Sichere Kommunikation im Internet

Vertraulichkeit, Integrität und Authentizität in einem anonymen Netzwerk

Inaugural-Dissertation
zur
Erlangung der Doktorwürde
der
Philosophischen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
zu Bonn

vorgelegt von

Jörg Hartmann

aus Köln

Bonn 2002

Gedruckt mit Genehmigung der Philosophischen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Berichterstatter: Prof. Dr. Winfried Lenders
2. Berichterstatter: Prof. Dr. Wolfgang Hess

Tag der mündlichen Prüfung: 24.4.2002

„We are not cynical. Our assessment is that security is very difficult, both to understand and to implement.“

Carl Ellison and Bruce Schneier [Ellison/Schneier 2000, S.7]

Inhalt

1 Einleitung

Wir leben in einem Zeitalter globaler Kommunikation. Diese Kommunikation hat sich in den letzten Jahrzehnten immer mehr zu einer gesellschaftlichen Notwendigkeit entwickelt, ohne die viele Errungenschaften unserer Zeit nicht mehr denkbar sind. So wie zu Beginn des 20. Jahrhunderts die Einführung des Telefons das Kommunikationsverhalten einer ganzen Generation veränderte, etabliert sich nun das Internet immer stärker als wichtigstes Medium der globalen Kommunikation.

Viele Forschungsvorhaben sind ohne das Internet gar nicht mehr denkbar. Dies betrifft in erster Linie die verteilte Zusammenarbeit von Wissenschaftlern über große Distanzen. Aktuelle Softwareprojekte (z.B. die Weiterentwicklung des Betriebssystems LINUX) zeigen, daß in dieser Zusammenarbeit ein ungeheures Potential liegt. Die koordinierte Einbeziehung von mehreren tausend Entwicklern in ein Projekt ist ohne das weltumspannende Datennetz völlig ausgeschlossen.

Auch in der Linguistik läßt sich dieses Potential nutzen. Die Erstellung, Prüfung und Weiterentwicklung linguistischer Datenbanken ist ein zeitintensiver Prozeß, der oft die Mittel kleinerer Forschergruppen übersteigt. Die Einbindung einer größeren Zahl von Wissenschaftlern wäre hier ein deutlicher Vorteil. Durch die Kombination verteilter Ressourcen ist eine enorme Produktivitätssteigerung erreichbar.

Dieses Vorgehen ist jedoch nicht ganz ungefährlich. Bei der Nutzung des Internet verlassen wir uns zunehmend auf die Vertraulichkeit, Integrität und Authentizität der übertragenen Daten. Und genau hier liegt das Problem. Globale Kommunikation über das Internet ist in der Wissenschaft

unerläßlich, gleichzeitig ist dieses Medium aufgrund seiner Struktur zum Transport sensibler Daten zunächst völlig ungeeignet.

Aus dieser einfachen Grundidee ergibt sich sowohl die Motivation als auch das Ziel der vorliegenden Arbeit. Sie stellt den Versuch dar, die Grundlagen, Anforderungen und Verfahren der sicheren Kommunikation systematisch darzustellen. Vor dem Hintergrund möglicher Bedrohungen und Schwachstellen existierender Verfahren wird ein neues, integratives Modell der sicheren Kommunikation entwickelt, welches den gestellten Anforderungen genügt. Dieses Modell wird im nächsten Schritt eingesetzt, um die Funktion eines sicheren Netzes anhand des Zugriffs auf die Bonner Wortdatenbank darzustellen.

Die Bonner Wortdatenbank wird in diesem Zusammenhang als reales System betrachtet, an dem das entwickelte Modell anschaulich dargestellt werden kann. Als Prüfstein des Modells werden die Komponenten einer Sicherheitsarchitektur für die Datenbank implementiert, welche den sicheren Zugriff realisieren.

Sicherheit im Internet ist ein umfassendes Thema, doch innerhalb dieses Themas stellen sich bei näherer Betrachtung immer wieder dieselben Probleme in verschiedenen Facetten. Oftmals werden einzelne Probleme herausgegriffen und im Detail dargestellt, die ohne den komplexen Zusammenhang, in dem sie stehen, nicht verständlich sind. Diese Arbeit ist auch der Versuch, Licht ins Dunkel zu bringen und die Problematik der sicheren Kommunikation systematisch darzustellen.

So dürfen wir beim Bau eines Systems nicht nur von Kommunikations- oder Netzwerkprotokollen sprechen. Diese sind notwendige Werkzeuge, mit denen wir unser Ziel - den Entwurf und die Implementierung eines sicheren Kommunikationssystems - erreichen können. Sie sind nur Komponenten

eines Systems, die ohne ein Gesamtkonzept zur Sicherung der Kommunikation im Internet nicht sinnvoll eingesetzt werden können.

In einer Zeit, in der Rechner und Netzwerke immer stärker zusammenwachsen, kann weder Rechnersicherheit noch Netzwerksicherheit isoliert betrachtet werden. Rechner- und Netzwerksicherheit sind gleichwertige Komponenten, die beim Bau eines sicheren Kommunikationssystems zusammenwirken.

Der Aufbau der Arbeit ergibt sich fast zwingend aus dem Thema. Der erste Abschnitt (Kapitel 2 und 3) befaßt sich mit den Grundlagen der sicheren Kommunikation, der zweite (Kapitel 4 und 5) mit der Kommunikationsstruktur im Internet. Beide Abschnitte stellen das zum Verständnis der folgenden Kapitel notwendige Hintergrundwissen systematisch dar.

Nun folgt der dritte Abschnitt (Kapitel 6 bis 8), in dem detailliert dargelegt wird, welchen Angriffen die Kommunikation im Internet ausgesetzt ist. Hier schließt sich der vierte Abschnitt (Kapitel 9 und 10) an, der mögliche Sicherheitsmechanismen erörtert, mit denen die zuvor geschilderten Angriffe abgewehrt werden können. Kapitel 11 stellt den aktuellen Stand der Technik zur Absicherung von linguistischen Datenbanken im Internet dar.

Im letzten Abschnitt (Kapitel 12) wird ein integratives Modell der sicheren Kommunikation entworfen und die eingesetzten sicherheitsrelevanten Komponenten werden anhand des Zugriffes auf die Bonner Wortdatenbank in die Praxis umgesetzt. Es folgt eine kurze Zusammenfassung der Ergebnisse mit einem Ausblick auf die weitere Entwicklung dieser Datenbank (Kapitel 13).

2 Grundlagen der sicheren Kommunikation

2.1 Gegenstand

Das Thema *Sichere Kommunikation im Internet* erfordert zunächst die Bestimmung des verwendeten Kommunikationsbegriffes. Wenn wir Kommunikation in einer ersten Abgrenzung als *Bedeutungsvermittlung zwischen Lebewesen* (vgl. [Maletzke 1998, S.37ff]) charakterisieren, wird schnell klar, daß unter Kommunikation im Internet nicht der bloße Datenaustausch zwischen Maschinen gemeint sein kann, sondern intentionale Informationsvermittlung, die von einer Maschine nicht zu leisten ist.

Menschen tauschen Informationen aus. Sie tun dies intentional in Form einer sozialen Interaktion. Der Austausch von Informationen durch Maschinen entspringt jedoch nicht einer von der Maschine verfolgten Intention, sondern stellt die bloße Vermittlung einer menschlichen Intention dar. Nur die menschliche Kommunikation ist zielgerichtet und spontan.
Die Maschine vermittelt Informationen, sie ist kein an der Kommunikation beteiligter Partner.

Damit wird die erste Abgrenzung des Gebietes *Kommunikation im Internet* deutlich. Kommunikation ist eine *Interaktion zwischen Lebewesen*. Der Austausch von Informationen zwischen Maschinen ist hingegen keine Kommunikation, da Maschinen weder Bedeutung vermitteln können, noch das Ziel der Informationsübermittlung selbst definieren, d.h. aus eigenem Antrieb zielgerichtet handeln können. Gleichwohl nehmen Maschinen die Rolle von Vermittlungspunkten zwischen den an der Kommunikation beteiligten menschlichen Partnern ein.

Der Prozeß der intentionalen Bedeutungsvermittlung in diesem Sinne führt nur dann zum Ziel, wenn zwei menschliche Lebewesen ihre kommunikativen Handlungen aufeinander richten und tatsächlich Inhalte miteinander teilen, d.h. eine Verständigung erreichen. So läuft Kommunikation im Internet auch nur dann erfolgreich ab, wenn die Bedeutungsvermittlung tatsächlich stattgefunden hat. Wir müssen also unterscheiden zwischen dem Ziel, Bedeutung zu vermitteln und seiner Erreichung in Form einer gelungenen Kommunikation.

Die allgemeine Intention, Bedeutung zu vermitteln, wird in der Regel begleitet von einer speziellen Intention. Diese spezielle Intention besteht in der Erreichung eines oder mehrerer Ziele, die über die bloße Bedeutungsvermittlung hinausgehen. Der aktive Kommunikationspartner verfolgt die Realisierung eines bestimmten Interesses. Kommunikation in diesem erweiterten Sinne ist erfolgreich, wenn nicht nur die Bedeutung, sondern auch das Interesse so vermittelt wird, daß es vom passiven Partner tatsächlich realisiert wird.

Gerade dann, wenn Kommunikation im Internet als Massenkommunikation verstanden wird, gewinnt die spezielle Intention an Bedeutung. Sie stellt die Grundlage für das Verständnis einiger Spielarten des E-Commerce dar, die erst entstanden sind, um diese Intention umzusetzen. So tritt beim Online-Shopping z.B. das Ziel, Waren zu verkaufen, in den Vordergrund.

Wenn wir über die *Vermittlung* von Bedeutung sprechen, ist darin schon enthalten, daß es eines Mittels bedarf, um Bedeutung und Intention zu transportieren. Der Begriff eines Mittel oder eines Mediums wird häufig in verschiedenen Zusammenhängen verwendet. Der Terminus kann verstanden werden als Bezeichner für eine materielle Repräsentation eines Bedeutungsinhaltes. In diesem Sinne sprechen wir z.B. vom Medium „Sprache“ als einem Ausdrucksmittel, über das Bedeutung vermittelt wird.

Immer häufiger findet sich jedoch eine Verwendung des Begriffes *Medium* im Sinne eines Kommunikationskanals, bzw. eines technischen Übertragungsmittels oder einer Infrastruktur zur Übermittlung von Informationen. In dieser Begriffsvariante sprechen wir z.B. vom „Medium Internet“ und meinen damit zunächst die Gesamtheit der vorhandenen Übertragungstechnik. In einem noch allgemeineren Sinne („die Medien“) umschließt der Begriff auch noch Institutionen und Organisationen, die Inhalte selbst erzeugen und nicht nur an ihrer Übertragung beteiligt sind.

Eine zweite Abgrenzung des Gebietes Kommunikation im Internet erfolgt über die Unterscheidung zwischen direkter und indirekter Kommunikation. Im Internet sind die Kommunikationspartner *räumlich getrennt*, oft über Kontinente hinweg. Neben der räumlichen tritt in vielen Fällen (z.B. E-Mail) auch eine *zeitliche Distanz* auf. Beide Merkmale kennzeichnen eine indirekte Kommunikation.

Eine vollständige Bestimmung des Gegenstandes „Internet-Kommunikation“ geht über die genannten grundlegende Charakteristika hinaus. Bisher wurde nur deutlich, daß es sich um eine indirekte Kommunikation zwischen Menschen handelt, die über eine noch näher zu bestimmende technische Infrastruktur abläuft. Noch ungeklärt ist, welche Ausdrucksmittel verwendet werden und unter welchen Rahmenbedingungen diese Kommunikation abläuft. In den nächsten Abschnitten wird diese Thematik erörtert.

2.2 Ausdrucksmittel

In der zwischenmenschlichen Kommunikation finden die verschiedensten Ausdrucksmöglichkeiten Verwendung. Unterschieden wird in der Regel zwischen linguistischer und nicht-linguistischer Kommunikation (z.B. Mimik und Körperhaltung). Kommunikation im Internet ist zu großen Teilen linguistische Kommunikation in Form geschriebener Sprache (z.B. E-Mail, Online-Chat etc.).

Mit dem Aufkommen neuer Technologien sind auch neue Ausdrucksmittel verbunden. Zur geschriebenen Sprache kommt die gesprochene hinzu (z.B. Voice-Mails oder Internet-Telefonie/Voice over IP). Durch die ständig ansteigende Verwendung des World Wide Web (WWW) wird geschriebene Sprache mit Bildern und Tönen verknüpft. Noch sehr selten ist die direkte Vermittlung eines körpergebundenen Ausdrucks zu finden, z.B. durch die sog. WebCam, mit der ein in kurzen Zeitintervallen aktualisiertes Bild des Kommunikationspartners übertragen wird.

Gleichzeitig ist ein Zusammenfließen verschiedener Ausdrucksmittel festzustellen. Das Übertragungsmedium Internet ermöglicht nicht nur den gleichzeitigen Transport der auf verschiedenste Art und Weise kodierten Informationen, sondern auch die zeitgleiche Darstellung dieser Informationen an einem Punkt, z.B. die Zusammenfassung von Bildern, Tönen und geschriebene Sprache auf einer WWW-Seite. Die Kombination verschiedener Ausdrucksmittel ist ein Charakteristikum der aktuellen Internet-Kommunikation.

Diese Qualität macht das Internet auch als Massenmedium attraktiv. Unter Massenkommunikation wird in der Regel öffentliche, indirekte, technisch vermittelte Kommunikation verstanden, die einseitig verläuft, d.h. mit

festgelegten Rollen für Kommunikator und Rezipient (vgl. [Rössler 1998, S.209ff]). In klassischen Massenmedien (Printmedien, Fernsehen etc.) ist keine direkte, auf dem selben Medium stattfindende, Interaktion möglich. Hier unterscheidet sich das Internet von anderen Massenmedien. Die Technologie ermöglicht eine direkte Interaktion auf dem Medium, z.B. durch Mausklick oder Texteingabe im WWW.

Als Marketinginstrument (z.B. im Bereich des Online-Shopping) besteht das Ziel darin, ein bestimmtes Verhalten bei potentiellen Kunden hervorzurufen. Das Internet unterliegt in diesem Fall den gleichen Gesetzen wie herkömmliche Marketinginstrumente. Um Zielgruppen anzusprechen bedarf es einer Wirkung, die auf der Glaubwürdigkeit der vermittelten Inhalte basiert. Die Problematik von Glaubwürdigkeit und Vertrauen wird im nächsten Abschnitt erörtert.

2.3 Kommunikation und Vertrauen

Das Vertrauen in ein Übertragungsmedium oder allgemeiner das Vertrauen in ein technisches System unterscheidet sich grundlegend vom Vertrauen in eine Person. Vertrauen in eine Person stützt sich im wesentlichen auf deren Reaktion auf bestimmte Ereignisse, d.h. auf die Zuverlässigkeit oder Korrektheit, mit der die Person auf eine bestimmte Situation reagiert. Vertrauen in ein technisches System ist hingegen Vertrauen in dessen Funktion und Sicherheit.

Der Begriff des „vertrauenswürdigen Systems“ ist in der Informationssicherheit klar definiert. Er hängt ab von „der korrekten Modellbildung, dem Prüfungsverfahren, dessen Detaillierungsgrad und Tiefe [Kersten 1991, S.59]“. In diesem Sinne wird unter der Vertrauenswürdigkeit eines Systems nichts anderes als die praktisch erreichbare Sicherheit dieses

Systems verstanden. Die zur Prüfung eingesetzten Standards und Verfahren sind in international gültigen Normen festgelegt (vgl. Kap. 10.5).

Vertrauen in diesem Sinn darf nicht verwechselt werden mit der Glaubwürdigkeit der dargestellten oder übertragenen Inhalte. Sowohl als Massenmedium als auch als Marketinginstrument kämpft das Internet mit dem Problem der Glaubwürdigkeit. Neuere Untersuchungen (vgl. [Rössler 1998, S.123ff]) haben beispielsweise ergeben, daß eine im WWW dargestellten Nachricht weit weniger glaubwürdig erscheint, als eine identische Nachricht, die über klassische Massenmedien verbreitet wird. Die Glaubwürdigkeit einer Information basiert jedoch zu einem erheblichen Maß auf dem Vertrauen, welches dem Medium entgegengebracht wird.

Auch wenn das Internet zur privaten Kommunikation genutzt wird, stellt sich das Problem des Vertrauens in einem anderen Zusammenhang. Die Art und Weise des Transportes von Informationen im Internet ist weit weniger greifbar als bei herkömmlichen Übertragungsmedien (z.B. Briefpost oder Telefon). Das dadurch ausgelöste Unbehagen bei der Verwendung des Mediums hängt auch mit der Unwissen darüber zusammen, welche Personen und Institutionen Zugriff auf die übertragenen Informationen haben. Im Beispiel der Briefpost vertrauen die Kommunikationspartner beispielsweise auf die Wahrung des Briefgeheimnisses durch die Überbringer der Informationen. In einem weltumspannenden Computernetz wie dem Internet kann diese Art des Vertrauens nicht aufgebaut werden, da die vermittelnden Organisationen und Institutionen weniger greifbar sind.

Noch problematischer ist, daß für die Kommunikationspartner nicht nur im Dunklen bleibt, wer zu welcher Zeit Zugriff auf die übertragenen Informationen hat, sondern daß in der Anonymität des Netzes die Identität des Partners oft nicht festgestellt werden kann. Dies ist kein neues Problem, sondern es tritt auch bei andere Übertragungsmedien auf. Anders als bei

einer Face-Face-Situation, ist bei einer indirekten, (technisch) vermittelten Kommunikation die Identität des Partners nicht direkt wahrnehmbar. Es existieren jedoch Erkennungsmerkmale, die mit der Information übertragen werden.

Betrachtet man die Erkennungsmerkmale in verschiedenen Übertragungsmedien, wird eine Hierarchie deutlich. Ausgehend von der direkten Face-Face-Kommunikation ergibt sich eine Abstufung hin zur indirekten Kommunikation. Während bei einigen Medien noch die Erkennungsmerkmale Bild und Ton übertragen werden (z.B. beim Fernsehen oder - in schlechterer Qualität - beim Bildtelefon), bleibt in anderen nur die Stimme (z.B. beim Telefon) oder ein tertiäres Merkmal (z.B. die Handschrift in einem Brief) zur Feststellung der Identität.
In der immer stärker eingesetzten E-Mail oder dem Online-Chat ist schließlich die Identität nicht mehr über ein personengebundenes Erkennungsmerkmal, sondern nur noch über eine Absendeadresse feststellbar.

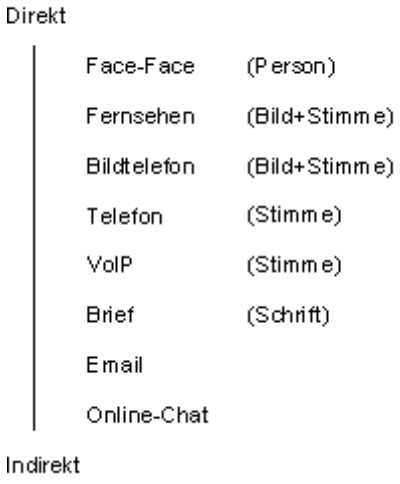


Abbildung 2.1: Hierarchie von Erkennungsmerkmalen

Während also die Feststellung der Identität des Kommunikationspartners eine Grundvoraussetzung zum Aufbau von Vertrauen darstellt, ist die

Kommunikation im Internet gerade durch eine Verminderung der zur Identitätsprüfung notwendigen Erkennungsmerkmale gekennzeichnet.

