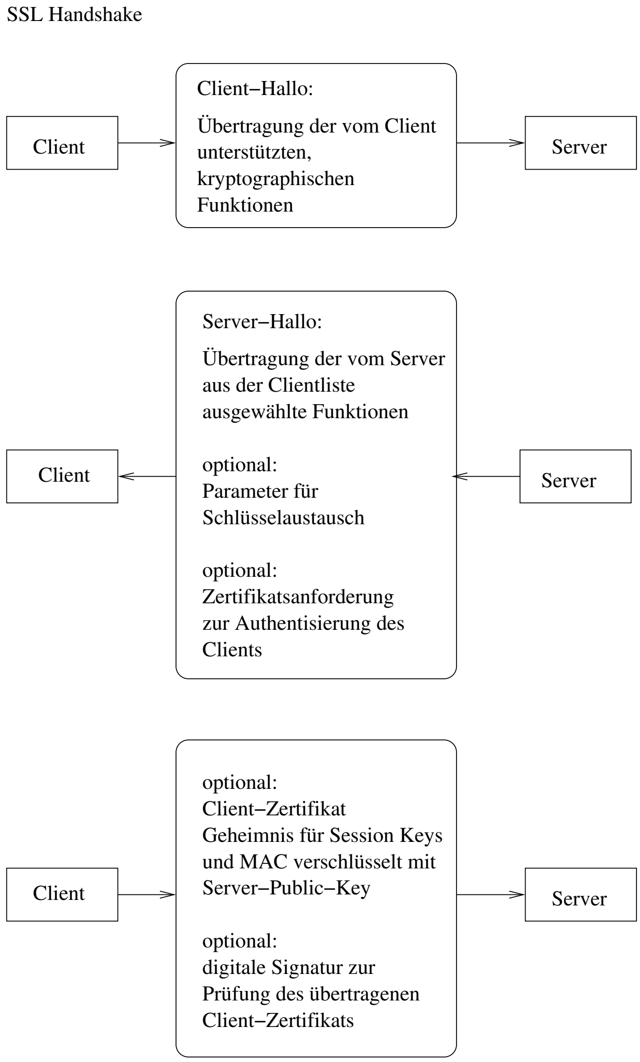

Abbildung 67: Prinzipielle Funktionsweise des Handshakes von SSL

Insgesamt kombiniert SSL fünf verschiedene Protokolle:

1. Das **SSL Application Data Protocol** wickelt die Datenübermittlung zwischen Anwendung und SSL ab.
2. Das **SSL Alert Protocol** dient der Weiterleitung von Warn- und Fehlermeldungen.
3. Das **SSL Change Cipher Spec Protocol** übernimmt die Initialisierung der festgelegten, kryptographischen Verfahren. Um über SSL eine gesicherte Verbindung aufzubauen, müssen sich die Kommunikationspartner zunächst einmal über die zu