Durch diese Tatsache ergeben sich - zusammen mit der oben dargestellten Problematik der Anonymität des Mediums - für die Kommunikationspartner zwei Unsicherheiten, die das Vertrauen in das Übertragungsmedium Internet beeinflußen. Erstens fehlt die Möglichkeit einer Identitätsprüfung, als Ersatz für fehlende Erkennungsmerkmale. Zweitens besteht Ungewißheit darüber, für wen die Inhalte dieser Kommunikation einsehbar sind. Gerade dann, wenn sensible Inhalte kommuniziert werden, führt die Anonymität des Mediums zu einem weiteren Vertrauensverlust.

Im Bereich des E-Commerce kommt eine neue Ungewißheit hinzu. Die Rechtsverbindlichkeit von Geschäftstransaktionen gründet sich nicht nur darauf, daß die Identität der daran beteiligten Personen festgestellt wird, sondern auch auf die unverfälschte Übertragung von Inhalten. Ist es im Bereich des Online-Shopping beispielsweise möglich, relevante Teile einer Bestellung (z.B. den Kaufpreis) unbemerkt zu verändern, ist diese Verbindlichkeit nicht mehr gegeben.

Welche Möglichkeiten haben wir nun, diesen Unsicherheiten entgegenzuwirken? Wie können wir für fehlende Erkennungsmerkmale einen Ersatz schaffen, mit dem Ziel, das Vertrauen in die Internet-Kommunikation zu stärken? Wie können Menschen indirekt kommunzieren und doch Vertrauen hinsichtlich der Identität des Kommunikationspartners aufbauen?

Auf dem Weg zu einer Beantwortung dieser Fragen muß zunächst deutlich werden, welche Ziele im einzelnen verfolgt werden sollen, um eine vertrauenswürdige Kommunikation über ein unsicheres Medium zu ermöglichen.

2.4 Sachziele

Von sicherer Kommunikation wird im Rahmen dieser Arbeit gesprochen, wenn die Kriterien der *Vertraulichkeit, Integrität und Authentizität* der übertragenen Informationen erfüllt sind.

Unter *Vertraulichkeit* wird der Schutz der Information gegenüber der Kenntnisnahme Dritter verstanden. Die Inhalte der Kommunikation, mitunter sogar deren Existenz, sollen vor der Einsicht unberechtigter Personen verborgen werden.

Integrität bezeichnet die Erkennung von beabsichtigter Manipulation oder unbeabsichtigter Veränderung der Daten durch Dritte. Die Manipulation selbst kann grundsätzlich in einem öffentlichen Netz wie dem Internet nicht verhindert werden, doch die Daten können auf Unversehrtheit geprüft werden.

Mit dem Begriff *Authentizität* wird die Übereinstimmung der behaupteten Identität des Kommunikationspartners mit seiner wahren Identität bezeichnet. In der Anonymität eines elektronischen Netzwerkes können nur dann rechtsverbindliche Aktionen erfolgen, wenn zuvor die Identität einer handelnden Person zweifelsfrei festgestellt wurde.

Die genannten Sachziele Vertraulichkeit, Integrität und Authentizität fasse ich zusammen unter dem Terminus *Primärziele*. Sie bilden die Grundlage der sicheren Kommunikation und definieren zugleich die Mindestanforderungen, die an diese Kommunikation gestellt werden müssen.

Weitere Sachziele werden im folgenden als *Sekundärziele* bezeichnet. Eine herausgehobene Stellung hat in diesem Kontext die *Verfügbarkeit*. Dieser Terminus bezieht sich in erster Linie auf Rechensysteme und nur in zweiter

Linie auf Kommunikation. Verfügbarkeit kennzeichnet die Möglichkeit der Nutzung eines Systems durch berechtigte Personen. Die Forderung nach der zeitlich uneingeschränkten Nutzung eines Systems kann natürlich auch auf die Kommunikation übertragen werden und kennzeichnet dann die Verfügbarkeit einer Kommunikationsverbindung.

Verfügbarkeit bezeichnet in diesem Sinne die Forderung, daß Berechtigte jederzeit Zugriff auf die transportierte Information erhalten bzw. daß der Kommunikationskanal jederzeit verfügbar ist. Verfügbarkeit ist ein zentrales Sachziel der Informationssicherheit im allgemeinen, nicht jedoch der sicheren Kommunikation im besonderen. Da die Sicherheit einer nicht erfolgten Verbindung auch nicht gefährdet sein kann, bleibt die Einordnung als Primärziel umstritten. Im Rahmen dieser Arbeit wird Verfügbarkeit daher als Sekundärziel eingeordnet.

Als weitere Sekundärziele sind *Nicht-Abstreitbarkeit, Abrechenbarkeit und Anonymität* zu nennen. Die Forderung nach *Nicht-Abstreitbarkeit* fällt in den Bereich der Verbindlichkeit, der sowohl den Nachweis des Ursprunges umfaßt und unter dem Terminus Authentizität ein Primärziel darstellt, als auch den Nachweis des Empfangs einer Nachricht. Die Nicht-Abstreitbarkeit des Empfanges kann z.B. durch eine digital signierte Quittung des Adressaten erfolgen.

Hier schließt sich auch die Forderung nach *Abrechenbarkeit* an, d.h. die Möglichkeit, die Kosten einer Online-Dienstleistung einer bestimmten Person in Rechnung zu stellen. Dies wird durch die Sicherstellung der Authentizität einer Anforderung gewährleistet und kann weiter über Mechanismen abgesichert werden, die auch zur Erreichung der Nicht-Abstreitbarkeit eingesetzt werden.

Anonymität schließlich ist ein Ziel, welches der Authentizität entgegensteht. In einem globalen Netz wie dem Internet kann es jedoch notwendig sein, die Identität einer Person zu verbergen, um beispielsweise die Erstellung von Kundenprofilen oder die Zustellung von E-Mails, die Werbung enthalten zu verhindern.

2.5 Sicherheit von Systemen

Die angeführten Sachziele der sicheren Kommunikation dürfen nicht darüber hinwegtäuschen, daß kein real existierendes System hundertprozentig sicher sein kann. Sicherheit ist ein relativer Begriff: Die *Sicherheit eines Systems* hängt vom Kenntnisstand und vom Aufwand ab, den sein Entwickler einsetzt, um es gegen Angriffe zu schützen. Dieser Aufwand muß in einem sinnvollen Verhältnis zum Wert der zu schützenden Informationen stehen. Je höher der Wert der zu übermittelnden Informationen ist, desto höher muß der Aufwand zum Schutz des Systems sein.

Ein System ist dann hinreichend geschützt, wenn der Wert der enthaltenen oder transportierten Informationen in einem angemessenen Verhältnis zu den möglichen Bedrohungen und Risiken steht. *Bedrohung* wird definiert als Summe aller Umstände, die zu einem Schaden am System führen können. Die Wahrscheinlichkeit, mit der eine Bedrohung in der Praxis auftritt, und die Höhe des möglichen Schadens bestimmen das *Risiko*, dem ein System ausgesetzt ist.

Ziel der Entwicklung sicherer Systeme ist ein möglichst hoher Widerstandswert des Systems und seiner Bestandteile. Der *Widerstandswert* beschreibt die Resistenz eines Systems gegenüber einer Bedrohung oder einem gezielten Angriff. Systeme mit hohem Widerstandswert erfordern

einen sehr viel höheren Aufwand an Ressourcen, Geldmitteln und Zeit, um das System erfolgreich anzugreifen.

Es gibt keine Systeme, die allen existierenden und zukünftigen Bedrohungen uneingeschränkt standhalten können. Ein System kann jedoch so beschaffen sein, daß es mit vertretbarem Aufwand in angemessener Zeit nicht überwunden werden kann. Ziel ist daher nicht die Errichtung eines absolut sicheren Systems sondern die ständige Verbesserung des Widerstandswertes.

Das wichtigste Hilfsmittel zum Bau sicherer Systeme stellt die Kryptographie dar, die im nächsten Abschnitt beschrieben wird.

3 Kryptographie

Der Wunsch nach sicherer Kommunikation, d.h. nach einer Kommunikation, die mindestens den primären Sachzielen Vertraulichkeit, Integrität und Authentizität genügt, ist nicht erst im Zeitalter des Internet entstanden.

Seit Menschen miteinander Informationen austauschen, unabhängig davon, ob die Inhalte der Kommunikation privater, geschäftlicher oder politischer Natur sind, besteht der Wunsch oder gar die Notwendigkeit, diese Inhalte vor Dritten geheimzuhalten.

In besonderen Situationen, z.B. im Krieg, ist die Notwendigkeit zur Geheimhaltung von Informationen unmittelbar evident, unter anderen Umständen, z.B. in der privaten Kommunikation, hängt sie vom Schutzbedürfnis der einzelnen an der Kommunikation beteiligten Individuen ab.

So entstanden die ersten Methoden zur Verschlüsselung von Informationen, z.B. die in Abschnitt 3.1 beschriebene Cäsar-Chiffre mit dem Ziel, militärische Geheimnisse zu bewahren.

Die grundlegenden Prinzipien der Kryptographie haben sich dabei über die Jahrtausende kaum verändert, die eingesetzten Methoden weichen jedoch erheblich voneinander ab. Die Kryptographie als Wissenschaft von der Ver- und Entschlüsselung von Informationen ist ein Teilgebiet der mathematischen Disziplin Kryptologie, die neben der Kryptographie auch die Kryptoanalyse umfaßt, welche sich mit der Frage beschäftigt, wie verschlüsselte Informationen ohne Berechtigung wieder entschlüsselt werden können.

Grundsätzlich wird bei der Verschlüsselung von Informationen Klartext in Chiffretext überführt. Der Prozeß der Entschlüsselung kehrt diesen Vorgang um und erzeugt aus dem Chiffretext wieder lesbaren Klartext. In beiden Fällen wird ein kryptographischer Algorithmus verwandt, d.h. eine auf einer mathematischen Funktion beruhende komplexe Anweisung.

Der kryptographische Algorithmus definiert dazu zwei grundlegende Operationen: *Substitution* und *Transposition*. Durch Ersetzung und Vertauschung entstehen aus Klartextnachrichten unleserliche Chiffretexte. Die Verfahren der Substitution und Transposition wurden schon in den frühen Kryptosystemen der Antike eingesetzt, historische Algorithmen haben zur Durchführung dieses Vorganges Zeichen vertauscht, moderne Varianten bedienen sich des Computers und vertauschen und ersetzen einzelne Bits des Klartextes nach denselben Operationen.

3.1 Historische Kryptosysteme

Eine der bekanntesten antiken Verfahren zur Verschlüsselung militärischer Informationen ist die *Caesar-Chiffre*. Hier wird jeder Buchstabe des Alphabets um eine oder mehrere Positionen verschoben. Bei einer Verschiebung um nur einen Buchstaben wird z.B. die Zeichenkette "CAESAR" zu "DBFTBS". Da hier Buchstaben durch andere ersetzt werden, handelt es sich um eine Substitutionschiffre, genauer genommen um einen Sonderfall der Substitutionschiffre, um eine *monoalphabetische Substitution*. Jedem Zeichen im Chiffretext entspricht immer genau ein Zeichen im Klartext.

Diese Art der Verschlüsselung war auch ohne kryptoanalytische Kenntnisse leicht zu entschlüsseln, da charakteristische Muster im Chiffretext erhalten blieben, die mit Zeichenfolgen im Klartext verglichen werden konnten. Da

sich die Verschlüsselungsvorschrift innerhalb eines Textes nicht veränderte, war dieser nach Ermittlung der Substitutionsvorschrift vollständig lesbar.

Einen besseren Schutz bot da die etwas kompliziertere *polyalphabetische Substitution*, bei der nicht nur einfache Ersetzungen (z.B. des Buchstabens "C" durch "D") vorgenommen wurden. Bei diesem Verfahren ist jedes Chiffretextzeichen nicht nur vom Klartextzeichen selbst, sondern auch von dessen Position abhängig, d.h. die Position ist Teil der Substitutionsvorschrift (in der Zeichenkette "CAESAR" wird beispielsweise des erste "A" zu "B", das zweite jedoch zu "C", aus "CAESAR" wird so "DBFTCS"). Ein Angriff durch einfache Mustererkennung wird durch die Erweiterung des Verfahrens deutlich erschwert. Die polyalphabetische Substitution wird oft auch als Vigenere-Chiffrierung bezeichnet, nach ihrem Erfinder Blaise de Vigenere, der sie im 16. Jahrhundert entdeckte. Laut [Wobst 1997] ist diese Bezeichnung jedoch nicht ganz korrekt, da Vigenere ein allgemeineres Verfahren beschrieb, welches beliebige Substitutionen zyklisch verschob (vgl. [WOBST 1997], S. 36).

Neben der Substitution wurde auch die Transposition bereits seit der Antike zur Verschlüsselung verwendet. Ein gutes Beispiel für ein antikes Verfahren der Transposition ist die *Skytale von Sparta*. Es handelt sich hierbei um einen kantigen Holzstab, um welchen ein Band gewickelt wurde. Das Band wurde so beschriftet, daß es nur dann wieder lesbar war, wenn es auf einem Holzstab gleichen Umfanges erneut aufgewickelt wurde. Ein nicht aufgewickeltes Band zeigte nur eine scheinbar bedeutungslose Abfolge von Zeichen. Da die Zeichen hier nicht ersetzt, sondern nur entlang der Kanten des Holzstabes verschoben wurden, ist die Skytale eine Transpositionschiffre.

Auch wenn Substitution und Transposition über die Jahrhunderte als Verfahren der Kryptographie erhalten blieben, so änderte sich doch deren

Durchführung. Ein deutlicher Nachteil der antiken Vorgehensweise lag in der Wahl der erforderlichen Hilfsmittel begründet. Die Caesar-Chiffre wurde umständlich auf Papier ausgeführt, die Skytale erforderte zusätzlich einen Holzstab.

Maschinen zur Durchführung der Ver- und Entschlüsselung wurden schon früh erfunden, z.B. die Drehscheibe des Gelehrten Alberti bereits im Mittelalter, doch erst seit der Möglichkeit einer industriellen Fertigung fanden sie Verbreitung. Eine Mechanisierung der kryptographischen Verfahren gelang in größerem Umfang mit den sog. Chiffrierwalzen, zylindrischen Metallscheiben auf denen permutierte Alphabete eingraviert waren und die, in einer Reihe angebracht, gegeneinander so verschoben werden konnten, daß aus der neuen Anordnung ein Chiffriertext abzulesen war. Neben der Stellung und Anordnung der Walzen unterlag auch die Auswahl bestimmter Walzen aus einer größeren Menge möglicher Walzen der Geheimhaltung.

Mit dem Aufkommen des elektrischen Stroms entstanden elektromechanische Verschlüsselungsgeräte, sog. Rotormaschinen, die besonders im zweiten Weltkrieg Verwendung fanden. Die Walzen der Rotormaschine sind den rein mechanischen Chiffrierwalzen sehr ähnlich, besitzen aber an beiden Seiten Kontaktflächen, die im Inneren der Walze miteinander unterschiedlich verbunden sind. Die Walzen werden so angebracht, daß ihre den Strom leitenden Kontakte aneinander liegen. Wird nun an die äußerste Walze eine Spannung angelegt, so wird über die Kontakte der einzelnen Walzen der Strom geleitet. An den Kontakten der letzten Walze werden Glühlampen angeschlossen, welche die einzelnen Buchstaben repräsentieren. Durch die Stellung der Walzen wird eine polyalphabetische Substitution vorgegeben.

Nun würde die bloße Kombination von mehreren Walzen auch wieder nur eine Substitution ergeben. Daher wird das Verfahren erweitert, indem jede

Walze nach einem Chiffrierschritt ein Stück weitergedreht wird, dann ergibt sich auch mit jedem Schritt eine neue Substitution.

Die wohl bekannteste Rotormaschine, die nach diesem Prinzip arbeitete, ist die ENIGMA der deutschen Wehrmacht. Hier wurden in der Regel vier Rotoren eingesetzt, teilweise auch drei Rotoren mit einem zusätzlichen sog. Reflektor, einer Umkehrwalze, die nur an einer Seite mit Kontakten versehen war und den Strom wieder zurückleitet, so daß die vorgeschalteten Walzen erneut durchlaufen wurden. Insgesamt standen acht Rotoren zur Verfügung (zu Beginn der Entwicklung nur fünf Rotoren), die in beliebiger Reihenfolge kombiniert werden konnten, sowie der Reflektor. In späteren Versionen, die ab 1942 eingesetzt wurden, stand ein zusätzliches Steckbrett zur Verfügung, mit dem einzelne Buchstaben nochmals vertauscht werden konnten.

3.2 Kryptoanalyse historischer Verfahren

Während die antiken Verfahren noch ohne großen Aufwand durch einfaches Ausprobieren gebrochen werden konnten, stellten die polyalphabetischen Verfahren schon eine stärkere Herausforderung dar. Doch selbst die Verschlüsselung komplizierter Rotormaschinen wurde durch Kryptoanalyse gebrochen. Im Folgenden werden die zugrundeliegenden Ideen kurz skizziert, die ENIGMA dient als Beispiel.

Da nie auszuschliessen ist, daß eine Verschlüsselungsmaschine in falsche Hände gerät, darf die Stärke eines Verfahrens nicht auf seiner direkten Geheimhaltung beruhen. Das Gerät, oder abstrakter formuliert, die Anweisung zur Durchführung der Verschlüsselung kann dem Gegner bekannt sein. Im zweiten Weltkrieg wurden beispielsweise gezielt kleinere Schiffe angegriffen, die zur Wetterbeobachtung eingesetzt waren. Obgleich sie kein militärisches Ziel im eigentlichen Sinne darstellten, wurden die

Angriffe durchgeführt, um die an Bord befindlichen ENIGMA-Geräte zu erbeuten, mit denen die Wetterdaten verschlüsselt wurden.

Basiert die Verschlüsselung lediglich auf der Stärke des verwendeten Algorithmus, so spricht man von einem *eingeschränkten Algorithmus*, der geheim gehalten werden muß. Aus der Pflicht zur Geheimhaltung ergeben sich jedoch gravierende Nachteile. Zum einen kann diese Geheimhaltung in einem unsicheren Umfeld nicht garantiert, zum anderen die Qualität des Algorithmus nicht überprüft werden.

Die genannte Problematik kann durch die Verwendung eines zusätzlichen Datenelementes im Algorithmus vermieden werden. Dieses Element wird als *Schlüssel* bezeichnet und beeinflußt die Art und Weise, wie die Operationen Substitution und Transposition auf den Klartext angewandt werden. Die Stärke eines modernen kryptographischen Algorithmus beruht nicht mehr auf seiner Geheimhaltung, sondern nur noch auf der Geheimhaltung des verwendeten Schlüssels. Die völlige Unabhängigkeit von Verschlüsselungsmechanismus und Schlüssel ist eine Anforderung an jedes Kryptosystem. Alle modernen Verfahren arbeiten mit dieser Trennung.