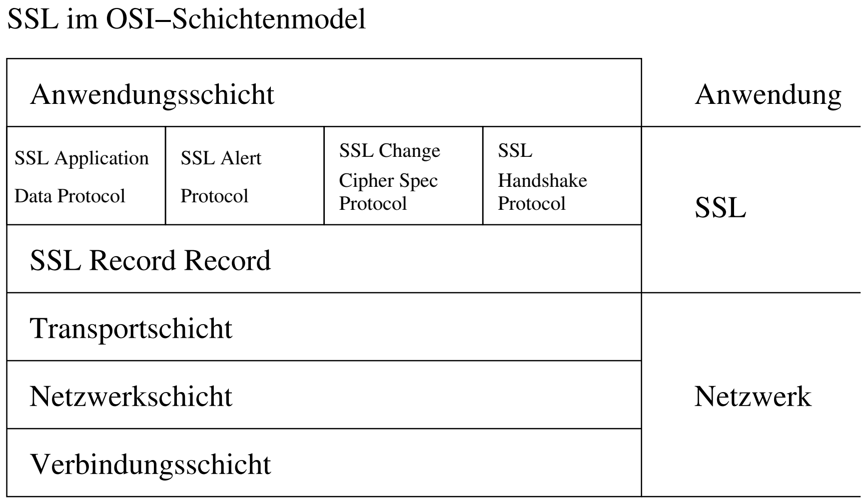

Abbildung 68: Prinzipielle Funktionsweise von SSL

verwendenden kryptographischen Verfahren und Parameter einig werden. Grundsätzlich bietet SSL dabei Schlüsselaustauschverfahren, eine symmetrische Verschlüsselung sowie die Berechnung einer kryptographischen Prüfsumme als Möglichkeit an. Für jede dieser Möglichkeiten lassen sich verschiedene Verfahren nutzen. Etwa RSA oder Diffie-Hellmann für den Schlüsselaustausch, DES oder IDEA für die Verschlüsselung sowie MD5 oder SHA für die Prüfsumme. Als kleiner Haken erweist sich dabei, daß die meisten SSL-relevanten Produkte aus den USA stammen und daß aufgrund der dort geltenden Exportbeschränkungen nur bestimmte Schlüssellängen genutzt werden können.

4. Über das **SSL Handshake Protocol** handeln Server und Client das zu verwendende, kryptographische Verfahren aus und authentisieren sich gegebenenfalls gegenseitig. Vor der Übertragung der eigentlichen Daten arbeiten Client und Server ein Handshake-Protokoll ab, in dem der Sitzungsschlüssel ausgetauscht und die Authentifikation vorgenommen wird. Der Client eröffnet den Handshake mit einem Einmalwert (**challenge**), der Liste der unterstützten Verschlüsselungsverfahren und, sofern vorhanden, einer Session-ID aus einer früheren Sitzung. Der Server antwortet mit einer neuen Connection-ID. Wenn er im Cache die angegebene Session-ID findet, können beide Seiten einen früher vereinbarten **Hauptschlüssel** (**master key**) benutzen. Anderenfalls sendet der Server sein **Zertifikat** und eine Liste der verwendbaren Chiffren. Der Client generiert einen neuen Hauptschlüssel und sendet ihn, mit dem Public-Key des Servers verschlüsselt, an diesen. Aus dem Hauptschlüssel und verbindungsbezogenen Daten werden mittels einer **Hash-Funktion** (**MD5**) die Sitzungsschlüssel abgeleitet, die für die Datenverschlüsselung Anwendung finden. Für jede Richtung (Senden/Empfangen) wird dabei ein eigener Sitzungsschlüssel benutzt - der Hauptschlüssel selbst kommt bei der Datenverschlüsselung nie zum Einsatz. Ab-

schließend schickt der Client die mit seinem Sendeschlüssel chiffrierte Connection-ID und der Server den, mit seinem Sendeschlüssel verschlüsselten, Einmalwert. Der Client überprüft unter Verwendung seines Empfangsschlüssels, ob der Einmalwert mit dem von ihm gesendeten übereinstimmt und kann damit sicher gehen, daß der Server als der tatsächliche Inhaber des Zertifikats authentisch ist. Der Einmalwert wird jeweils mit dem Private-Key digital signiert. Damit wird der Zusammenhang zwischen ID bzw. Zertifikat und Daten hergestellt. Der Server hat ebenfalls die Möglichkeit, die Authentizität des Clients zu überprüfen. Die Aufforderung dazu enthält einen Einmalwert und eine Liste anwendbarer Verfahren zur Authentifikation. Der Client antwortet mit seinem Zertifikat und Authentifikationsinformationen. Für diese wird mit MD5 ein digest über Sende- und Empfangsschlüssel, den Einmalwert und das Zertifikat des Servers erzeugt und mit dem privaten Schlüssel des Clients (entsprechend dem Zertifikat) verschlüsselt. Zum Abschluß schickt der Server dem Client verschüsselt die neue Session-ID.

5. Das **SSL Record Protocol** nimmt die Ver- bzw. Entschlüsselung sowie, falls verlangt, eine Komprimierung der Nutzdaten vor. Nach erfolgreicher Beendigung des Handshake-Protokolls sind beide Seiten zur Übertragung der Anwendungsdaten bereit. Diese werden im Rahmen eines Record-Protokolls nach dem vereinbarten Verfahren verschlüsselt und mit einem Message Authentication Code zur Gewährleistung der Datenintegrität versehen. Das SSL-Protokoll bietet einen sehr einfachen und effizienten Mechanismus zur Befriedigung der Sicherheitsbelange vieler Anwendungsprotokolle.

Die genauen Erläuterungen zu Zertifikaten und kryptographischen Verfahren wurden bereits in Kapitel 5 durchgesprochen.
Die Organisation von SSL als Layer zwischen Applikations- und Transportschicht erlaubt dabei, zusätzlich auf Anwendungsebene Sicherheitsprotokolle wie **S-HTTP** (**Secure HTTP**) oder **SET** (**Secure Electronic Transactions**) zu implementieren. Dadurch läßt sich die Gesamtsicherheit des Systems noch einmal steigern, wenn auch zu Lasten der Performance.

Portnummer	**Art**	**Protokoll**
443	HTTP über SSL	HTTPS
465	SMTP über SSL	SSMTP
563	NNTP über SSL	SNNPS
992	Telnet über SSL	LNETS
995	POP3 über SSL	SPOP3

Tabelle 13: Gesicherte Verbindungen

Neben der Verschlüsselung bietet SSL optional noch eine zweite Methode an, die Verbindung zu sichern: Die Authentisierung eines oder beider beteiligter Kommunikationspartner.

Diese Variante stützt sich auf den Einsatz externer **Certification Authorities** (wie etwa Verisign `https://www.verisign.com`), über die man sich digitale Zertifikate ausstellen lassen kann. Die digitalen Unterschriften der wichtigsten Zertifizierungs-Instanzen bringen Internet Explorer und Navigator dabei schon mit, so daß zur Kommunikation mit diesen nicht noch extra Zertifikate angefordert werden müssen. Die verschiedenen SSL-Protokollversionen bieten bei der Authentifizierung unterschiedliche Möglichkeiten.

- **SSL-2** erfordert lediglich eine Zertifizierung des Servers, so daß der Kommunikationspartner vom Client aus einwandfrei identifizierbar ist. Dadurch lassen sich gängige Täuschungsmanöver wie serverseitiges IP-Spoofing (also das Vorgaukeln einer fremden IP-Adresse zum Abfangen der an diese IP-Adresse gerichteten Daten, vgl. Seite 89) unterbinden. Beim Verbindungsaufbau erhält der Client vom Server das digitale Zertifikat der Site. Daraufhin generiert der Client einen Session-Key, mit dem er die gesamte Kommunikation mit der Site verschlüsselt. Den Key selber verschlüsselt er mit dem Public-Key des authentifizierten Servers, so daß nur dieser die Verbindungsdaten lesen kann.
- **SSL-3** setzt die Zertifizierung beider Kommunikationspartner voraus. Dazu muß der Anwender bei einem **Trust Center** ein Client-Zertifikat anfordern, das meist kostenpflichtig ist (ca. 15 Dollar). In der Kombination von Server- und Client-Authentifizierung bietet SSL dann komplett abgesicherte Verbindungen via Internet.

Neben den 'Standard'-Varianten von SSL existieren auch spezialisierte Varianten dieses Verfahrens. Die bekannteste davon ist Fortezza, welche von den US-Behörden zum Austausch sensitiver Daten benutzt wird. Sie setzt zur schnelleren Verschlüsselung auf Hardware und verwendet als Schlüsselaustauschmechanismus KEA anstatt RSA. Über kurz oder lang wird SSL als Standard vermutlich von dem auf ihm basierenden TLS 1.0 - RFC 2246 [DiAl 99], ganz abgelöst werden.

Ausführliche Whitepapers zu Aufbau und Funktion des Secure Sockets Layer und den dazugehörigen Sicherheits-Zertifikaten finden Sie unter [net 00].

Typische Beispiele für Certification Authorities sind Verisign (`http://www.verisign.com/`) und Globalsign (`http://www.globalsign.com/`).

Dort erhalten Sie sowohl Server- als auch Client-Zertifikate. Von den Client-Zertifikaten offerieren beide Anbieter auch kostenlose, jedoch zeitbeschränkte Versionen. Eine umfassende Darstellung des neuen IETF-Standards TLS 1.0 direkt aus erster Hand bietet der RFC 2246 [DiAl 99].
Zur Verifizierung von Zertifikaten sind offizielle CA-Zertifikate, mit denen die Webserver-Zertifikate unterschrieben werden, nötig, die bereits fest in den Browsern intergriert sind.