Im Falle der ENIGMA besteht der Schlüssel aus der Auswahl und Anordnung der Rotoren, ihrer Anfangsstellung und gegebenenfalls aus der Anordnung der Stecker auf dem Steckbrett. Diese auch *Grundstellung* genannte Information änderte sich täglich und dürfte nur den beiden an der Kommunikation beteiligten Parteien bekannt sein. Sie wurde daher in ein Codebuch eingetragen, welches beiden Kommunikationspartnern zur Verfügung stand. Da insbesondere U-Boote längere Zeit auf See waren, führten sie Codebücher für mehrere Monate mit sich. Obgleich diese Codebücher auf wasserlöslichem Papier gedruckt wurden und höchster Geheimhaltung unterlagen, gerieten sie doch in die Hände der Alliierten. Damit konnte während der Gültigkeitsdauer des Codebuches die

übertragene Information mitgelesen werden. Da einzelne Schiffe und Heereseinheiten jedoch über eigene Codebücher verfügten, war nur die Informationsübermittlung zwischen dem Oberkommando und diesen Einheiten betroffen.

Als zusätzliche Sicherheit wurde ein sog. Spruchschlüssel eingeführt, eine vom Funker ausgewählte aus drei Buchstaben bestehende Zeichenkette, die jedem Funkspruch vorangestellt wurde. Diese Zeichenkette wurde mit der Grundstellung chiffriert, die weiteren Teile der Nachricht dann mit dem Spruchschlüssel.

Doch auch ohne Kenntnis des Codebuches und des Spruchschlüssels wurde die ENIGMA gebrochen. Der erste und notwendige Schritt bestand darin, daß den Alliierten eine vollständige ENIGMA in die Hände fiel, die vom polnischen Zoll abgefangen wurde. Später folgten weitere; damit war der Aufbau und die Arbeitsweise des Gerätes, oder abstrakter formuliert, der Algorithmus bekannt und konnte auf Schwächen untersucht werden.

Eine besondere Schwäche bestand in der Verwendung der Reflektorscheibe. Mit ihr sollte die Sicherheit der ENIGMA dahingehend verbessert werden, daß die Zahl der durchgeführten Substitutionen erhöht wurde. Der Nachteil bestand jedoch darin, daß durch Einsatz der Reflektorscheibe niemals ein Buchstabe in sich selbst übergehen konnte, da dies einen Kurzschluß verursacht hätte. Bei langen Zeichenketten, die häufig in Texten vorkamen, etwa "DASOBERKOMMANDOGIBTBEKANNT", konnte durch dieses Ausschlußkriterium die wahrscheinliche Position der Zeichenkette im Text bestimmt werden und das Chiffrat mit dem vermuteten Klartext verglichen werden.

Auch andere Standardformulierungen erleichterten die Analyse. Die Texte begannen meist mit "AN " und endeten mit "HEILHITLER". Dies erleichterte

einen *Klartextangriff*, bei dem der Inhalt des Chiffrats teilweise bekannt ist oder erraten werden kann. Der direkte Vergleich zwischen Chiffrat und vermutetem oder bekanntem Klartext läßt Rückschlüsse auf den verwendeten Schlüssel zu. Noch wirkungsvoller ist die Kryptoanalyse, wenn es gelingt, dem Gegner eine vorher definierte Zeichenkette als Klartext vorzugeben. Die Alliierten erreichten dies durch gezielte Militäraktionen, z.B. durch die Bombardierung einer Leuchttonne auf See, nur um die Zeichenkette "LEUCHTTONNE" im kurz darauf gesendeten Funkspruch als Klartext zu unterlegen.

Weitere Angriffspunkte lieferten Chiffrierfehler, d.h. Fehler des Funkers. So wurden von den deutschen Funkern überwiegend einfach zu merkende Spruchschlüssel verwendet, z.B. "XYZ", "AUF" oder "VON". Dies führte zu einem eingeschränkten Schlüsselraum, d.h. nicht alle möglichen Schlüssel wurden eingesetzt und einige Schlüssel waren überrepräsentiert.

Auch wenn die Hauptangriffspunkte gegen die ENIGMA in den genannten Schwächen begründet waren, wurden auch andere Ansätze verfolgt. Ein sehr viel allgemeinerer Ansatz führt über die Häufigkeitsverteilung der Buchstaben der deutschen Sprache. Wenn bekannt ist, daß die Buchstaben "e" und "n" am häufigsten vorkommen, die Buchstaben "q", "x" und "y" hingegen sehr selten, dann kann der Chiffretext auf diese Häufigkeitsverteilung hin untersucht werden, um Zuordnungen zwischen Chiffretext- und Klartextzeichen herzustellen. Einfache Substitutions- und Transpositionschiffren sind hier besonders anfällig.

Die Wahrscheinlichkeit eines erfolgreichen Angriffes über die Häufigkeitsverteilung sinkt, wenn die Zeichen des Chiffretextes annähernd gleich verteilt sind. Schon 1949 hat Claude Shannon in seiner vielbeachteten Arbeit *„Communication Theory of Secrecy Systems“,* die u.a. in [Sloane/Wyner 1993] veröffentlicht ist, zwei grundlegende Prinzipien der

Verschlüsselung beschrieben: *Konfusion* und *Diffusion*. Mittels der *Konfusion* wird der Zusammenhang zwischen Klartext und Chiffretext verschleiert, so daß keine eindeutige Zuordnung zwischen den Klartextzeichen und Chiffretextzeichen mehr möglich ist. Durch *Diffusion* werden die im Klartext enthaltenen Informationen gleichmäßig über den Chiffretext verteilt, um Angriffe, die auf Häufigkeitsverteilung beruhen, zu erschweren. Beide Prinzipien finden ihre Anwendung in gängigen Algorithmen.

Der allgemeine Vorgang der Chiffrierung und Dechiffrierung unter Einsatz eines kryptographischen Algorithmus und eines Schlüssels wird in Abbildung 3.1 dargestellt.

.

Abbildung 3.1: Vorgang der Ver- und Entschlüsselung

Sind die zur Ver- und Entschlüsselung verwendeten Schlüssel identisch, so handelt es sich um einen *symmetrischen Algorithmus*, sind sie verschieden, handelt es sich um einem *asymmetrischen Algorithmus*. Beide Arten von Algorithmen werden in den nächsten Abschnitten erläutert.

3.3 Symmetrische Algorithmen

Wird nur ein Schlüssel eingesetzt, spricht man von einem *konventionellen* oder *symmetrischen* Verfahren. Abbildung 3.2 zeigt den Aufbau und Ablauf einer symmetrischen Verschlüsselung.

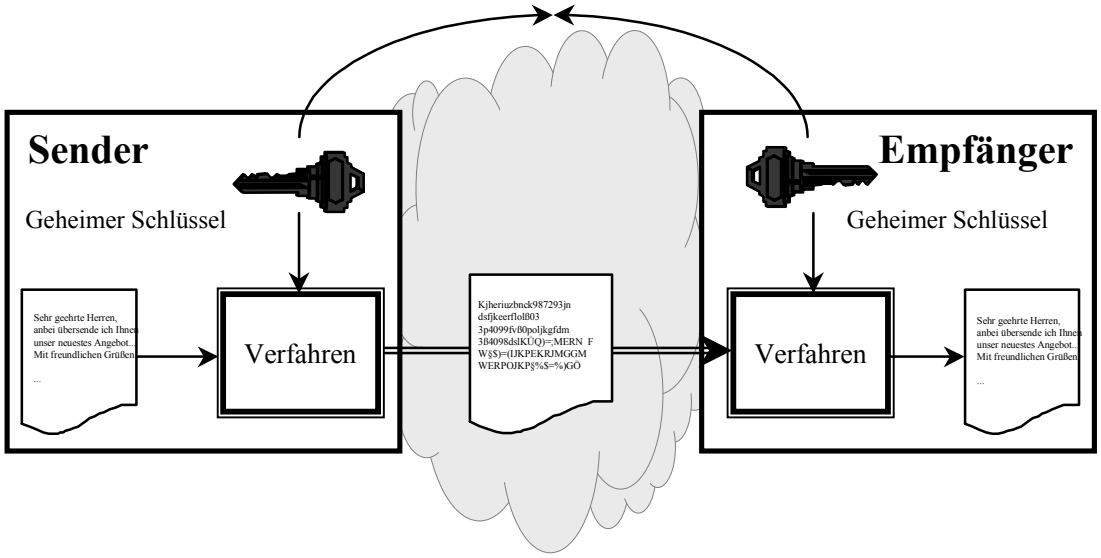


Abbildung 3.2: Symmetrisches Verfahren

Symmetrische Verfahren können als Block- oder Stromchiffrierung durchgeführt werden. Bei einer *Blockchiffrierung* wird jeder Teil des Klartextes mit einer gleichbleibenden Operation in einen Chiffretext gleicher Länge transformiert. Die *Stromchiffrierung* hingegen behandelt den Klartext als einen konstanten Eingabestrom, der mit einer sich ständig verändernden Operation in einen Ausgabestrom übersetzt wird. Dazu wird ein Schlüsselstrom erzeugt, der mit dem Klartext verknüpft wird.

Ein entscheidender Nachteil der Blockchiffrierung besteht darin, daß ein Klartextblock bei gleichbleibendem Schlüssel immer in denselben Chiffretextblock ungewandelt wird. Um die Sicherheit zu erhöhen, kann die Art der Ausführung einer Blockchiffre durch die Anwendung bestimmter

Betriebsmodi des Algorithmus abgeändert werden. Die wichtigsten heute eingesetzten Modi für Blockchiffren sind:

- *Electronic-Codebook-Modus* (ECB): Hier wird jeder Block unabhängig von allen anderen Blöcken des Klartextes verschlüsselt.

- *Cipher-Block-Chaining* (CBC): In dieser Betriebsart fließt das Ergebnis der Verschlüsselung eines vorangehenden Blockes durch XOR-Verknüpfung in die Verschlüsselung des aktuellen Blocks ein.

- *Output-Feedback-Modus* (OFB): Das Ergebnis einer Verschlüsselung wird als Zufallsgenerator eingesetzt (s.u.) und bestimmt das Ergebnis der weiteren Operationen.

- *Cipher-Feedback-Modus* (CFM): Dieser Modus ähnelt dem Output-Feedback-Modus, eine Rück- und Vorwärtskopplung des Chiffretextes stärkt jedoch das Verfahren.

Es existieren noch einige weitere Modi (vgl. [Schneier 1996, Kap. 9.10]), die in der Praxis jedoch kaum Anwendung finden.

Die bekanntesten auf symmetrischen Verfahren basierenden Algorithmen sind:

- der *Data Encryption Standard* (DES). DES wurde ursprünglich von IBM entwickelt und später von der US-Regierung zum Standard erklärt. Es handelt sich bei diesem Algorithmus um eine 64 Bit Blockchiffre, die einen Schlüssel von 56 Bit verwendet. Der Algorithmus wurde oftmals abgewandelt, um seine Sicherheit zu erhöhen. Die bekannteste Abwandlung ist der Triple-DES, bei der

DES dreimal hintereinander auf einen Klartextblock angewendet wird. Hier werden zwei konventionelle Schlüssel mit einer Gesamtlänge von 112 Bit eingesetzt.

- der *International Data Encryption Algorithm* (IDEA). Dieser Algorithmus wurde von den Kryptoanalytikern Lai und Massey entwickelt (vgl. [Lai/Massey 1991]) und arbeitet ebenfalls mit einer Blocklänge von 64 Bit. IDEA wendet jedoch einen Schlüssel mit 128 Bit an und ist etwa doppelt so schnell wie DES. Aus diesem Grund wird der Algorithmus zunehmend implementiert.

- die *Rivest Cipher 4* (RC4), eine Stromchiffre, die 1987 von Ron Rivest entworfen wurde. Der Algorithmus konnte einige Jahre geheim gehalten werden und wurde erst 1994 im Internet veröffentlicht. RC4 arbeitet theoretisch mit variabler Schlüssellänge, in vielen Fällen wird jedoch eine Schlüssellänge von 128 Bit verwendet. Die US-Regierung erlaubt den Export von Produkten, die auf RC4 basieren nur, sofern die Schlüssellänge 40 Bit nicht übersteigt.

- der *Advanced Encrypton Standard* (AES), ein Gemeinschaftswerk der beiden belgischen Computerwissenschaftler Joan Daemen und Vincent Rijmen. AES wurde in einem Wettbewerb im Oktober 2000 als Nachfolger des *Data Encryption Standard* (DES) ermittelt und arbeitet mit einer Schlüssellänge von 128 Bit. Die häufigste Einsatzart ist der CBC-Modus.

Bei allen symmetrischen Verschlüsselungsverfahren ergibt sich ein entscheidendes Problem: Ein geheimer Schlüssel muß übertragen werden. Geht man davon aus, daß mehrere Personen oder Personengruppen

miteinander kommunizieren wollen, und daß die Schlüssel aus Sicherheitsgründen regelmäßig geändert werden, dann steigt die Zahl der verwendeten Schlüssel schnell an.

Die Frage, wie diese Schlüssel sicher ausgetauscht werden können, wird in Abschnitt 3.4 angesprochen. Asymmetrische Algorithmen, deren Aufbau und Funktion im folgenden Abschnitt erläutert werden, leisten ihren Beitrag zur Lösung des Problems.

3.4 Asymmetrische Algorithmen

Asymmetrische Verfahren verwenden zur Ver- und Entschlüsselung zwei verschiedene Schlüssel, die jedoch gleichartige Eigenschaften aufweisen. Die Kryptographen Diffie und Hellman veröffentlichten 1976 erstmals ein Verfahren, das den Einsatz zweier Schlüssel beschrieb. Einer dieser Schlüssel ist geheim, der andere öffentlich. Die Schlüssel sind so beschaffen, daß Informationen, die mit dem öffentlichen Schlüssel chiffriert worden sind, nur mit dem geheimen oder „privaten“ Schlüssel wieder entschlüsselt werden können (vgl. Abbildung 3.3).

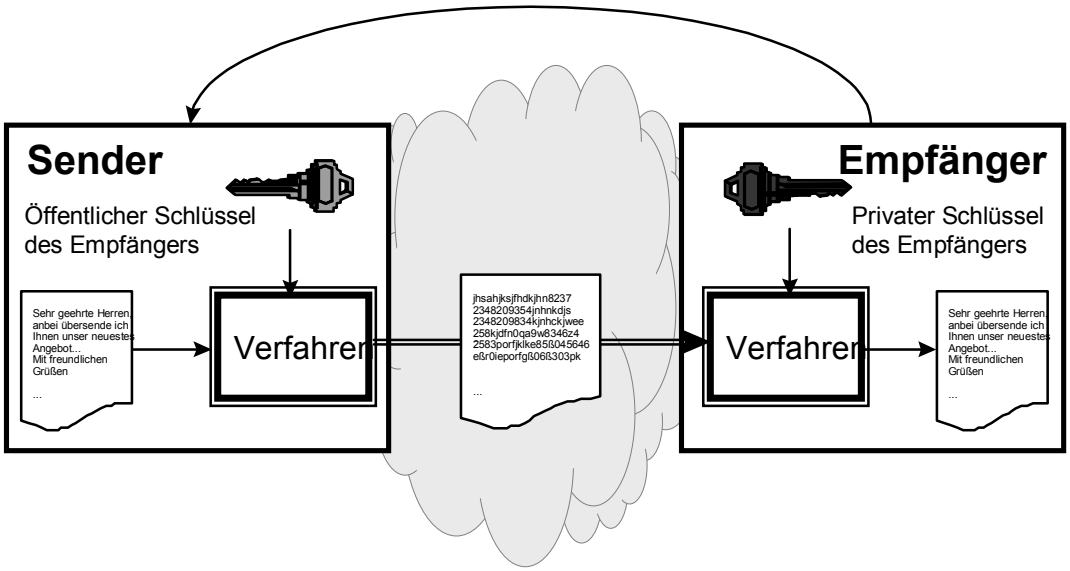


Abbildung 3.3: Asymmetrisches Verfahren

Der mit Abstand bekannteste asymmetrische Algorithmus ist RSA, benannt nach seinen Entwicklern Rivest, Shamir und Adleman. RSA war das erste asymmetrische Verfahren, das sowohl zur Verschlüsselung als auch zur digitalen Signatur (s.u.) eingesetzt werden konnte. Die von den drei Entwicklern gegründete Firma RSA Data Security veröffentlichte einige Public Key Standards (PKCS), die u.a. den Einsatz von RSA definieren und zunehmend Verbreitung finden.

Das von Diffie und Hellman entwickelte Verfahren entspricht in seiner Leistungsfähigkeit dem RSA-Verfahren, beruht jedoch auf einem anderen mathematischen Prinzip. Weitere bekannte Verfahren, die jedoch wesentlich seltener eingesetzt werden, sind ElGamal und Rabin (vgl. [Schneier 1996, S.607ff]).

3.5 Hybride Verfahren

Ein wesentlicher Nachteil asymmetrischer Kryptographie liegt in ihrer Geschwindigkeit. Asymmetrische Verfahren sind in der Regel mindestens um den Faktor 100 langsamer als symmetrische und eignen sie sich daher nicht zur Verschlüsselung großer Datenmengen. Dieses Problem kann umgangen werden, indem bei einem höheren Datenvolumen zunächst die schnelle symmetrische Kryptographie und dann erst die langsame asymmetrische eingesetzt wird. Im ersten Schritt wird ein zufällig generierter Sitzungsschlüssel (ein sog. *session key*) erzeugt und in einem symmetrischen Verfahren auf die Daten angewandt. Der Schlüssel selbst wird dann mittels asymmetrischer Kryptographie verschlüsselt (mit dem öffentlichen Schlüssel des Empfängers der Daten) und zusammen mit den Daten übermittelt. Der Empfänger der Daten kann mit seinem privaten Schlüssel den *session key* ermitteln und erneut auf die Daten anwenden, um den Klartext wieder herzustellen. Dieses zweistufige Verfahren verbindet die hohe Geschwindigkeit der symmetrischen Kryptographie mit den Vorteilen asymmetrischer Verfahren im Hinblick auf die sichere Verteilung von Schlüsseln. Damit asymmetrische Verfahren zur digitalen Signatur (s.u.) eingesetzt werden können, werden sie mit den im nächsten Abschnitt dargestellten Hashfunktionen verbunden.

3.6 Einweg-Hashfunktionen

Einweg-Hashfunktionen erzeugen aus einer Nachricht beliebiger Länge einen Hashwert fester Länge. Dies ist zunächst nichts ungewöhnliches. Das besondere Charakteristikum der Einweg-Hashfunktion besteht in zwei Eigenschaften. Erstens erlaubt die Funktion keinerlei Rückschluß auf die Eingabedaten, zweitens ist es mit vertretbarem Aufwand unmöglich, eine zweite Nachricht zu konstruieren, die denselben Hashwert besitzt. Dieser Umstand wird *Kollisionsresistenz* genannt.

Aufgrund dieser beiden Eigenschaften wird das Ergebnis der Funktion oftmals als *digitaler Fingerabdruck* der Eingabedaten bezeichnet. Die Sicherheit der Einweg-Hashfunktion liegt darin begründet, daß die Funktion des Hashings relativ einfach, ihre Umkehrung jedoch extrem schwierig durchzuführen ist.

Die zur Zeit am häufigsten eingesetzten Verfahren sind:

- Der *Secure-Hashing-Algorithm* (SHA). SHA wurde von der amerikanischen Behörde NIST (National Institute of Standards and Technology) entwickelt und erzeugt einen 160 Bit Hashwert. Bisher sind keine gelungenen Angriffe auf SHA bekannt geworden.