7.2.3 S-HTTP - Secure Hypertext Transfer Protokoll

Dieses Protokoll sei hier nur am Rande erwähnt, da es sich nicht als verwendeter Standard durchsetzen konnte.
Das Secure Hypertext Transfer Protocol nimmt nicht nur am Transferprotokoll Erweiterungen vor, sondern definiert auch neue Elemente für die HTML-Sprache. S-HTTP stellt einen Rahmen für die Anwendung verschiedener kryptografischer Standardmethoden dar. Jede Nachricht kann durch eine beliebige Kombination aus drei Mechanismen geschützt werden: digitale Unterschrift, Datenverschlüsselung und Authentifizierung. Eine S-HTTP-Nachricht besteht aus einer gekapselten HTTP-Nachricht und einigen vorangestellten Kopfzeilen, die das Format der gekapselten Daten beschreiben.
Beide Seiten können im Rahmen einer Verhandlung Angaben über die verwendbaren beziehungsweise geforderten Erweiterungen gegenüber HTTP machen. Dazu gehören: Nachrichtenformate, Typen der Zertifikate, Schlüsselaustauschmechanismus, Verfahren für digitale Unterschriften, Hash-Algorithmus für den **digest** sowie Verschlüsselungsverfahren für Kopf und Inhalt. Das könnte folgendermaßen aussehen: 'Dieser Client verschlüsselt alle Nachrichten mittels DES und vermag mit DES oder RC5 verschlüsselte Nachrichten zu empfangen.' Sowohl asymmetrische als auch symmetrischen Verfahren sind aufgeführt. Im ersten Fall stehen auch digitale Unterschriften zur Verfügung, im zweiten kann der Austausch des Geheimschlüssels auf drei Wegen erfolgen: in-band (Schlüssel wird mit dem öffentlichen Schlüssel des Servers geschützt), out-band (ein vorher festgelegter Schlüssel) und Kerberos (Nutzung von Kerberos-Tickets). Mit dem Sitzungsschlüssel läßt sich ein Transaktionsschlüssel chiffrieren, der bei der Datenverschlüsselung Anwendung findet.
S-HTTP definiert einen neuen URL-Protokolltyp `shttp`, der auf die Fähigkeiten des Servers bezüglich S-HTTP hinweist. Der Client wird damit aufgefordert, bereits die Anforderung gekapselt zu senden. Die dazu notwendigen Informationen sind in neuen Attributen (DN, NONCE, CRYPTOPTS) des Ankers für diesen Link beziehungsweise dem neuen Sprachelement (tag) CERTS enthalten. Damit lassen sich beispielsweise die Informationen eines Formulars verschlüsseln.

7.2.4 Der Webserver „Apache“

Die Apache Konfigurationsdateien sind im Verzeichnis `/etc/httpd/` zu finden. Früher gab es eine Konfigurationsaufteilung zwischen den Files `access.conf`, `srm.conf` und `httpd.conf`. Die komplette Konfiguration wird aber nur noch in der `httpd.conf` vorgenommen. Unter `ServerRoot` wird das Verzeichnis festgelegt, in dem defaultmäßig die Verzeichnisstruktur des Webservers beheimatet ist. Hier liegen die Konfigurationsfiles und alles, was innerhalb des `httpd.conf` nicht explizit festgelegt ist. Mit `Listen` kann der Port angegeben werden, auf den der Apache gebunden wird. Unter `LoadModule` stehen alle im Apache geladenen Module. `DocumentRoot` ist das Verzeichnis, das die Quelle aller Webserverinformationen beinhaltet. Ebenso wird festgelegt, wohin die

Logdaten geschrieben werden und in welcher Form: `ErrorLog`, `LogLevel`, `LogFormat`, `CustomLog`, etc. Sollen auf einem Apache mehrere virtuelle Webserver laufen, so werden diese als sogenannte Virtual Hosts konfiguriert (`VirtualHost`). Das kann auf Namens- oder IP-Adressbasis erfolgen, wobei für virtuelle Server, die HTTPS können sollen, die IP-Adressbasis zu verwenden ist, da sonst die Zuordnung der Zertifakte nicht mehr stimmt.