- Der von Ron Rivest 1991 entwickelte *Message Digest 5* (MD5) liefert wie die Vorgängerversionen MD2 und MD4 einen 128 Bit Hashwert. Gegen MD5 sind seit 1995 einige Angriffe bekannt geworden (vgl. [Schneier 1996, S.503]). Die Funktion gilt dennoch als hinreichend sicher.

- Der Algorithmus *RIPE-MD*, die europäische Antwort auf MD5 und den Vorgänger MD4. RIPE-MD erzeugt einen 128 Bit Hashwert durch die Verknüpfung zweier parallel ablaufender Versionen des Algorithmus und gilt ebenfalls als hinreichend sicher.

Einweg-Hashfunktionen werden oft in Verbindung mit anderen Verfahren eingesetzt, um das Sachziel Integrität zu erreichen. Wird über eine Nachricht ein eindeutiger Hash-Wert gebildet und dem Kommunikationspartner sicher übermittelt, kann dieser die Integrität der Nachricht prüfen, indem er selbst erneut einen Hashwert nach demselben Verfahren erzeugt. Stimmen die Werte überein, ist die Nachricht nicht verändert worden.

3.7 Signaturen

Durch die Kombination asymmetrischer Verfahren mit einer Einweg-Hashfunktion kann eine digitale Signatur erzeugt werden. Eine digitale Signatur belegt, daß eine Nachricht oder ein Dokument authentisch ist und auf dem Kommunikationsweg nicht verändert wurde. Um dies zu gewährleisten, wird eine vollständige Nachricht oder ein eindeutiger Hashwert über die Nachricht mit dem privaten Schlüssel des Absenders verschlüsselt. Der Empfänger kann mit dem öffentlichen Schlüssel des Absenders die Nachricht wieder entschlüsseln.

Da asymmetrische Verfahren sehr viel langsamer als symmetrische Verfahren sind, wird in der Regel nicht die gesamte Nachricht, sondern nur ein Hashwert der Nachricht signiert, d.h. mit dem privaten Schlüssel des Absenders verschlüsselt. Aus diesem Grund erfolgt die Signatur einer Nachricht in zwei Schritten.

Im ersten Schritt wird mittels einer Einweg-Hashfunktion ein eindeutiger, nur dieser Nachricht zugeordneter Hashwert erzeugt. Dieser Hashwert wird mit dem privaten Schlüssel des Absenders signiert. Der Empfänger der Nachricht kann nun mit dem öffentlichen Schlüssel des Absenders prüfen, ob die Nachricht tatsächlich vom entsprechenden Absender kommt. Dann bildet der Empfänger erneut den Hashwert über die Nachricht und vergleicht diesen Wert mit dem vom Absender übermittelten Wert. Stimmen beide Werte überein, ist die Nachricht erstens authentisch (da ja nur der Eigentümer des privaten Schlüssels den Hashwert unterzeichnen konnte) und zweitens unverändert, da sich eine Veränderung der Nachricht auch auf den Hashwert ausgewirkt hätte. Abbildung 3.4 stellt den gesamten Vorgang dar.

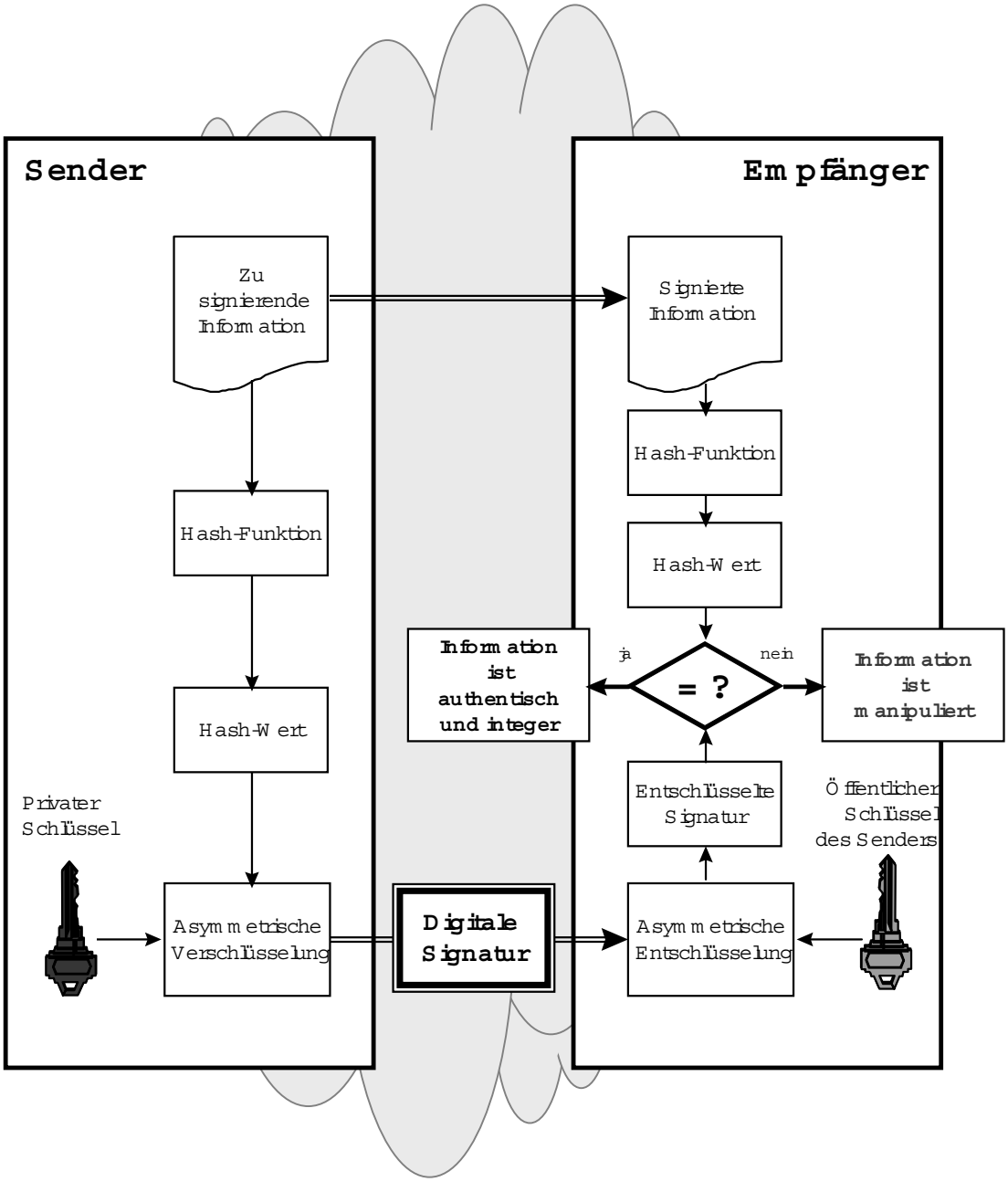


Abbildung 3.4: Ablauf einer digitalen Signatur

Der wichtigste Standard für digitale Signaturen ist zur Zeit der *Digital Signature Standard* des amerikanischen National Institute of Standards and Technology. Dieser Standard definiert einen eigenen Algorithmus, den *Digital Signature Algorithm* (DSA). Obgleich DSA wesentlich langsamer als der ebenfalls eingesetzte RSA-Algotithmus (s.o.) ist und nur zur Signatur eingesetzt werden kann, wird er zunehmend implementiert.

3.8 Zertifikate

Das größte Problem in der Anwendung asymmetrischer Algorithmen besteht darin, die Zuordnung des öffentlichen Schlüssels zu einer bestimmten Person sicherzustellen. Zu diesem Zweck werden Zertifikate eingesetzt. Ein Zertifikat belegt, daß ein öffentlicher Schlüssel zu einer ganz bestimmten Person gehört. Das Zertifikat umfaßt die digitale Signatur eines Namens, des zugehörigen öffentlichen Schlüssels sowie weiterer Merkmale, die eine Person identifizieren, z.B. eine E-Mail Adresse. Mit der Signatur wird die Zuordnung eines öffentlichen Schlüssels vom Unterzeichner beglaubigt.

Zertifikate können von Privatpersonen ausgestellt werden oder von sogenannten *Zertifizierungsstellen* (Certification Authority, kurz CA). Mit Hilfe des öffentlichen Schlüssels einer CA können alle Zertifikate geprüft werden, die von dieser Zertifizierungsstelle unterschrieben wurden. Oftmals entsteht so eine hierarchische Struktur, da der öffentliche Schlüssel einer Zertifizierungsstelle selbst wieder von einer anderen Stelle signiert wird. Der hierarchischen Struktur steht eine Netzwerkstruktur gegenüber, in der jede Person ein Zertifikat ausstellen kann, nicht nur die Zertifizierungsstelle. Diese Struktur wird als *Web of Trust* bezeichnet und z.B. in der Software *Pretty Good Privacy* (PGP) eingesetzt (vgl. [Smith 1998, Kap. 12]).

3.9 Zufallszahlen

Einige kryptographische Verfahren erfordern Zufallszahlen. Die Generierung eines asymmetrischen Schlüsselpaares wird z.B. von einem Programm übernommen, das bei jedem Durchlauf einen anderen Schlüssel erzeugen muß. Ist die Erzeugung eines bestimmten Schlüssels vorhersagbar oder ist die Zahl aller möglichen Schlüssel (der *Schlüsselraum*) sehr klein, dann ist das Verfahren der Schlüsselgenerierung ein möglicher Angriffspunkt gegen das Kryptosystem. Oftmals wird übersehen, daß es für einen Angreifer um vieles leichter sein kann, alle möglichen Schlüssel zu generieren und auf ein Chiffrat anzuwenden, als den Verschlüsselungsalgorithmus selbst zu brechen.

Wie kommt man aber mittels eines Computers, einer von Grund auf deterministischen Maschine, zu echten Zufallszahlen? Es gibt sogenannte Pseudo-Zufallsgeneratoren, die schwer vorhersagbare Zahlenfolgen generieren. Diese Zahlenfolgen sehen zufällig aus, sind aber periodisch, wenn auch mit sehr großen Perioden. Eine Möglichkeit, bessere Zufallsfolgen zu erzeugen, besteht in der Einbeziehung von Ereignissen, die außerhalb des Computers stattfinden (z.B. das Hintergrundrauschen oder die Zeitspanne zwischen zwei Tastendrücken). Ob diese Ereignisse tatsächlich als zufällig angesehen werden, hängt von der Perspektive des Betrachters ab. Im Rahmen dieser Arbeit soll gelten: Eine im kryptographischen Sinne sichere Zufallsfolge ist eine Folge, die weder vorhersehbar noch reproduzierbar ist, selbst wenn der verwendete Algorithmus und die verarbeiteten Ereignisse bekannt sind.

3.10 Key Escrow und Key Recovery

Bei symmetrischer Verschlüsselung wird der verwendete Schlüssel häufig gewechselt, um sicherzustellen, daß die Kompromittierung des Schlüssels (die Kenntnisnahme durch unberechtigte Dritte), mit dem eine Nachricht codiert wurde, nicht automatisch auch die Entschlüsselung weiterer (vorangegangener oder nachfolgender) Nachrichten ermöglicht. Die dabei anfallende große Zahl von Schlüsseln, die in symmetrischen Verfahren eingesetzt wurden (sog. *Konzelationsschlüssel*), erfordert ein effizientes Schlüsselmanagement.

Das *Schlüsselmanagement* umfaßt mindestens die Generierung, Distribution, Prüfung, Archivierung und die Wiedergewinnung (Retrieval) von Konzelationsschlüsseln. Die sichere Archivierung dieser Schlüssel und das Retrieval durch Berechtigte (Einzelpersonen, Unternehmen, Behörden etc.) fällt in den Bereich *Key Escrow* und *Key Recovery*, die anderen Aufgaben fallen in den Bereich eines Key Generation Servers oder eines Key Distribution Centers.

Unter Key Escrow und Key Recovery wird in einer sehr allgemeinen Definition die sichere Hinterlegung eines Schlüssels auf einem Archivserver und dessen Wiedergewinnung verstanden. Durch Hinterlegung des Konzelationsschlüssels kann sichergestellt werden, daß dieser Schlüssel ständig verfügbar ist, auch wenn ein Mitarbeiter eines Unternehmens Dokumente verschlüsselt hat, und das Unternehmen bei Abwesenheit des Mitarbeiters auf diese Dokumente zugreifen muß. Um den Bereich Key Recovery ist eine politische Diskussion entbrannt, da einige Implementierungen von Key Recovery Servern auch einen Zugriff auf die hinterlegten Schlüssel durch Strafverfolgungsbehörden und Regierungsstellen vorsehen. Es ist jedoch durchaus möglich, Key Recovery

auch ohne diese Möglichkeit zu realisieren. Abbildung 3.5 verdeutlicht den Prozeß der Hinterlegung eines Schlüssels auf einem Speicherserver und Abbildung 3.6 die Wiedergewinnung des Schlüssels.

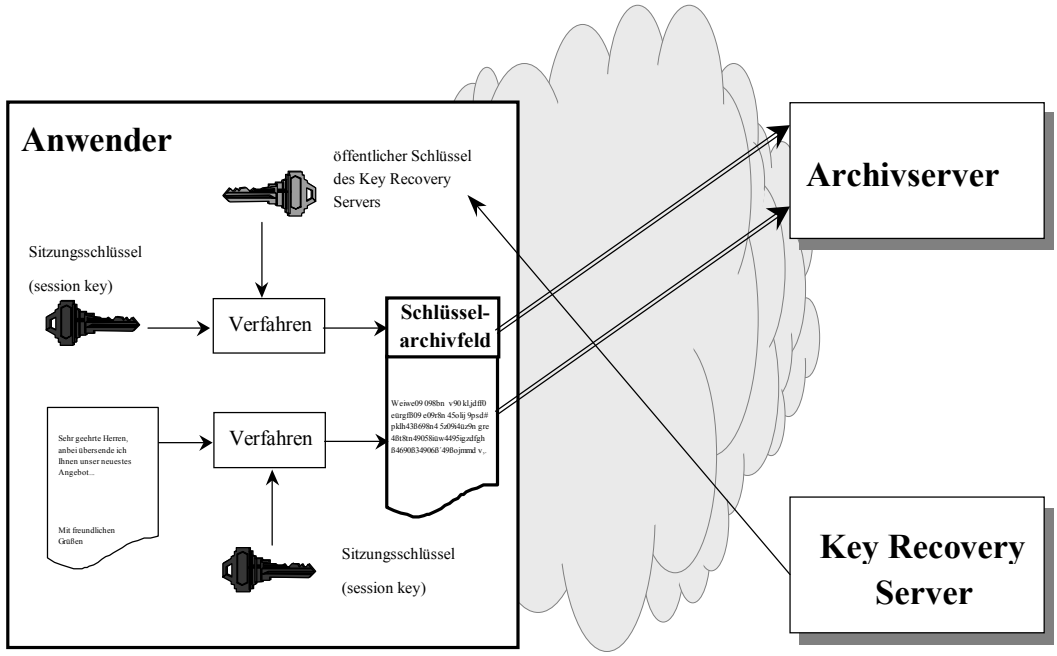


Abbildung 3.5: Ablage eines Schlüssels beim Key Recovery Verfahren

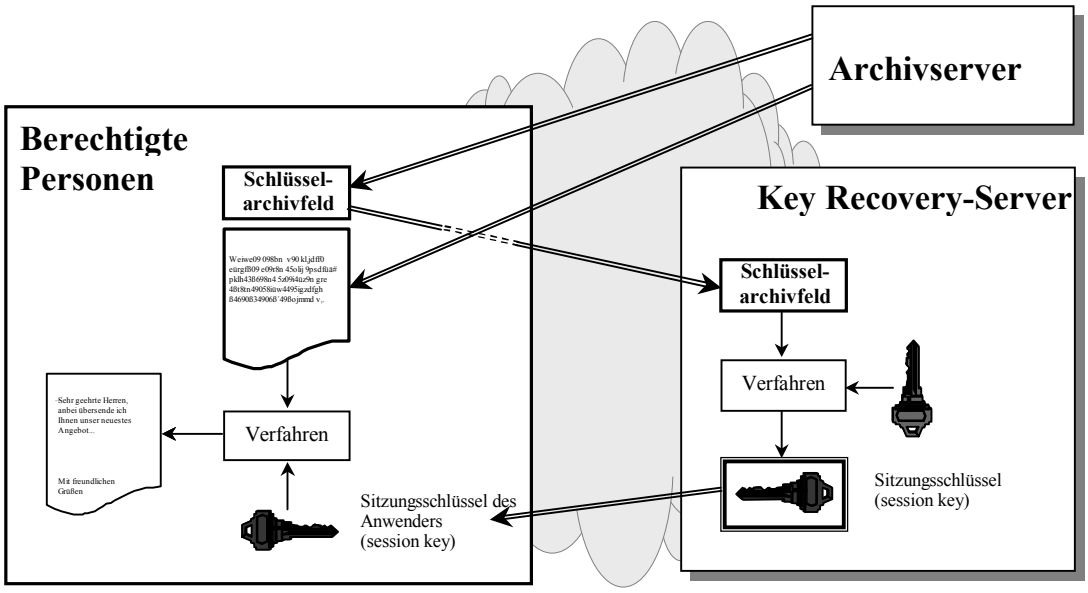


Abbildung 3.6: Retrieval des Schlüssels durch Berechtigte

3.11 Public Key Infrastructure

Die zuvor genannten Mechanismen und Verfahren können miteinander kombiniert werden. Ziel ist die Erstellung einer vollständigen Infrastruktur rund um die digitalen Signatur. Um die Signatur wirksam und rechtsverbindlich einsetzen zu können, sind neben den eigentlichen asymmetrischen Algorithmen und Einweg-Hashfunktionen weitere Komponenten erforderlich.

Eine Public Key Infrastructure besteht aus den folgenden Elementen:

- dem *Key-Distribution-Center* (KDC), einer Instanz zur sicheren Erzeugung und Verteilung von Schlüsseln. Neben der zentralen Generierung von Schlüsseln mittels eines KDC ist auch die Generierung auf dem Rechner des Anwenders möglich.

- der *Certification Authority* (CA) zur Erzeugung und Prüfung von Zertifikaten. Abhängig von der Sicherheitsstufe können verschiedene Attribute vergeben werden, die kennzeichnen, welche Identitätsprüfung vor Ausgabe des Zertifikates erfolgt ist. Certification Authorities können hierarchisch organisiert sein und implementieren dann eine sog. *Chain of Trust* (vgl. [Fuhrberg 1998] S.96ff).

- der *Registration Authority* (RA) zur Interaktion mit der CA. Hier erfolgt die Registrierung von Benutzern und die Weiterleitung einer Zertifikatsanforderung an die CA. Beide Komponenten können gemeinsam auf einem System oder verteilt betrieben werden. Diese Auswahl hat jedoch Auswirkungen auf das Sicherheitsniveau (vgl. [Ellison/Schneier 2000] S.4).

- dem *Directory* (DIR), in dem die Zertifikate und Rückruflisten gespeichert werden. In der Praxis wird hier meist ein LDAP-Server eingesetzt.

- dem *Key-Recovery-Center* (KRC) in dem die in Abschnitt 3.10 beschriebenen Mechanismen zur sicheren Archivierung von Schlüsseln und dem Retrieval durch Berechtigte umgesetzt werden.

- der *Time-Stamping-Authority* (TSA). Mit diesem Dienst können Signaturen mit Zeitstempeln versehen werden. So wird sichergestellt, daß eine Signatur zu einem bestimmten Zeitpunkt erfolgt ist.

Nicht Teil der eigentlichen Public Key Infrastructure sind die Applikationen, die digitale Signatur und Zertifikate nutzen. Aufgrund der zum Aufbau und Betrieb erforderlichen Ressourcen ist die Verbreitung von Public Key Infrastructures derzeit noch gering.

3.12 Sicherheitsaspekte

Die Sicherheit eines Kryptosystems hängt nicht allein vom Einsatz eines starken Verschlüsselungsalgorithmus ab. Immer wieder haben Kryptographen vermeintlich sichere Algorithmen entworfen, die schließlich an vorher nicht absehbaren Faktoren gescheitert sind.

Neben einem starken Algorithmus, der gründlich kryptoanalysiert wurde (und folglich auch nicht geheimgehalten werden darf), spielen weitere Faktoren beim Bau eines Kryptosystems eine Rolle. Ein wichtiger Faktor ist die Länge des verwendeten Schlüssels. Symmetrische Verfahren arbeiten üblicherweise mit Schlüssellängen zwischen 56 und 128 Bit, asymmetrische

Verfahren mit Schlüssellängen zwischen 512 und 2048 Bit. Weiterhin darf der Schlüsselraum, d.h. die Menge der tatsächlich verwendeten Schlüssel, nicht durch unsaubere Implementierungen eingeschränkt werden, z.B. durch einen Zufallszahlengenerator, der nur einen kleinen Teil des Schlüsselraumes ausschöpft.

Auch wenn der Algorithmus stark und die Schlüssellänge angemessen ist, sind noch verschiedene Angriffe auf Kryptosysteme möglich, u.a.:

- Die *Exhaustionsmethode* (brute force attack), das Durchprobieren aller möglichen Schlüssel innerhalb des Schlüsselraumes. Dieser Angriff führt theoretisch immer zum Ziel, ist jedoch in der Praxis meist nicht möglich, da Zeit und Hardware-Ressourcen nicht hinreichen. Statistisch gesehen führt diese Methode nach dem Austesten der Hälfte aller möglichen Schlüssel zum Ziel, Zufallstreffer können den Weg aber erheblich abkürzen.

- Der *Klartextangriff* (known plaintext attack), bei dem der Inhalt des Chiffrats teilweise bekannt ist, oder zumindest ein Muster vermutet werden kann. Diese Angriffsmethode hat verschiedene Spielarten je nachdem, ob und in welchem Maße es dem Angreifer möglich ist, selbständig neue Chiffrate mit gewähltem Klartext zu erzeugen (choosen plaintext attack).

Weitere, seltener eingesetzte Angriffsmethoden werden in [Schneier 1996, S. 416ff] und [Wobst 1997, Kap. 5.8] beschrieben.

3.13 Erreichung der Sachziele sicherer Kommunikation

Was leisten die einzelnen Verfahren der Kryptographie, und welche der in Kapitel 2.2 erläuterten Primärzeile können erreicht werden? Durch symmetrische Verschlüsselung wird Vertraulichkeit möglich. Der Inhalt symmetrisch verschlüsselter Kommunikation bleibt für unberechtigte Dritte verborgen. Authentizität wird jedoch bei symmetrischer Verschlüsselung selbst nicht gewährleistet. Es kann zu keiner Zeit sichergestellt werden, daß nur berechtigte Kommunikationspartner im Besitz eines Schlüssels für diese Verfahren sind. Darüber hinaus kann ein Angreifer auch Teile der Kommunikation erneut einspielen (*replay attack*) oder während der Übertragung modifizieren (*man-in-the-middle attack*). Werden Blockchiffren im Zusammenhang mit der symmetrischen Verschlüsselung eingesetzt, so ist es für den Angreifer möglich, einzelne Teilstücke der übertragenen Daten gegeneinander auszutauschen, die Manipulation ist nur über den Inhalt einer Nachricht festzustellen.

Integrität und Authentizität werden daher nur durch digitale Signaturen sichergestellt. Ein Einweg-Hashverfahren mit hoher Kollisionsresistenz garantiert, daß die Veränderung der Daten sich in einer Veränderung des Hashwertes widerspiegelt. Die asymmetrische Verschlüsselung, d.h. die eigentliche Signatur belegt, daß derjenige, der sie ausführt, sich im Besitz des privaten Signaturschlüssels befindet.

Es darf zu keiner Zeit außer acht gelassen werden, daß auch mittels digitaler Signaturen und symmetrischer Verschlüsselung nicht immer die genannten Sachziele erreicht werden können.

Oft scheitert sichere Kommunikation im Internet aufgrund von Faktoren, die außerhalb der Kryptographie liegen. Bevor diese Faktoren dargestellt werden können, ist es nötig, zunächst ein Blick auf das Internet und seine Kommunikationsstrukturen zu werfen, die später durch den Einsatz von Kryptographie abgesichert werden sollen.

4 Computernetze und das Internet

4.1 Netzwerke und das OSI-Modell

Mit der Abkehr von einer zentralen Datenverarbeitung hin zu verteilten Systemen wuchs die Bedeutung von Netzwerken, die diese Systeme miteinander verbinden. Kleineren Einheiten, sogenannte *Local Area Networks* (LAN), die sich über einen kleinen räumlichen Bereich, z.B. ein Firmengelände erstrecken, werden zu größeren Einheiten zusammengefügt. Als *Wide Area Network* (WAN) verbinden diese Einheiten größere Bereiche innerhalb eines Landes miteinander, die sich als *Global Area Network* (GAN) schließlich weltweit ausdehnen.

Zwischen einzelnen Einheiten finden die verschiedensten *Übertragungsmedien* (Kupferkabel, Infrarot, Richtfunk usw.) Verwendung. Der Austausch von Daten über diese Medien ist durch *Netzwerkprotokolle* vorgegeben. Protokolle beschreiben Aufbau und Struktur der zu übertragenden Daten sowie die zur Fehlererkennung und Korrektur verwendeten Mechanismen. Um die Vielzahl der existierenden Protokolle einheitlich beschreiben zu können, begann die *International Organisation for Standardization* (ISO) im Jahre 1977 mit der Entwicklung eines Beschreibungsmodells für Netzwerkprotokolle.

Das aus dieser Arbeit entstandene *Open Systems Interconnect Reference Model* (OSI-Modell) besteht aus sieben Schichten und beschreibt grundlegende Funktionen des Netzbetriebes, ohne diese Funktionen jedoch selbst zu realisieren. Einzelne Netzwerkprotokolle können in ihrer Arbeitsweise den Schichten des OSI-Modells zugeordnet und damit eindeutig beschrieben werden.

Protokolle, die eine bestimmte Schicht des OSI-Modells auf einem Rechner implementieren, bauen eine Verbindung zur korrespondierenden Schicht auf einem anderen Rechner auf. Dabei werden die einzelnen Nachrichten über die jeweils darunter liegenden Schichten des Protokolls übertragen. Abbildung 4.1 stellt diesen Ablauf dar.

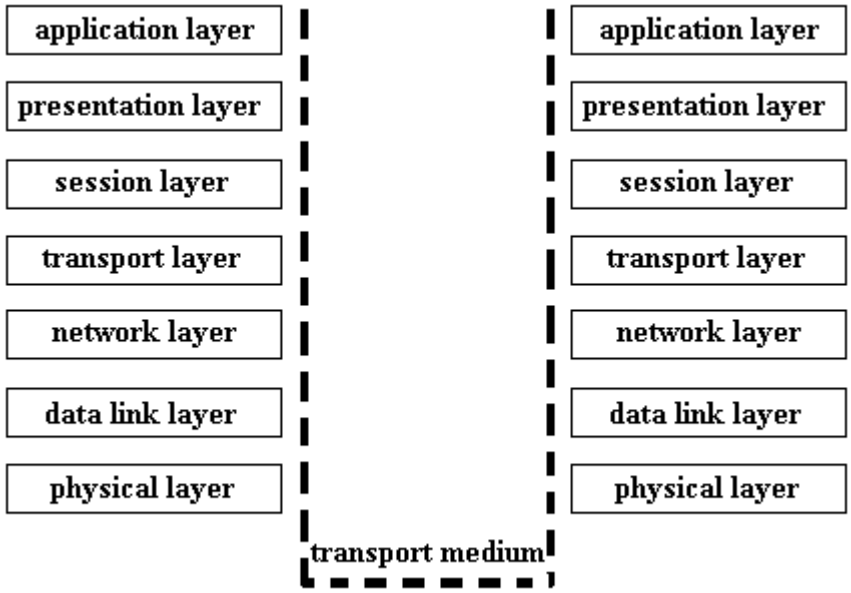


Abbildung 4.1: Datenfluß im OSI-Modell

Jede Schicht fügt Kontrollinformationen zu den übertragenen Daten hinzu, die diese Daten beschreibt und weitere Informationen für angrenzende Schichten enthält, wie in Abbildung 4.2 dargestellt.

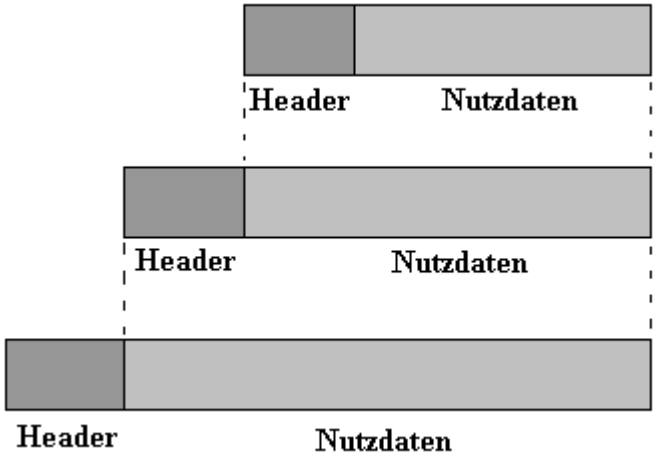


Abbildung 4.2: Protokollstruktur

Schicht 1 (physical layer) definiert Netztopologien, Medientypen und andere Charakteristika der verwendeten Hardware. Die angrenzende Schicht 2 (data link layer) formt Datensegmente aus dem von Schicht 1 erhaltenen Bitstrom und sorgt für die Sicherung der Datenübertragung auf Ebene des physikalischen Netzes. Die Schichten 1 und 2 stellen gemeinsam die physikalische Verbindung zum Netzwerk für die oberen Schichten bereit.

Auf Schicht 3 des OSI-Modells (network layer) werden einzelne Datenpakete (*datagram*) transportiert. Die logische Adressierung der Daten und der Prozeß der Wegwahl (*routing*) findet ebenfalls hier statt. Schicht 4 (transport layer) sorgt für Fehlerkorrektur der auf Schicht 3 übetragenen Daten. Hier werden die einzelnen Datagramme in die korrekte Reihenfolge gebracht und an die nächsthöhere Schicht weitergereicht. Beide Schichten sorgen für den Transport der Daten von Rechner zu Rechner, unabhängig vom zugrundeliegenden (in Schicht 1 und 2 definierten) Netzwerktyp und Übertragungsmedium.

Die Schichten 5 bis 7 stellen gemeinsam Netzwerkdienste für den Benutzer zur Verfügung. Dabei übernimmt Schicht 5 (session layer) die Aufgabe der Dialogkontrolle, Schicht 6 (presentation layer) ist verantwortlich für die

identische Repräsentation der Daten auf verschiedenen Rechnerarchitekturen. Schicht 7 (application layer) stellt die Schnittstelle zum Benutzer dar.

4.2 TCP/IP

Das am weitesten verbreitete Netzwerkprotokoll ist heute das TCP/IP-Protokoll. Die wichtigsten Teilprotokolle, das *Transmission Control Protocol* (TCP) und das *Internet Protocol* (IP) gaben der Protokollfamilie ihren Namen.

TCP/IP kann, wie alle Netzwerkprotokolle, in Aufbau und Funktion durch das im vorigen Abschnitt dargestellte OSI-Modell beschrieben werden. Es ergibt sich die in Abbildung 4.3 dargestellte Zuordnung.

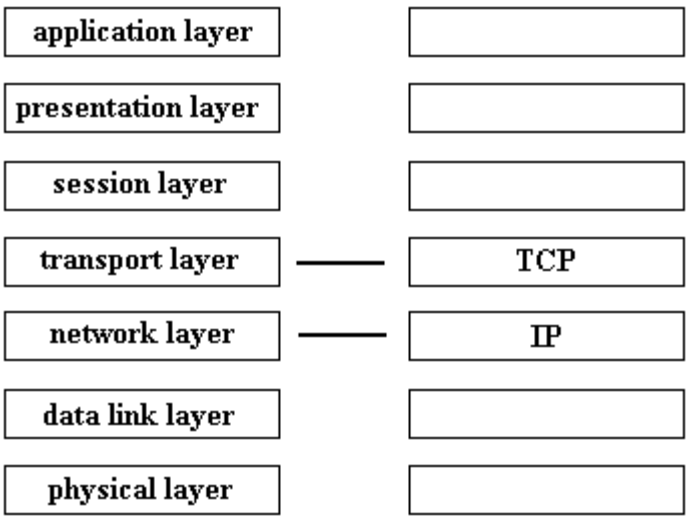


Abbildung 4.3: OSI-Modell mit Zuordnung zu TCP/IP

Da TCP/IP rund eine Dekade älter ist als das Referenzmodell, wurde TCP/IP ursprünglich mit dem Modell des US-Verteidigungsministeriums beschrieben.

Dieses sog. *DoD-Modell* (Department of Defense) besteht aus vier Schichten, die mit ihren Zuordnungen zum neueren OSI-Modell in Abbildung 4.4 dargestellt sind.

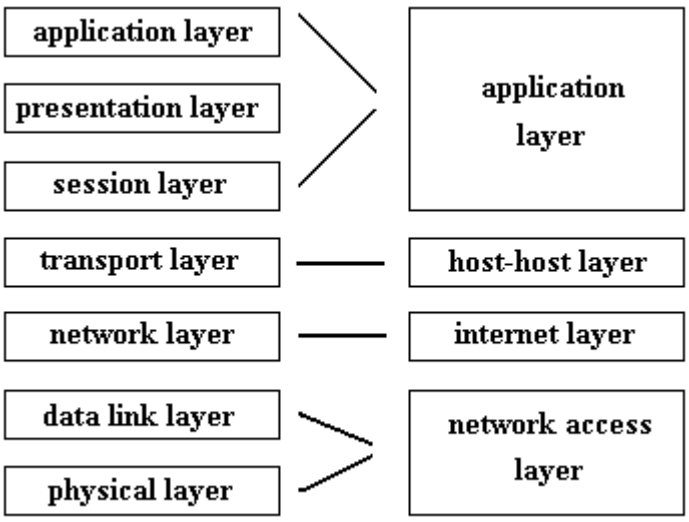


Abbildung 4.4: OSI-Modell mit Zuordnung zum DoD-Modell

Die einzelnen Teilprotokolle von TCP/IP werden in Kapitel 5 zusammen mit den auf TCP/IP aufbauenden Netzdiensten beschrieben.

4.3 Das Internet

Das *Internet* ist ein Zusammenschluß organisatorisch und finanziell voneinander unabhängiger Teilnetze auf Basis des Netzwerkprotokolls TCP/IP. Kommunikation im Internet läuft daher immer über verschiedene Teilprotokolle der Protokollfamilie TCP/IP ab.

Als Global Area Network hebt sich das Internet vom *Intranet* ab, einem ebenfalls TCP/IP basierten Netz, welches jedoch räumlich begrenzt ist, in der Regel als Local Area Network. Verbindungen solcher lokalen Netzwerke

zwischen mehreren Standorten werden meist mit dem Terminus *Extranet* bezeichnet.

Im Internet finden sich keine hierarchischen Strukturen und keine zentrale Kontrolle. Jeder Rechner im Netz kann Dienste anbieten und die Dienste anderer Rechner in Anspruch nehmen. Die Zahl der einzelnen Rechner im Internet steigt ständig an, im Juli 1999 wurde nach Angaben des *Deutschen Network Information Center* (DE-NIC) die Grenze von weltweit 43 Millionen angeschlossenen Rechnern überschritten.

Ein Netzwerk mit diesen Dimensionen ist über lange Zeit gewachsen. Aus einem im Jahre 1969 ins Leben gerufenen Projekt des amerikanischen Verteidigungsministeriums entstand bis 1975 ein einfaches paketvermittelndes Netz zu militärischen Zwecken. Etwas später (1983) wurde dieses Netz auf das neu entwickelte Netzwerkprotokoll TCP/IP umgestellt. Der militärisch genutzte Teil wurde abgespalten. Zu dieser Zeit entstand die Bezeichnung Internet für die Gesamtheit des militärischen und zivilen Teils des Netzes.

Wenn die Fragestellung lautet „Wie können wir sicher über ein unsicheres Netzwerk kommunizieren?“ (unter Einbeziehung der in Abschnitt 2.2 genannten Kriterien), ist es natürlich unabdingbar aufzuzeigen, wie das Internet arbeitet und warum es unsicher ist. In den folgenden Abschnitten werde ich daher die Kommunikationsstrukturen innerhalb des Netzes beschreiben.

5 Kommunikation im Internet

Kommunikation im Internet ist immer Kommunikation über das Protokoll TCP/IP und die von TCP/IP unterstützten Dienste. Um eine systematische Darstellung der einzelnen Abläufe zu geben, werden die jeweiligen Mechanismen anhand der Schichten des OSI-Modells dargestellt.

5.1 Netzwerkzugriff (OSI Schicht 1+2)

In TCP/IP sind die OSI-Schichten 1 und 2 nicht mit Teilprotokollen belegt, vielmehr setzt TCP/IP auf jedes paketvermittelnde physikalische Netz auf. Die Netzwerkzugriffsschichten sind durch verschiedene Hardwareprotokolle realisiert, die im Gegensatz zu den Protokollen der höheren Schichten an die vielen im Internet eingesetzten Übertragungsmedien angepaßt werden können. Diese Protokolle sorgen dafür, daß die zu übertragenden Daten in ein Format umgewandelt werden, mit denen das physikalische Netz arbeiten kann.

Auf dieser Ebene werden verschiedene *Topologien* definiert (Anordnungen von Rechnern in einem Netz), z.B. Ring-, Stern- oder Bus-Topologien. Ebenfalls festgelegt werden Zugriffsverfahren, die den Datenfluß im physikalischen Netz regeln, z.B. CSMA/CD oder TokenRing. Es erfolgt eine Beschreibung einzelner Medien, z.B. Kupferkabel oder Glasfaser, die in verschiedenen Topologien genutzt werden können. Details zu den auf dieser Ebene verwendeten Verfahren finden sich in [Zenk 1994, Kap. 4.1].

5.2 Internet (OSI-Schicht 3)

Als wichtigstes Teilprotokoll der TCP/IP-Familie arbeitet auf Schicht 3 das *Internet-Protokoll* (IP). Dieses Protokoll überträgt Daten zwischen den einzelnen am Internet angeschlossenen Rechnern in Form von Datenpaketen. IP ist verantwortlich für die Fragmentierung großer Datenpakete in Pakete mit einer an die Kapazität des zugrundeliegenden physikalischen Netzes angepaßten Größe. Das Internet-Protokoll arbeitet *verbindungslos*, d.h. der Austausch von Pakten zwischen zwei Rechnern wird nicht durch Kontrollinformationen abgesichert. IP garantiert daher nicht die ordnungsgemäße Zustellung eines Datenpaketes.