Zugriffsbeschränkungen

Falls es Dokumente auf dem Webserver gibt, die nicht allgemein zugänglich sein sollen, gibt es die Möglichkeit, den Zugriff soweit einzuschränken, daß entweder nur von bestimmten IP-Adressen aus zugegriffen werden kann oder daß sich die Anwender authentisieren müssen.
Befassen wir uns zuerst mit der Zugriffsbeschränkung auf IP-Adressen: Angenommen, es gibt in dem `index.html` File des Webservers Links auf Dateien in einem Verzeichnis `restricted`. Nun soll aber nur ein bestimmter IP-Adressbereich auf diese Informationen zugreifen können. Diese Einschränkung ist so zu konfigurieren:

```
<Directory /restricted>
order allow,deny
allow from 10.10.10.0/24 192.168.1.1
</Directory>
```

Es ist aber nicht immer sinnvoll, Zugriffsbeschränkungen nur auf IP-Adressbasis zu machen. Apache ist in der Lage, Userauthentisierung jeglicher Art anzubieten. Dies soll hier mit einem Standardverfahren vorgestellt werden. Dazu muß man zuerst ein File anlegen, in das die User mit verschlüsseltem Passwort eingetragen sind. Dieses File sollte aber nicht im `DocumentRoot` stehen, da es sonst evtl. abfragbar und veränderbar ist. In diesem Beispiel wird die Authentisierungsdatei `testuser` unter `/etc/httpd/` abgelegt. Nun soll der User test eingerichtet und das Authentisierungsfile neu erzeugt werden:
`htpasswd -c /etc/httpd/testuser test`
`-c` sorgt für die Erzeugung des Files. Werden weitere User dem bestehenden File zugewiesen, so ist diese Option nicht mehr anzugeben.
Nun ist im Konfigurationsfile `httpd.conf` noch festzulegen, daß das Verzeichnis `restricted` nur von Usern, die im Authentisierungsfile `/etc/httpd/testuser` stehen, gelesen werden darf:

```
<Directory /restricted>
AuthName "restricted stuff"
AuthType Basic
AuthUserFile /etc/httpd/testuser
```

```
require valid-user
</Directory>
```

`AuthName` gibt den sogenannten Realm an, d.h. sobald ein Anwender seine User-ID mit gültigem Passwort angibt, muß er nicht bei jedem Verbinden die Authentisierungsdaten neu eingeben, sondern kann alle Resourcen, die dem Realm `restricted stuff` angehören, einsehen.
`AuthType` gibt den Authentisierungstype an. Zu Zeit ist nur `Basic` verfügbar.
`AuthUserFile` ist die Location des Authentisierungfiles.
`require` gibt an, was nötig ist, um den Zugang zu bekommen. `valid-user` heißt, daß jeder User aus dem Authentisierungsfile, der sich richtig angemeldet hat, Zugang bekommt. Man kann aber auch nur einzelnen Usern aus dem File den Zugriff gewähren: `user userid`.
Da bei HTTP alle Daten unverschlüsselt übertragen werden, sind UserID und Passwort im Klartext im Netz lesbar. Daher sollte für solche Seiten HTTPS verwendet werden.

Verschlüsselte Verbindungen

Natürlich gibt es auch Daten (User-IDs, Passwörter, Dokumente, etc.), die nicht ungeschützt übertragen werden sollten. Um aber verschlüsselte Verbindungen anbieten zu können, muß am Webserver noch einiges getan werden. Die SSL-Verschlüsselung kann man mit Apache-SSL oder durch Einbeziehen von `mod_ssl` erzielen. Der Webserver mit Verschlüsselung soll nun `www.test.de` heißen und über `https://www.test.de` erreichbar sein.
Betrachten wir nun die Vorgehensweise bei Apache-SSL:

- Zuerst sollte man in sein SSL Verzeichnis wechseln: `cd /usr/local/ssl/private`
- Nun wird der private Schlüssel generiert: `openssl genrsa -des3 512/1024 > www.test.de.key`. Dieser Schlüssel sollte zusammen mit der Passphrase gut verwahrt werden. Es muß dabei aber bedacht werden, daß die hier verwendete Passphrase bei jedem Neustart des Apache eingegeben werden muß. Das kann sich in der Praxis als hinderlich erweisen.
- Nach dem Wechsel in das Zertifikatsverzeichnis `/usr/local/ssl/certs` wird ein **Zertifikatrequest** (**CSR**) erstellt: `openssl req -new -key ../private/www.test.de.key > www.test.de.csr`
- Mit diesem CSR kann dann selbst ein **Zertifikat** (**CRT**) erstellt werden: `openssl req -x509 -key ../private/www.test.de.key -in www.test.de.csr > www.test.de.crt`

`genrsa -des3 512/1024` zeigt an, daß ein RSA-Schlüssel mit Verschlüsselungsalgorithmus DES3 und Schlüssellänge 512/1024 erzeugt werden soll.