Die an der Kommunikation mittels IP beteiligten Systeme werden durch eindeutige Adressen bezeichnet. Diese 32 Bit großen *Internet-Adressen (IP-Adressen)* werden abhängig von ihrem Aufbau drei verschiedenen Adreßklassen zugeordnet. Um jeden Rechner eindeutig zu identifizieren, besteht jede Adresse aus zwei Teilen. Der erste Teil kennzeichnet das Netzwerk, in dem sich ein Rechner befindet, der zweite Teil den einzelnen Rechner im Netzwerk. Die verschiedenen Adressklassen bestimmen das Verhältnis beider Adreßteile zueinander (vgl. Abbildung 5.1). Die Klassen D und E werden zur Zeit noch nicht verwendet.

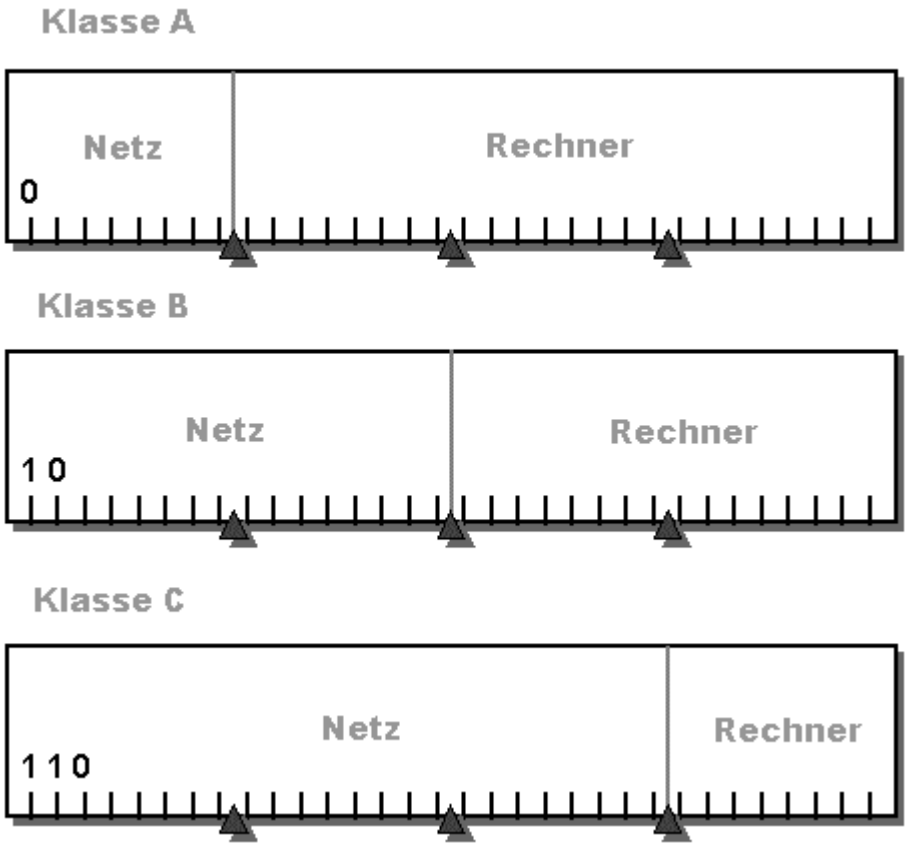


Abbildung 5.1: Adressklassen A bis C

Um den vorhandenen Adreßraum bestmöglich auszunutzen, kann eine Feinunterteilung zwischen Netz- und Rechneradreßteil vorgenommen werden. Mit Hilfe einer sogenannten *Subnetzmaske* wird das Teilungsverhältnis verändert. Die Subnetzmaske legt fest, welche Bits des Rechneradreßteils dem Bereich der Netzwerkkennung zugeordnet werden. Eine Subnetzmaske ist ebenso wie die IP-Adresse eine Folge von 4 Bytes oder 32 Bit, die mit der IP-Adresse verknüpft wird, und angibt, welche Bits der IP-Adresse zur Adressierung des Netzes und welche zur Adressierung eines Rechners in diesem Netz verwendet werden (vgl. Abbildung 5.2).

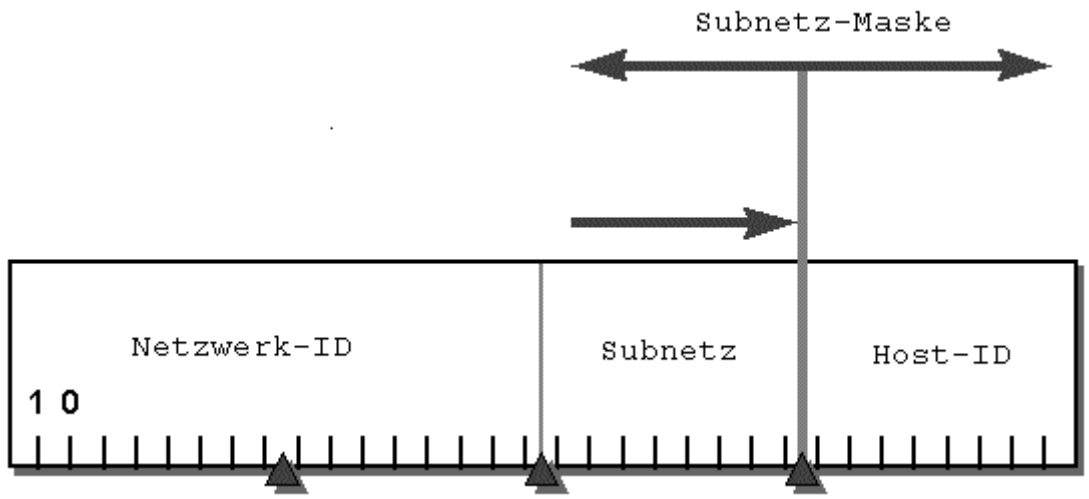


Abbildung 5.2: Aufbau der Subnetzmaske

Befinden sich zwei Rechner in einem physikalischen Netzwerk, so können sie direkt miteinander kommunizieren. Hier tritt ein weiteres Teilprotokoll von TCP/IP in Aktion, das *Address Resolution Protocol* (arp). Auf der Ebene von IP wird mit den oben erwähnten logischen Adressen gearbeitet, auf den Schichten 1 und 2 jedoch mit physikalischen Adressen (z.B. mit 48 Bit Ethernet Adressen). Die Umsetzung von logischen in physikalische Adressen wird durch arp realisiert.

Verwandt mit *arp* ist *rarp*, das *Reverse Address Resolution Protocol*, dessen Funktion eine Umkehrung von *arp* darstellt. Zusammen mit dem *Bootstrap Protocol* (bootp) wird *rarp* eingesetzt, um Geräten und Rechnern im TCP/IP-Netz eine IP-Adresse aufgrund ihrer eindeutigen Hardware-Adresse zuzuordnen. Bootp verwaltet ebenso wie *rarp* eine Tabelle mit Einträgen von Internetadressen und ihren zugehörigen Hardwareadressen. Erfolgt eine Anfrage mittels einer bestimmten Hardwareadresse, wird durch *bootp* oder *rarp* die zugehörige Internetadresse mittels der Tabelle zugeordnet. Das statisch arbeitende *bootp* wird zunehmend durch das nicht-statische *Dynamic Host Configuration Protocol* (dhcp) verdrängt, eine Erweiterung von *bootp*.

Befinden sich zwei Rechner nicht in demselben physikalischen Netz, können Datenpakete nicht direkt zugestellt werden, da die Zuordnung von IP- zu Hardwareadresse nicht durchgeführt werden kann. Die Zustellung der Pakete muß in diesem Fall über einen oder mehrere Vermittlungsrechner (*IP-Router*) erfolgen. Als IP-Router bezeichnet man einen Rechner, der IP-Datenpakete abhängig von ihrer Zieladresse in verschiedene Teilnetze weiterleitet (s. Abbildung 5.3).

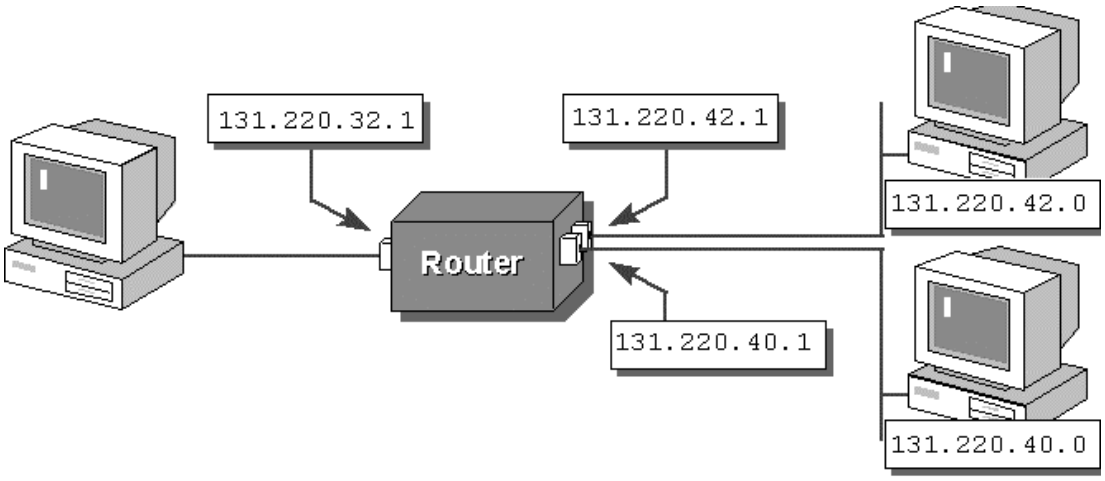


Abbildung 5.3: Aufbau eines IP-Routers

Der Routingprozess wird entweder statisch oder dynamisch durchgeführt. Im ersten Falle entscheidet der Router über die Wegwahl aufgrund voreingestellter Werte, im zweiten Falle tauschen verschiedene Router Informationen über erreichbare Rechner aus und generieren so eine sich verändernde Tabelle mit Informationen zur Wegwahl.

Statisches Routing wird aufgrund des hohen Verwaltungsaufwandes nur innerhalb kleinerer Netze angewandt und in Fällen, bei denen es nur einen Weg zum Ziel gibt. Einzelne Wege werden dabei manuell in eine Routing-Tabelle eingetragen.

Beim dynamischen Routing werden neue Wege automatisch durch verschiedene Routing-Protokolle hinzugefügt. In Bereich des dynamischen

Routings für kleine und mittlere TCP/IP-Netze kommen meist die Protokolle *RIP* (Routing Information Protocol) und *OSPF* (Open Shortest Path First) vor. Gruppen von Netzen mittlerer Größe werden zu sogenannten *Autonomen Systemen* (autonomous systems) zusammengefaßt und mit Hilfe des *Exterior Gateway Protocol* (EGP) verbunden. Ein solches System besteht aus mehreren Routern, die nach einem gemeinsamen Protokoll (RIP oder OSPF) arbeiten, und die mit der Außenwelt durch *Grenzrouter* (Autonomous System Boundery Router) verbunden sind. Die Grenzrouter tauschen ihrerseits Informationen mit den Grenzroutern anderer autonomer Systeme aus und sind meist direkt mit den zentralen Routern des Internet (dem *Backbone*) verbunden.

Neben den Routing-Protokollen arbeitet auch *das Internet Control Message Protocol* (ICMP) auf OSI-Schicht 3. ICMP überträgt Fehlermeldungen und Informationen, die zur Diagnose von Netzverbindungen eingesetzt werden können.

5.3 Transport-Protokolle (OSI Schicht 4)

Auf Schicht 4 arbeiten die beiden Transportprotokolle *TCP* (Transmission Control Protocol) und *UDP* (User Datagram Protocol). TCP ist ein *verbindungsorientiertes* Protokoll, d.h. es wird eine virtuelle End-End-Verbindung hergestellt, bevor Nutzdaten übertragen werden. Der Empfang jeder erhaltenen Information wird bestätigt. UDP hingegen überträgt Daten *verbindungslos*, d.h. Datenpakete werden nicht in der korrekten Reihenfolge zugestellt und es erfolgt keine Bestätigung des Empfangs. Da keine zusätzlichen Protokollinformationen übertragen werden müssen, arbeitet UDP deutlich schneller, allerdings auch unzuverlässiger.

Die Transportprotokolle leisten jedoch noch mehr. Wenn zwei Rechner miteinander kommunizieren, geschieht dies in der Regel über eine Verbindung zwischen zwei Prozessen oder Anwendungsprogrammen. Um Daten nicht nur einem bestimmten Rechner zuzustellen, sondern auch einer bestimmten Anwendung auf diesem Rechner, muß die Anwendung selbst adressierbar sein. Aus diesem Grunde erhält der Anwendungsprozeß eine 16 Bit große Portnummer zugewiesen, die ihn identifiziert. Eine Kombination aus IP-Adresse und Portnummer, die auch als *Socket* bezeichnet wird, identifiziert eine Anwendung innerhalb eines Netzes eindeutig.

5.4 Anwendungen (OSI Schicht 5-7)

Eine unüberschaubar große Zahl von Anwendungen setzt auf TCP/IP als Transportprotokoll auf. Fast alle Anwendungen arbeiten nach dem *Client/Server-Modell*. Dabei arbeitet auf einem Rechner ein Hintergrundprozeß (der Server), der Anweisungen eines Anwendungsprozesses (des Client) entgegennimmt und ausführt. Client und Server arbeiten typischerweise auf zwei verschiedenen, über TCP/IP verbundenen Rechnern, können jedoch auch beide auf einem Rechner ausgeführt werden.

Einige im Internet weit verbreitete TCP/IP-Dienste werden nun kurz vorgestellt. Da bei ihrer Entwicklung Sicherheitsaspekte nicht im Vordergrund standen, werden diese Dienste in Kapitel 7 im Hinblick auf ihre Sicherheit eingehender untersucht.

- *World Wide Web* (WWW): Hier handelt es sich um einen Netzdienst, dessen Nutzung innerhalb einer kurzen Zeitspanne enorm zugenommen hat. WWW stellt ein benutzerfreundliches Interface zur Abfrage von Informationen zur Verfügung, die von verschiedenen Rechnern im Internet

abgerufen werden können. Die Datenübertragung erfolgt mittels des *Hypertext Transfer Protocol* (http). Die zu übertragenden Informationen werden über eine einheitliche Konvention, den sogenannten *Uniform Resource Locator* (URL) bezeichnet.

- *Domain Name System* (DNS): Dieser Dienst sorgt für die Zuordnung zwischen der numerischen IP-Adresse und dem Rechnernamen. Damit wird eine Kommunikation unter Angabe des Rechnernamens möglich, ohne daß die IP-Adresse dem Anwender bekannt sein muß. Der DNS-Namensraum ist hierarchisch aufgebaut und gliedert sich in verschiedene Domänen. Die einzelnen Rechner in diesen Domänen werden eindeutig über den Namen des Rechners, die organisatorische Einheit und die Länderkürzel (außerhalb der USA) oder die organisatorischen Funktion (innerhalb der USA) identifiziert (vgl. [Kyas 1998] S.75).

- *File Transfer Protocol* (FTP): FTP dient zur interaktiven Übertragung von Dateien. Dabei werden gegebenenfalls Anpassungen an die jeweilige Form der Datenspeicherung unterschiedlicher Betriebssysteme vorgenommen. FTP konvertiert ein Set von Standardbefehlen zur Datenspeicherung in die für ein gegebenes Betriebssystem gültigen Funktionsaufrufe und sorgt daher für eine betriebssystemunabhängige Dateiübertragung. Eine im Funktionsumfang reduzierte Version von FTP, das *Trivial File Transfer Protocol* (TFTP) wird beim Start von Systemen ohne Festplatte eingesetzt. TFTP erfordert im Gegensatz zu FTP keine Benutzeridentifizierung und Authentifizierung.

- *Teletype Network* (TELNET): Der Dienst Telnet ermöglicht das Arbeiten auf einem entfernten Rechner im Netz. Telnet nimmt Benutzereingaben entgegen und leitet sie an den ausführenden Rechner weiter. Nach der Bearbeitung wird das Ergebnis wieder auf dem Terminal des Benutzers dargestellt. Die Konvertierung der unterschiedlichen Funktionen einzelner

Terminals wird von Telnet durch Übertragung und Auswertung von Steuerinformationen vorgenommen. Die Anmeldung eines Benutzers wird nicht von Telnet durchgeführt, sondern an das jeweilige Serverbetriebssystem weitergeleitet.

- *Berkeley R-Befehle* (RCMD): Diese Sammlung von Befehlen erweitert vorhandene UNIX-Kommandos im Hinblick auf den Einsatz in Netzwerken. So dient beispielsweise der Befehl rcp zur Übertragung von Dateien zwischen UNIX-Rechnern und erweitert damit den Grundbefehl cp (copy), der innerhalb lokaler Filesysteme eingesetzt wird.

- *E-Mail* (SMTP): Die Übertragung von Nachrichten zwischen Benutzern verschiedener Systeme erfolgt über das Simple Mail Transfer Protocol (SMTP). Dieses Protokoll ermöglicht den Austausch von Informationen zwischen verschiedenen E-Mail Anwendungen. Nachrichten werden nicht direkt übertragen, sondern bis zur Übermittlung an den Zielrechner auf einem Serversystem zwischengespeichert.

- *News* (NNTP): Die Net-News sind ein offenes Diskussionsforum im Internet. Das Network News Transport Protocol (NNTP) sorgt für die Übermittlung der einzelnen Beiträge sowohl zwischen zwei Servern als auch zwischen Server und Newsreader.

- *Network File System* (NFS): Das von der Firma SUN entwickelte NFS dient der verteilten Datenhaltung in TCP/IP-Netzen. Die auf einem NFS-Server abgelegten Dateien werden einem oder mehreren anderen Rechnern zur Verfügung gestellt. Dabei erfolgt eine Umsetzung der Benutzerrechte zwischen lokalem Rechner und Serversystem. Probleme bei der Verwaltung von Benutzerrechten bei Zugriff auf Netzressourcen werden durch NIS (s.u.) umgangen.

- *Network Information Service* (NIS): Dieser Dienst wurde ebenfalls von der Firma SUN entwickelt und bietet die Möglichkeit der zentralen Benutzerverwaltung. Der vormals *Yellow Pages* genannte Dienst ermöglicht den Zugriff auf festgelegte Dateien auf einem Server durch verschiedene Rechner. Während NFS ganze Verzeichnisse exportiert, stellt NIS nach einem anderen Mechanismus einzelne Dateien im Netz zur Verfügung.

- *X-Windows* (X11): Hier handelt es sich um eine vom MIT entwickelte netzwerkfähige graphische Oberfläche für UNIX-Systeme. Der Begriff X11 schließt auch das zugrundeliegende Protokoll zwischen dem X-Server und X-Client ein. Ein X-Server ist die Komponente, die für die Graphikausgabe zuständig ist, der X-Client die Komponente, auf der eine Anwendung ausgeführt wird (entgegen der sonst bei TCP/IP üblichen Terminologie). X11 ermöglicht die verteilte Programmausführung auch zwischen Rechnern mit unterschiedlicher Hard- und Softwareausstattung.