`req -new -key` sorgt dafür, das ein neuer X.509 **Certificate Signing Request** (**CSR**) mit dem soeben generierten Schlüssel erzeugt wird.
`req -x509 -key ... -in` sorgt dafür, daß ein neues Zertifikat aus dem Schlüssel und dem CSR erzeugt wird (**self signing certificate**). Das Zertifikat wurde nun mit dem selbsterzeugten Schlüssel unterschrieben. So könnte man sich eine eigene **CA** (**Certificate Authority**) aufbauen.
Dieses Zertifikat wird als nicht offizielles Zertifikat durch den Browser erkannt. Für temporäre Tests ist dies ein gangbarer Weg, für Produktivsysteme sollte aber ein offizielles Zertifikat durch eine der CA's ausgestellt werden (`www.thawte.com`, etc.), indem der CSR an die CA geschickt wird. Organisatorische Informationen dazu finden Sie auf den Webseiten der Anbieter.
Im Konfigurationsfile `httpd.conf` wird der Pfad für das Zertifikat und den privaten Schlüssel festgelegt:

```
SSLCertificateFile    /usr/local/ssl/certs/www.test.de.crt
SSLCertificateKeyFile /usr/local/ssl/private/www.test.de.key
```

Die Befehle bei der `mod_ssl` Methode sind analog.
Es gibt auch noch folgende Darstellung der Befehle:

- `openssl genrsa -des3 -out www.test.de.key 1024`
- `openssl req -new -key www.test.de.key -out www.test.de.csr`
- `openssl x509 -req -in www.test.de.csr -signkey www.test.de.key -out www.test.de.crt`

Es muß noch darauf geachtet werden, daß das Modul `mod_ssl` installiert ist. Bei SuSE sind folgende Eintäge in der `/etc/sysconfig/apache` abzuändern:

- `HTTPD_START_TIMEOUT=10` (nur nötig, wenn Passphrase bei Zertifikaten vergeben wurde)
- `HTTPD_SEC_MOD_SSL=yes`

Danach muß der Befehl `SuSEconfig` aufgerufen werden, damit die Einträge wirksam werden können. Weitere Informationen zum Befehl `openssl` finden Sie unter `man openssl`. Die hier aufgeführten Informationen zu Zertifkatserstellung, Apache allgemein und noch vielem mehr finden Sie unter [tha 00], [apa 02] und [aps 02].

7.3 Praktische Aufgaben

7.3.1 Topologie

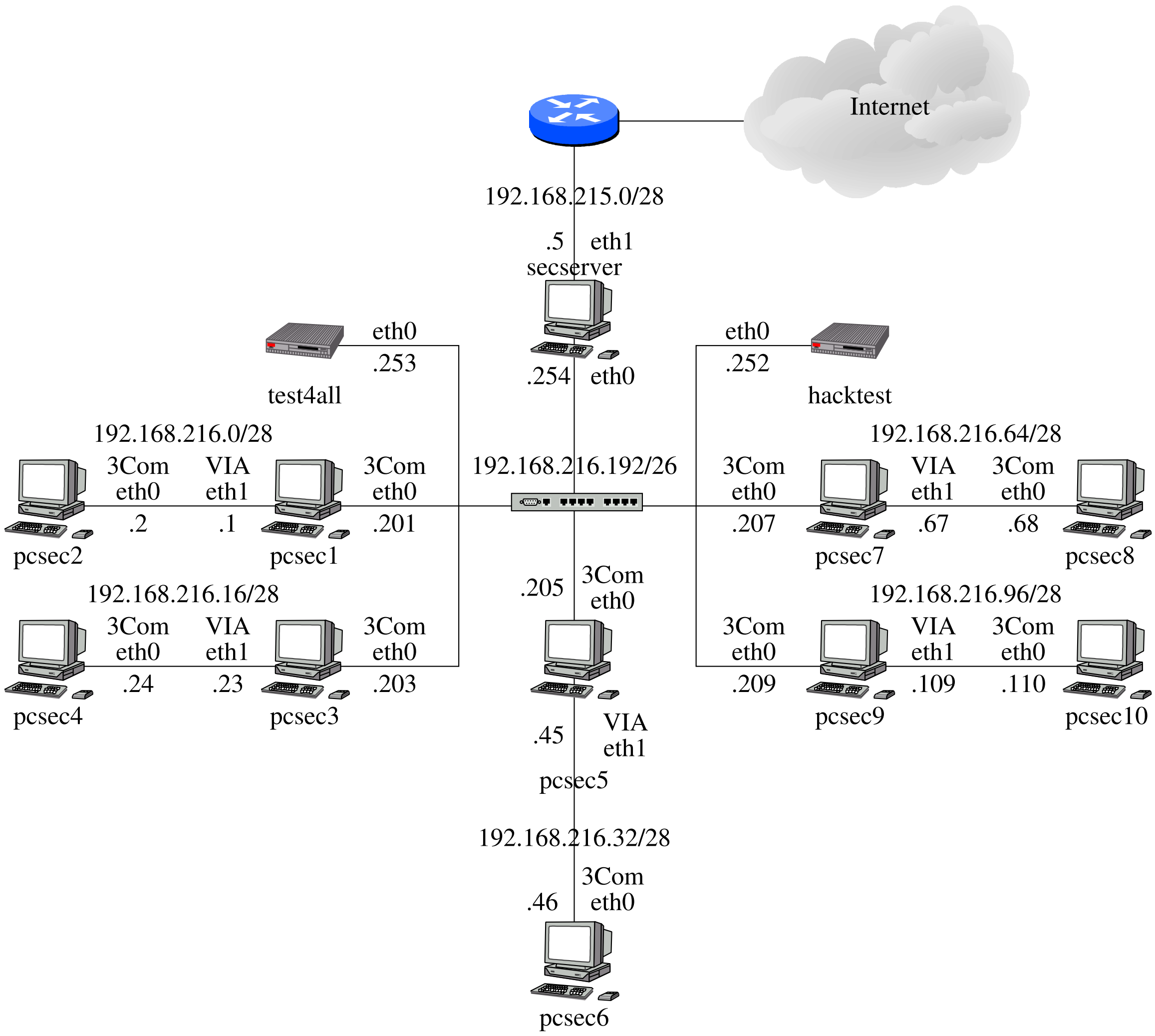

Abbildung 69: Der Versuchsaufbau für die weiteren Versuche des Praktikums

7.3.2 Konfiguration Sendmail

Arbeiten Sie bei Konfigurationen immer mit IP-Adressen nicht mit den Namen der Maschinen.

1. Richten Sie Ihren Mailserver so ein, daß er den `secserver` als Smarthost verwendet. Was hat das für Konsequenzen?

2. Schicken Sie die für die Praktikumsrechner bestimmte Mails nicht über DNS MX Records, sondern über Einträge in den internen Listen direkt an die Praktikumsrechner. Test sind mit `mail -v test@pcsec?.secp.nm.informatik.uni-muenchen.de` möglich.

3. Erlauben Sie ihrem Nachbarrechner, Ihren Rechner als Relay zu verwenden und testen Sie es über `telnet <IP-Adresse> 25`.

4. Sorgen Sie dafür, daß Sie von anderen Rechnern nur Mails für Ihren Rechner annehmen.

5. Sorgen Sie dafür, daß der `secserver` und einer Ihrer Nachbarrechner keine Mails an Sie schicken darf.

6. Überprüfen Sie Ihre Konfiguration mittels `telnet <IP-Adresse> 25` und `mail -v` und anhand der Logfileeinträge.

Siehe auch `man mail`. Die Option `-v` sorgt dafür, daß das Mailprogramm im Verbose Modus läuft und die aufzubauenden Verbindungen anzeigt.