- *Simple Network Management Protocol* (SNMP): Dieses Protokoll wurde zur Fernverwaltung von TCP/IP basierten Netzkomponenten entwickelt. SNMP bietet die Möglichkeit, die Konfiguration eines Rechners oder eines SNMP-fähigen Gerätes (z.B. eines Routers) über das Netz auszulesen und zu verändern. Die Konfigurationsmöglichkeiten sind in der sog. *Management Information Base* (MIB) festgelegt. Auf jedem durch SNMP verwalteten System muß ein Hintergrundprozeß (snmpd) aktiv sein, der die Verbindung zum lokalen Betriebssystem herstellt. SNMP sieht abgestufte Benutzerrechte bei der Konfiguration vor.

Abschließend ergibt sich das in Abbildung 5.4 dargestellte Bild der TCP/IP-Protokollfamilie mit den wichtigsten Teilprotokollen und Diensten.

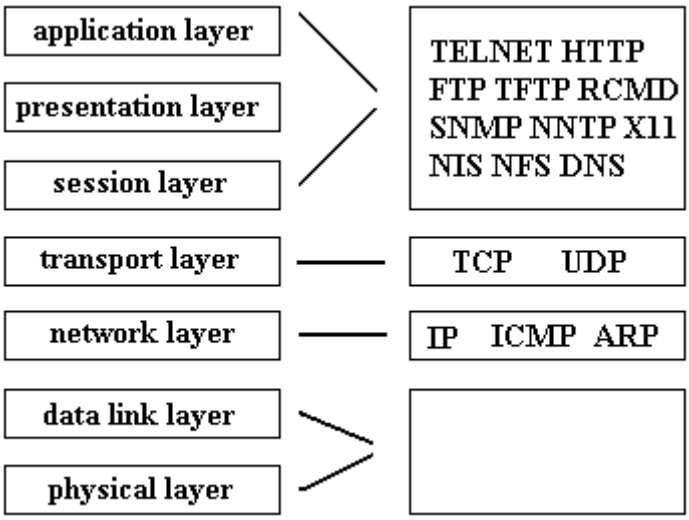


Abbildung 5.4: TCP/IP-Protokolle und Dienste

Da Sicherheitskriterien bei der Entwicklung der beschriebenen Dienste nicht im Vordergrund standen, ergeben sich zahlreiche Möglichkeiten für Angriffe, deren verschiedene Formen der nächste Abschnitt beschreibt. Daran schließt sich eine Untersuchung der Sicherheit der einzelner Protokolle an.

6 Formen von Angriffen auf die Internet-Kommunikation

Computer und Computernetze sind prinzipiell angreifbar. Dies gilt in besonderem Maße auch für das Internet und die im Internet eingesetzten Dienste. Angriffe auf die Internet-Kommunikation werden in der einschlägigen Literatur (vgl. [Fuhrberg 1998, Kap. 3.2], [Kyas 1998, Kap 8.1], [Raepple 1998, Kap. 1.13] und [Nusser 1998, Kap. 2.5]) nur fragmentarisch dargestellt. Dies resultiert aus der unüberschaubaren Zahl möglicher Angriffsmechanismen und -verfahren, die schwer zu klassifizieren sind.

Im Rahmen dieser Arbeit wird versuche ich, eine Einordnung zu finden, die allen möglichen Varianten Rechnung trägt und einzelne Sicherheitslücken und die daraus resultierenden Angriffe systematisch dem OSI-Modell zuordnet. Diese Systematik wird später auch ausgedehnt auf Mechanismen zum Schutz der Internet-Kommunikation, so daß ein vollständiges Bild der Angriffe auf die sichere Kommunikation und der möglichen Abwehrmaßnahmen entsteht.

Angriffe werden durch *Sicherheitslücken* in Hard- und Software möglich, die aus menschlichen Fehlern resultieren. Dazu zählen *Designfehler*, *Implementierungsfehler*, *Konfigurationsfehler* und *Fehler beim Betrieb* eines Systems. Diese Fehler haben Auswirkungen auf die im Internet eingesetzten Betriebssysteme, Dienste und Protokolle.

Es wird unterschieden zwischen *aktiven* und *passiven* Angriffen. Bei passiven Angriffen nimmt der Angreifer keine Modifikationen an den übertragenen Daten vor, er bricht lediglich die Vertraulichkeit der Kommunikation und wertet die erhaltenen Informationen aus. Diese Form des Angriffs ist häufig der Vorläufer zu einem aktiven Angriff, bei dem der Angreifer gezielt Daten manipuliert. Passive Angriffe sind, im Gegensatz zu aktiven Angriffen, oft nicht zu erkennen oder nachzuweisen.

Die verschiedenen Angriffsverfahren sind in den folgenden Abschnitten dargestellt.

6.1 Sniffing

Unter *Sniffing* wird das Abhören der im Netz übertragenen Informationen verstanden. Erforderlich zum Abhören der Internetkommunikation ist entweder der Zugriff auf das physikalische Netz des Senders bzw. Empfängers, d.h. ein Zugriff auf den OSI-Schichten 1 und 2, oder der Zugriff auf einen Vermittlungsrechner im Internet (IP-Router, Schicht 3).

An jeder Stelle zwischen dem sendenden und dem empfangenden Rechner im Internet können Informationen abgehört werden, da TCP/IP Daten prinzipiell unverschlüsselt überträgt, solange keine besonderen Maßnahmen getroffen werden. Da der Aufbau von TCP/IP und den zugehörigen Teilprotokollen bekannt ist, können die auf den Schichten 1 bis 4 abgehörten Daten den jeweiligen Anwendungen und Diensten zugeordnet werden.

Auf diese Art gewinnt der Angreifer Informationen über Rechner- und Benutzernamen, über Paßwörter und Netzkonfigurationen, die zu weiteren Angriffen genutzt werden können. Da es sich um einen passiven Angriff handelt, bestehen kaum Möglichkeiten, diesen Angriff nachzuweisen.

6.2 Scanning

Unter *Scanning* versteht man das systematische Absuchen eines Netzes nach bestimmten Diensten. Dabei wird versucht, ganze Subnetze dahingehend zu kartographieren, daß erkennbar wird, welche TCP/IP-Portadressen auf den einzelnen Rechnern in Verwendung sind. Diese Informationen erlauben Rückschlüsse auf die Art des eingesetzten Betriebssystems, seine Konfigurationen und über aktivierte Serverdienste.

In einer zweiten Phase versucht der Angreifer herauszufinden, welche Version eines bestimmten Dienstes aktiv ist. Wird eine Version identifiziert, von der Fehler bekannt sind und veröffentlicht wurden, können diese für einen späteren Angriff genutzt werden.

Scanning dient primär der Informationsgewinnung, wird jedoch wegen des aktiven Aufbaus von Verbindungen zu bestimmten TCP/IP-Ports oftmals bereits als Angriff auf ein System gewertet. Einfache Angriffe mit fortlaufenden Tests von Portadressen können vom angegriffenen System bemerkt werden, Angriffe mit zufälliger Auswahl von Portadressen und Pausen zwischen den einzelnen Tests sind schwer nachzuweisen.

6.3 Spoofing

Das Vortäuschen einer falscher Identität während einer Kommunikation im Internet wird mit dem Terminus *Spoofing* bezeichnet. Der Angreifer erzeugt dazu Datenpakete, die eine falsche Absenderangabe enthalten. Da viele Mechanismen in TCP/IP den Zugriff auf Rechner und Dienste davon

abhängig machen, von welcher Maschine ein Datenpaket stammt, kann der Angreifer sich so unberechtigterweise Zugriff verschaffen.

Spoofing funktioniert auf verschiedenen Ebenen der Übertragung im Netz. Adressen im physikalischen Netz (Schicht 2) können ebenso verändert werden wie Adreßangaben im Internet-Protokoll (Schicht 3) und auf der Anwendungsebene. Moderne Netzwerktools sind in der Lage, jedes beliebige Datenpaket zu erzeugen und im Netz zu versenden.

Spoofing ist ein aktiver Angriff, dessen Erkennung in komplexen Netzwerken einen hohen technischen Aufwand erfordert.

6.4 Denial of Service

Angriffe auf die Verfügbarkeit eines Rechners werden als *Denial of Service-Attack* bezeichnet. Ziel des Angriffes ist es, einen Rechner teilweise (d.h. auf Ebene einzelner Dienste) oder vollständig lahmzulegen. Angriffe dieses Typs nutzen Designfehler und Implementierungsfehler in Netzwerkprotokollen, Betriebsystemen und Diensten.

Das gezielte Ausschalten einzelner Dienste oder Sicherheitsmechanismen erfolgt oft als Vorbereitung eines weiterführenden Angriffs. Das Vortäuschen einer falschen Identität kann beispielsweise erfordern, daß ein Rechner gezielt ausgeschaltet wird, damit der Angreifer dessen Internetadresse verwenden kann.

Aktive Angriffe dieser Art werden meist mit Fehlfunktionen der Software verwechselt, da der Absturz eines Systems nicht mit einem Angriff in Verbindung gebracht wird. Beruht der Angriff darüber hinaus auf Designfehlern im TCP/IP-Protokoll, dann ist oft keine Schutz möglich.

6.5 Hijacking

Die Übernahme einer bestehenden TCP/IP-Verbindung wird als *Hijacking* bezeichnet. Der Angreifer nutzt die Tatsache, daß eine Identifizierung und Authentifizierung des Benutzers in der Regel zu Beginn einer TCP/IP-Verbindung erfolgt. Gelingt es, die Verbindung zu einem späteren Zeitpunkt zu übernehmen, arbeitet der Angreifer mit allen Rechten des ursprünglichen Benutzers.

Eine Absicherung der Datenübertragung durch zusätzliche Steuerinformationen erschwert die Übernahme von Verbindungen. Da jedoch der gesamte Aufbau des TCP/IP-Protokolls bekannt und veröffentlicht sein muß, um eine Implentierung auf verschiedenen Plattformen zu ermöglichen, stehen diese Informationen auch dem Angreifer zur Verfügung.

6.6 Routing Attack

Angriffe auf Vermittlungsrechner oder auf die Wegwahl-Funktionen von TCP/IP werden als *Routing Attack* bezeichnet. Der Angreifer kann in diesem Fall mehrere Ziele verfolgen. Im einfachsten Fall wird versucht, Informationen über den Aufbau des Zielnetzes zu erhalten, d.h. Informationen über die Positionierung einzelner Rechner, insbesondere solcher, die sicherheitsrelevante Funktionen ausführen.

Ein Angriff dieser Art kann jedoch auch aktiv das TCP/IP-Protokoll beeinflussen. Werden die Funktionen der Wegwahl im Internet manipuliert, können Datenpakete mit sensiblem Inhalt auf den eigenen Rechner umgeleitet werden.

Da nicht vorhersehbar ist, welchen Weg ein Paket durch das Netz nehmen wird und welche Rechner es passieren wird, sind Angriffe auf die Wegwahl schwer zu erkennen und zu bekämpfen.

6.7 Replay Attack

Durch das erneute Einspielen bereits im Netz übertragener Informationen, der sog. *Replay Attack*, versucht ein Angreifer, aufgezeichnete Identifizierungs- und Authentifizierungs-Informationen ein- oder mehrmals wiederzuverwerten. Werden Zugriffsmechanismen nicht über Zeitstempel oder ähnliche Verfahren abgesichert, können solche Aufzeichnungen nicht von aktuellen Informationen unterschieden werden.

Angriffe nach diesem Muster laufen in der Regel auf höheren Protokollschichten (Schicht 3 aufwärts) ab, sind jedoch nicht in jedem Fall daran gebunden. Die in TCP/IP verwendeten Sequenznummern sollen u.a. vor diesen Angriffen schützen, sind jedoch unter bestimmten Umständen vorhersagbar und damit für den Angreifer nutzbar.

Es handelt sich um einen aktiven Angriff, der mit geeigneten Mechanismen erkannt werden kann. Allein die Verwendung von Zeitstempeln ist jedoch nicht ausreichend, da auch diese manipuliert werden können. Digitale Signaturen im Zusammenhang mit Zeitstempeln können eine Replay Attack wirkungsvoll abwehren.

6.8 Social Engineering

Unter *Social Engineering* wird kein Angriff mit technischen Mitteln verstanden, dennoch ist diese Methode sehr effektiv. Ein Angreifer versucht Informationen über Rechnerkonfigurationen, Paßwörter und Schutzmechanismen eines Systems zu erhalten. Dazu gibt er sich z.B. als Angestellter einer Servicefirma oder als neuer Systemadministrator aus und fordert Mitarbeiter über E-Mail oder Telefon auf, bestimmte Informationen preiszugeben.

Unter dem Terminus Social Engineering kann allgemeiner auch jede Form der Informationsgewinnung zur Vorbereitung oder Durchführung eines Angriffes verstanden werden, solange die eingesetzten Mittel nicht technischer Natur sind. Dieser Angriff fällt in den Bereich der personellen Sicherheit und kann mit technischem Aufwand nicht verhindert werden.

Alle übrigen Angriffsmöglichkeiten werden im folgenden Abschnitt anhand des OSI-Modells (vgl. Kapitel 4.1) systematisch dargestellt.

7 Angriffe auf Internet-Protokolle und Dienste

7.1 ARP

Das *Adress Resolution Protocol* (arp) sorgt für die Vermittlung zwischen logischer IP-Adresse und physikalischer Hardware-Adresse. Ist einmal eine Zuordnung erfolgt, speichert *arp* das Ergebnis in einem Cache, um bei einer erneuten Anfrage die Antwortzeiten zu optimieren. Ein Angriff beruht auf der Tatsache, daß Systeme im lokalen Netz auch dann eine neue Zuordnung speichern, wenn sie selbst gar keine Anfrage gestellt haben. Der Angreifer sendet eine künstlich erzeugte Antwort an ein System, die diese in den Cache einträgt. Verwendet der Angreifer dabei seine eigene Hardware-Adresse, so kann er eine falsche Zuordnung zwischen einer fremden IP-Adresse und seiner eigenen Hardware-Adresse im Cache speichern. Das angegriffene System sendet dann alle Pakete mit der fremden IP-Adresse an das System des Angreifers. Dieser Angriff fällt in den Bereich des Spoofing.

Ein weiterer Angriff, der in den Bereich Denial-of-Service fällt, besteht im Erzeugen von Anfragen für eine nicht existierende IP-Adresse. Diese Pakete werden von IP-Routern in angeschlossene Subnetze weitergeleitet und lösen einen sog. Broadcast-Storm aus, d.h. eine Anfrage an alle Systeme nach der nicht existenten Adresse. Wird nach einiger Zeit ein künstliches Antwortpaket erzeugt, wird dieses wiederum per Broadcast verteilt. Aufgrund der dadurch erzeugten enorm hohen Netzlast sind einzelne Rechner und Subnetze kurzfristig nicht mehr ansprechbar. Dieses Resultat kann für weitere Angriffe genutzt werden (s.u.).

7.2 IP

Auf Ebene des IP-Protokolls sind ebenfalls verschiedene Angriffe möglich. Viele Mechanismen zur Identifizierung eines Kommunikationspartners arbeiten über IP-Adressen, d.h. die Identität eines Rechners wird nur anhand der IP-Adresse überprüft. Stellt ein Angreifer nun fest, daß ein Vertrauensverhältnis zwischen zwei Systemen besteht, wird er versuchen, die Identität eines der beiden Systeme anzunehmen um das Vertrauensverhältnis zu seinen Gunsten auszunutzen. Ein mögliches Vertrauensverhältnis ist z.B. bei Einsatz der Berkeley R-Befehle gegeben. Hier führt ein Serversystem eine Liste von zugriffsberechtigten Clients, die ausschließlich anhand ihrer IP-Adresse identifiziert werden. Gelingt es dem Angreifer, die IP-Adresse eines Client vorzutäuschen, hat er vollen Zugriff auf die Ressourcen des Servers.

Der Angriff beginnt in der Regel mit dem Ausschalten des Systems, dessen Identität der Angreifer annehmen will, da zwei Systeme mit derselben IP-Adresse zu Problemen im Netz führen würden. Die einfachste und sicherste Möglichkeit, ein System lahmzulegen, besteht in einem Denial-of-Service-Angriff. Hat der Angreifer Zugriff auf das physikalische Netz der anzugreifenden Maschine, kann ein Angriff auch unmittelbar auf das Netz erfolgen, z.B. durch Trennung der Netzverbindung.

Im nächsten Schritt versendet der Angreifer Datenpakete mit der Adresse der zuvor ausgeschalteten Maschine. Schwierigkeiten kann hier die von TCP verwendete Sequenznummer dem Angreifer bereiten. Unter bestimmten Bedingungen ist diese jedoch vorhersagbar. Da UDP ohne Sequenznummern arbeitet, ist hier das Verfahren einfacher zu realisieren.

Ein weiterer Angriff setzt bei der Fragmentierung von IP-Paketen an. Überschreitet ein IP-Datenpaket eine festgelegte Größe, so wird es in mehrere kleine Pakete zerlegt. Der Angreifer erzeugt Pakete von minimaler Größe, die nicht alle Informationen für höhere Schichten (im Regelfall für TCP) enthalten können. Firewalls und Router filtern nicht alle diese Datenpakete aus, sondern nur das erste Paket. Weitere Pakete dürfen aufgrund der aufgespaltenen TCP-Informationen passieren.

Eine Variante dieser Methode nutzt die Möglichkeit, den Inhalt vorausgegangener Pakete durch nachfolgende zu überschreiben. So kann eine vollständige Verbindung zum Zielsystem aufgebaut werden. Mit diesen beiden Angriffen können Mechanismen zum Schutz lokaler Netze gegen Angriffe aus dem Internet umgangen werden.

7.3 ICMP

Das Protokoll ICMP nimmt Steuerungsaufgaben in TCP/IP-Verbindungen wahr und ist besonders für Denial-of-Service-Angriffe geeignet. Bei dieser Art des Angriffs werden ICMP-Pakete generiert, die einem oder mehreren Systemen im Netz Steuerinformationen zusenden. Diese Steuerinformationen enthalten z.B. Meldungen darüber, daß ein bestimmtes System nicht erreichbar ist, daß eine weitere Fragmentierung von Datenpaketen notwendig ist oder die Übertragungsrate reduziert werden soll.

Eine Veränderung der Erreichbarkeitsinformationen (*destination unreachable*) führt dazu, daß die Kommunikation zu einer bestimmten Maschine unterbrochen wird, da das sendende System irrtümlich annimmt, sein Ziel sei nicht zu erreichen.

Falsche Fragmentierungsaufforderungen führen aufgrund der deutlich größeren Zahl von Paketen im Netz zu einer Erhöhung der Netzlast, die auch zum Ausfall eines ganzen Netzes führen kann.

ICMP-Nachrichten, die eine Reduzierung der Übertragungsrate zwischen zwei Systemen veranlassen, können ein System lahmlegen, wenn sie wiederholt eingesetzt werden und das angegriffene System seine Übertragungsrate soweit reduziert, daß es am normalen Netzverkehr nicht mehr teilnehmen kann.

Ein indirekt über ICMP ausgeführter Angriff nutzt das unter allen TCP/IP-Implementierungen verfügbare Utility *ping,* um einem System übergroße Testpakete in einzelnen Fragmenten zuzusenden. Durch einen Fehler in der Verarbeitung dieser Pakete wird das angegriffene System zum Absturz gebracht. Diese als *ping-of-death* bekannte Methode wurde lange Zeit gegen Systeme unter Windows NT eingesetzt.

Die durch ICMP übertragenen Steuerungsinformationen, die im Zusammenhang mit dem Prozeß der Wegwahl im Internet stehen, werden im Abschnitt 7.6 erläutert.