7.3.3 Konfiguration Apache

1. Installieren Sie das Apache-Paket und `mod_ssl`. Dabei sind folgende Eintäge in der `/etc/sysconfig/apache` abzuändern (das ist SuSE spezifisch):

   - `HTTPD_START_TIMEOUT=10` (nur nötig, wenn Passphrase bei Zertifikaten vergeben wurde)
   - `HTTPD_SEC_MOD_SSL=yes`

   Danach muß der Befehl `SuSEconfig` aufgerufen werden, damit die Einträge wirksam werden können.

2. Richten Sie unter Ihrem `DocumentRoot` ein Verzeichnis ein, auf das nur der Masterserver Zugriff per HTTP erhalten soll.

3. Schützen Sie ein von Ihnen gewähltes File mit Userauthentisierung.

4. Ändern Sie den HTTP Port auf 8080 ab.

5. Richten Sie eine SSL-Verschlüsselung mit `mod_ssl` ein.

8 Proxies und Application Level Gateways

Ein **Application Level Firewall** erlaubt die Kontrolle der in den Anwendungsprotokollen abgesetzten Kommandos. Er erlaubt jedoch keine Kontrolle der in von den Anwendungsn übertragenen Daten. Dies kann nur von Content Filtering-Systemen (siehe Abschnitt 2.5.5) erledigt werden.
Bei einem Application Level Firewall muß für jedes durch den Firewall vermittelte Protokoll ein spezielles Programm, ein sogenannter **Proxy**, vorhanden sein. Der Proxy vermittelt zwischen den beiden Seiten des Firewalls. Das Proxy-Programm verhält sich genau so wie das Originalprogramm des jeweiligen Protokolls, z.B. wie ein FTP-Server. Mit einem Proxy ist es möglich, auf allen Ebenen der Kommunikation den Datenstrom zu filtern und zu beeinflussen.
Die heutigen Firewall-Produkte sind meist eine Kombination der unter Abschnitt 8.1 geschriebenen Firewalltechnologien. Je nach Produkt sind die einzelnen Funktionen unterschiedlich stark ausgeprägt.
Ein Firewall kann aus einer einzelnen Maschine oder aus einer mehrstufigen Anordnung bestehen. Eine mehrstufige Anordnung ist vor allem dann sinnvoll, wenn man bestimmte Dienste der Öffentlichkeit zur Verfügung stellen will, etwa einen WWW- oder FTP-Server. Die entsprechenden Hosts können dann in einem Zwischennetz isoliert werden, wie auch in Abschnitt 9.2 genauer beschrieben wird.

8.1 Philosophie

Es gibt drei Arten von Firewall-Softwareebenen:
Die aus Kapitel 3 bekannten **Paketfilter** überprüfen die Quell- und Zieladresse (IP-Adresse und TCP/UDP-Port) eines Pakets und entscheiden, ob es passieren darf oder nicht. Der Vorteil besteht in der Transparenz für den Anwender. Diese Transparenz ist aber zugleich von Nachteil: Paketfilter können nicht zwischen Nutzern und deren Rechten unterscheiden. Paketfilter sind im allgemeinen auf Routern angesiedelt und werden heute von den meisten Herstellern mitgeliefert. Intelligente Paketfilter analysieren zusätzlich den Inhalt der Pakete und erkennen auch die Zulässigkeit von Verbindungen, die einfache Paketfilter nicht erlauben würden (z. B. Datenverbindung bei ftp, siehe auch Kapitel 3).
Circuit Level Gateways sind mit Paketfiltern vergleichbar, arbeiten jedoch auf einer anderen Ebene des Protokollstacks. Verbindungen durch solch ein Gateway erscheinen einer entfernten Maschine, als bestünden sie mit dem Firewall-Host. Somit lassen sich Infomationen über geschützte Netzwerke verbergen.
Application Level Gateways, auch **Proxy** (Stellvertreter) genannt, stellen ein anderes Firewall-Konzept dar. Hierbei wird auf dem Firewall-Host für jede zulässige Anwendung ein eigenes Gateway-Programm installiert. Der Client muß sich dabei oftmals gegenüber dem Proxy-Programm authentifizieren. Dieser Proxy führt dann alle Aktionen im LAN stellvertretend für den Client aus. Damit lassen sich zum einen benutzerspezifische Zu-