7.4 TCP

Angriffe über TCP nutzen die Tatsache, daß Identifizierung und Authentifizierung des Kommunikationspartners (z.B. durch Abfrage eines Paßworts) in der Regel zu Beginn einer Verbindung erfolgt. Der Angreifer versucht, die Verbindung zu einem späteren Zeitpunkt zu übernehmen. Gelingt dies, kann er mit den Rechten des regulären Benutzers auf einem Server arbeiten. Diese in den Bereich des Hijacking fallende Technik beginnt mit der Suche des Angreifers nach einer bestehenden TCP-Verbindung.

Wird eine solche Verbindung gefunden, protokolliert der Angreifer alle zwischen beiden Maschinen verwendeten Sequenznummern und versucht, die weitere Abfolge dieser Steuerinformationen vorherzusagen. Hat er damit Erfolg, generiert er Datenpakete, die scheinbar von einem der beiden Kommunikationspartner kommen und unterläuft die Sicherungsmechanismen auf Ebene einzelner TCP/IP-Dienste.

Eine weiterer Angriff über TCP zielt auf die Verfügbarkeit eines anderen Rechners ab. Da TCP ein verbindungsorientiertes Protokoll ist, werden zu Beginn jeder Verbindung Steuerinformationen ausgetauscht. Ein Angreifer nutzt diesen Austausch von Informationen und stellt in rascher Folge Verbindungsaufforderungen an einen Server. Der Server reagiert auf jedes dieser Datenpakete mit einer Bestätigung und wartet auf weitere Pakete. Durch dieses Verhalten wird der Server während der Wartezeit für andere ankommende Verbindungen lahmgelegt. Diese Technik wird als *SYN-Flooding* bezeichnet

7.5 UDP

Das User Datagram Protocol (UDP) arbeitet im Gegensatz zu TCP verbindungslos und ist daher noch einfacher anzugreifen. Da UDP weder Bestätigung noch Steuerinformationen vom Zielsystem erwartet, können UDP-Verbindungen leichter als TCP-Verbindungen übernommen werden.

Dienste, die dieses Teilprotokoll nutzen, erfordern jedoch in der Regel keine Benutzeridentifikation, so daß eine Überwachung von UDP dem Angreifer keine diesbezüglichen Informationen liefert.

7.6 Routing

Der Angriff auf Router kann mehrere Ziele haben. Zunächst gibt ein nicht besonders abgesicherter Router Informationen über die interne Struktur des Netzes preis. Aus diesen Informationen erstellt der Angreifer eine detaillierte Übersicht des anzugreifenden Netzes. Routingprotokolle lokaler Netze, insbesondere Version 1 des Routing Information Protocol (RIP), versenden Informationen über lokal erreichbare Rechner und deren Adressen an alle anfragenden Systeme, solange dies nicht explizit unterbunden wird.

Der Angreifer startet auf seinem Rechner ebenfalls RIP, gibt seine Maschine damit als Router aus und empfängt diese Informationen. Version 2 des Routing Information Protocol erlaubt die Verwendung von Paßwörtern zur Authentifizierung eines fremden Systems und ist daher bei richtiger Konfiguration als sicherer anzusehen.

Im Gegensatz zum meist passiven RIP-Angriff sind auch aktive Angriffe möglich, die eine Veränderung der Routing-Informationen im anzugreifenden Netz zum Ziel haben. Bei dieser Art des Angriffs werden entweder einem Router gefälschte RIP-Pakete zugesendet, oder IP-Pakete mit zusätzlichen Routing-Informationen versehen. Die IP-Zusatzinformationen *Strict Source Routing* und *Loose Source Routing* erlauben es, den Weg eines IP-Paketes vorzugeben. Im ersten Fall werden alle Vermittlungsrechner explizit angegeben und Abweichungen von der vorgegebenen Route nicht akzeptiert, im zweiten Fall ist die Vermitllung über weitere Router zulässig. Über Loose Source Routing bietet sich damit eine einfache Möglichkeit, IP-Pakete durch Zusendung von Datenpaketen umzuleiten, die einen Weg über das eigene System festlegen.

Auch das Protokoll ICMP kann zur Beeinflussung der Wegwahl verwendet werden. Durch die gesetzte Redirect-Option teilt der Angreifer einem Router

mit, daß eine neue, bessere Route zu einem Zielsystem existiert. Übernimmt der Router diese Information, werden alle Datenpakete entsprechend den Vorgaben des Angreifers umgeleitet.

7.7 E-Mail

Auf jedem System, das E-Mail im SMTP-Format empfangen oder weiterleiten kann, läuft ein Hintergrundprozess (Daemon), der die eingehenden Nachrichten in Empfang nimmt und weiterverarbeitet. Der Programmcode des Hintergrundprozesses ist ebenso komplex und fehleranfällig wie die Konfiguration der Verarbeitungsregeln. Darüber hinaus läuft der Prozeß oftmals mit Superuser-Rechten, um allen Benutzern E-Mail zustellen zu können. Diese Tatsachen können für einen Angriff verwendet werden.

Der Angriff erfolgt z.B. über einen Aufruf der Debug-Option des Hintergrundprozesses. Ist eine ältere Version der Software installiert, kann der Angreifer auf diesem Wege Befehle auf dem Zielsystem ausführen oder eigene Dateien einschleusen.

Ein weiteres Problem tritt bei der Übermittlung von E-Mails zwischen Benutzerrechner und Mailserver auf. Arbeitet der Mailserver mit dem *Post Office Protocol* (POP), dann wird vor der Übertragung der Mail zum Rechner des Benutzers dessen Paßwort abgefragt. Dieses Paßwort wird unverschlüsselt übertragen und kann daher einfach mitgelesen werden. Verwendet der Benutzer dieses Paßwort auch in anderen Zusammenhängen (z.B. als Login-Paßwort), kann der Angreifer weitere Rechte erlangen.

Ein Denial-of-Service-Angriff erfolgt über das sog. Mail-Bombing. In seiner einfachsten Ausführung erfogt dieser Angriff mittels einer großen Zahl

künstlich erzeugter E-Mails, die gleichzeitig an eine Maschine verschickt werden und diese für einige Zeit lahmlegen. Eine Variante des Angriffs nutzt Konfigurationsfehler und erzeugt über Rückkopplungen sogenannte Mail-Loops, die einen Mail-Server ebenfalls innerhalb kürzester Zeit überlasten.

Mit dem Aufkommen von Protokollen mit erweiterten Funktionen, insbesondere der *Multipurpose Internet Mail Extensions* (MIME), kam eine neue Generation von Angriffsmethoden hinzu. Der MIME-Standard bestimmt u.a., welche Anwendung auf einem Rechner - abhängig vom Inhalt der E-Mail - gestartet werden soll. Ist die Konfiguration nicht entsprechend abgesichert, kann der Angreifer z.B. TFTP (s.u.) als Anwendung starten und daraufhin über diese Verbindung auf das Dateisystem eines fremden Rechners zugreifen.

7.8 Telnet

Telnet ermöglicht das Arbeiten auf einem entfernten Rechner im Netz und muß zu diesem Zweck die Tastatureingaben eines Benutzers über eine TCP/IP-Verbindung zu einem Server übertragen. Die gesamte Übertragung erfolgt im Klartext. Davon sind auch Benutzername und Paßwort zur Anmeldung am Server nicht ausgenommen. Durch Sniffing erlangt ein Angreifer daher sofort ein aktuelles Paßwort.

Auch aus anderen Gründen ist Telnet ein Sicherheitsrisiko. In der Vergangenheit sind immer wieder Sicherheitslücken im Zusammenhang mit dem zugehörigen Hintergrundprozess (dem Telnet-Daemon) bekannt geworden, bei denen der Angreifer z.B. durch einen Überlauf eines Zwischenspeichers eine Shell erlangen konnte.

7.9 FTP

Ebenso wie Telnet überträgt auch FTP alle Informationen (einschließlich Paßwörter) im Klartext. Der Angreifer gelangt darüber hinaus ohne Schwierigkeiten an den Inhalt aller übertragenen Dateien.

Die Konfiguration des zu FTP gehörenden Hintergrundprozesses ist komplex. Insbesondere wenn der Server für den anonymen Zugriff konfiguriert ist, kann es zu Fehlern kommen, die einen Angriff ermöglichen. Einige FTP-Server erwarten beispielsweise eine Kopie der systemweiten Paßwortdatei in einem allgemein zugreifbaren Unterverzeichnis. Achtet der Systemverwalter nicht darauf, die realen Paßwörter durch andere Zeichenfolgen zu ersetzen, gelangt der Angreifer an verschlüsselte Paßwörter, die durch geeignete Programme entschlüsselt werden können.

Der Serverprozeß wird in der Regel mit Superuser-Rechten gestartet, um über privilegierte TCP-Ports kommunizieren zu können. Daraus ergeben sich weitere Sicherheitsrisiken, wenn Fehler im Programmcode vorliegen.

Sowohl FTP als auch Telnet sind noch für einen weiteren Angriff empfänglich. Gelingt es einem Angreifer, auf dem lokalen System die Befehle ftp und telnet durch modifizierte Versionen zu ersetzen, ist es möglich, alle eingegebenen Benutzer-Paßwörter zu speichern, die von Anwendern eingegeben werden.

7.10 TFTP

Mittels TFTP erfolgt eine Dateiübertragung ohne vorherige Abfrage eines Benutzernamens oder eines Paßworts. TFTP wird daher in Situationen genutzt, in denen eine solche Übertragung notwendig ist, z.B. beim Start

eines Rechners ohne eigene Systemdatenträger, da hier noch keine Benutzereingaben möglich sind. Gelingt es einem Angreifer, zum richtigen Zeitpunkt während des Startvorganges die TFTP-Datenpakete zu manipulieren, kann er Rechte auf dem neu gestarteten System erlangen.

Eine weitere Angriffsmöglichkeit ergibt sich, wenn der zugehörige Hintergrundprozeß ohne Eingabe eines Root-Verzeichnisses (z.B. /tftpboot) gestartet wird. In diesem Fall können alle Dateien eines lokalen Systems vom Angreifer kopiert werden. Bei neueren Versionen von TFTP wird der Zugriff automatisch auf ein bestimmtes Verzeichnis beschränkt, wenn der TFTP-Daemon nicht anders konfiguriert ist.

7.11 Berkeley R-Befehle

Um UNIX-Kommandos netzwerkfähig zu machen, müssen die Rechte von Benutzern auf fremden Maschinen festgelegt werden, z.B. die Leserechte einer Datei, die später mittels *remote copy* (rcp) von einem Rechner transferiert werden soll.

Diese Rechte auf einem fremden Rechner verwalten die R-Befehle abhängig vom Ausgangsrechner des Benutzers entweder über eine zentrale Datei (*hosts.equiv*), die festlegt, welchen anderen Rechner vertraut wird, oder über eine Datei im Home-Directory des Benutzers (*.rhosts*), die angibt, von welchen Accounts auf fremden Rechnern ein Zugriff erfolgen darf.

Gelingt es einem Angreifer, eine der beiden Dateien zu modifizieren, so erlangt er unberechtigt Zugriff auf einen fremden Rechner. Gerade durch die dezentrale Ablage der Datei *.rhost* im Home-Directory der Benutzer entstehen Lücken in der Sicherheit, da diese Dateien nicht vom Super-User

angelegt werden, sondern ihre Verwaltung in der Verantwortung regulärer User liegt.

7.12 WWW

Mit der starken Verbreitung und Nutzung des WWW-Dienstes wurden einige Angriffe entwickelt, die speziell auf diesen Dienst abzielen. Begünstigt werden diese Angriffe einerseits durch die Tatsache, daß Verbindungen über das zugehörige *Hypertext Transfer Protokoll* (HTTP) unverschlüsselt erfolgen, andererseits durch die Ausführung von Programmen auf dem WWW-Server, die zur Erstellung dynamischer Webseiten verwendet werden.

Bei solchen dynamischen Webseiten wird nicht eine vorgefertigte, in HTML (Hypertext Markup Language) erstellte Seite zum Benutzer übertragen, sondern es werden Benutzereingaben vom Server entgegengenommen, verarbeitet und in eine automatisch generierte Webseite übernommen.

Enthält nun der Programmcode der auf dem Server ausgeführten Anwendungen Fehler oder wird die Konfiguration nicht sorgfältig ausgeführt, kann ein Angriff auf den entsprechenden Server erfolgen. Häufig werden auf dem Server Interpretersprachen, z.B. TCL oder Perl verwendet, bei denen sowohl der Programmcode des Interpreters als auch das auszuführende Script fehlerhaft sein können. Auch die Vermittlungsschicht zwischen Serverprozeß und Interpretersprache, das *Common Gateway Interface* (CGI), war in der Vergangenheit mehrfach Ziel von Angriffen.

Mit dem Einzug der Programmiersprache Java sind neben WWW-Servern auch zunehmend Clients von Angriffen betroffen. Die plattformunabhängige Sprache Java erlaubt den Download von Programmen, die auf dem Rechner des Benutzers ausgeführt werden. Sicherheitslücken im Java-Interpreter

führen nach Download und Start von ausführbarem Java-Programmcode (sog. Java-Applets) zum Zugriff auf sensible Daten durch den Angreifer. Die Sicherheitsmechanismen von Java werden in Kapitel 11.5.1 dargestellt.

Eine neue Bedrohung ergibt sich auch durch die Erstellung von Benutzerprofilen. In diesen Profilen wird gespeichert, welche Webseiten von einem Benutzer wie oft aufgerufen wurden. Der Abruf einer bestimmten Seite wird in einer Datei (einem sog. *Cookie*) auf dem Rechner des Benutzers gespeichert. Diese Datei kann von jedem anderen Webserver gelesen und ausgewertet werden.

Die so gewonnenen Informationen werden z.B. eingesetzt, um dem Benutzer auf ihn abgestimmte Werbung zukommen zu lassen. Es sind natürlich auch andere, weniger harmlose Verwendungen solcher Informationen über eine bestimmte Person denkbar. Die von einem Benutzer aufgerufenen Webseiten verbleiben darüber hinaus mit vollständigem Inhalt für einige Zeit im WWW-Zwischenspeicher (WWW-Cache) seines Rechners.

7.13 DNS

Das Domain Name System (DNS) ist für die Zuordnung zwischen numerischer IP-Adresse und Rechnernamen zuständig. Gelingt einem Angreifer die Manipulation eines DNS-Servers, liefert dieser bei Anfrage nach der IP-Adresse eines Systems die vom Angreifer vorgegebene Adresse zurück. Wird nun von anderen Rechnern eine Anfrage gestartet, z.B. vermittelt durch die Verwendung eines WWW-Browsers oder der Telnet- und FTP-Befehle, werden diese Dienste nicht mit dem eigentlich vorgesehenen Serversystem verbunden, sondern mit dem Rechner des Angreifers. Dieser gewinnt damit die Möglichkeit, dem Anwender falsche Informationen (z.B.

manipulierte Webseiten) zuzuspielen oder Paßwörter (bei Telnet oder FTP) mitzulesen.

Ein Angriff auf DNS wird dadurch erleichtert, daß lokale Systeme bei DNS-Zugriffen die Antwort des Servers in einem Cachespeicher ablegen, um die Antwortzeiten bei weiteren Anfragen zu verbessern. Dieser Cache kann gezielt durch Zusendung falscher Namen- / Adreßpaare manipuliert werden. Neuere DNS-Server prüfen eingehende Pakete entsprechend und weisen den einzelnen Anfragen eine eindeutige ID zu, die ebenfalls bei Erhalt der Antwort geprüft wird. Ebenso wie die TCP-Sequenznummer ist diese ID jedoch mit etwas Aufwand vorherzusehen.

Weiterhin kann ein DNS-Server auch detaillierte Informationen über den Aufbau eines anzugreifenden Netzes liefern, da er alle dort existierenden Rechnernamen und Adressen in seiner Datenbank gespeichert hat, zusammen mit Zusatzinformationen über die Funktion einzelner Rechner, z.B. der Angabe, welches System als Router arbeitet.

7.14 NFS

Das Network File System (NFS) stellt ausgewählte Verzeichnisse eines Servers zur Nutzung durch verschiedene Clients bereit. Problematisch ist bei NFS zunächst, daß zur Identifizierung und Authentifizierung eines Client nur dessen IP-Adresse verwendet wird. Durch IP-Spoofing ist es daher einem Angreifer möglich, unberechtigt Zugriff auf die Serverdaten zu erhalten.

Weiterhin findet die Kommunikation zwischen Client und Server unverschlüsselt statt. Ein Client erhält zu Beginn der Verbindung eine Verwaltungsinformation für den Zugriff auf die Dateien des Servers, die über die gesamte Dauer der Verbindung nicht geändert wird. Gelingt es einem

Angreifer, Datenpakete mitzulesen, kann er diese Information einsetzen, um vom Server Dateien anzufordern.

NFS ist modular aufgebaut und besteht auf UNIX-Systemen in der Regel aus den Hintergrund-Prozessen *mountd* und *nfsd*. Prüft bei diesem Verfahren nur der Prozeß *mountd* die Berechtigung des Client, kann durch direkte Verbindung zum *nfsd* die Prüfung umgangen werden.

Wird NFS nicht entsprechend konfiguriert, sind freigegebene Verzeichnisse für alle anfragenden Maschinen lesbar. Gelingt es einem Angreifer, sensible Informationen auszulesen (z.B. Dateien mit Paßworteinträgen), kann mit diesen neu erworbenen Informationen ein weiterer Angriff gestartet werden.

7.15 NIS

Über NIS werden zentrale Dateien mit Verwaltungsinformationen für alle Rechner im Netz zur Verfügung gestellt. Diese Verwaltungsinformationen umfassen auch eine netzweite Paßwortdatei, über die sichergestellt werden kann, daß alle Benutzer im UNIX-Netz mit derselben Benutzeridentifikation (UID) arbeiten. Dienste, die Benutzerrechte über die UID verwalten (z.B. NFS) sind auf diesen Mechanismus angewiesen.

Problematisch ist, daß auch hier die Übertragungen im Klartext erfolgen. Ein Angreifer kann so Paßwortinformationen mitlesen und auswerten. Es ist auch möglich, anfragenden Rechnern gezielt falsche Informationen zuzuspielen. Akzeptiert ein Rechner beispielsweise eine gefälschte Paßwortdatei, kann sich der Angreifer mit allen Rechten auf dieser Maschine anmelden (vgl. [Kyas 1998] S. 143).

Neuere Versionen dieses Dienstes, z.B. die von der Firma SUN eingeführte Version NIS+, bieten bessere Möglichkeiten zum Schutz sensibler Informationen.

7.16 NetBIOS

Die Netzdienste von Microsoft-Betriebssystemen kommunizieren miteinander über *NetBIOS*, einem Protokoll auf Schicht 5, mit dessen Hilfe z.B. auf freigegebene Verzeichnisse zugegriffen werden kann. NetBIOS war zunächst für lokale Netze ausgelegt und arbeitete über das nicht routingfähige Transportprotokoll NetBEUI. Mit der Anpassung von NetBIOS an TCP/IP als Transportprotokoll konnten spezifische Microsoft-Dienste auch im Inter- und Intranet genutzt werden.

Das über NetBIOS ablaufende *Server Message Block Protocol* (SMB) ist sehr fehleranfällig. Werden die zugehörigen TCP/IP-Ports 135 bis 139 nicht gegen Zugriff geschützt, kann der Angreifer Schwächen der Sicherheitsarchitektur von Windows-Systemen ausnutzen. Die häufigste Angriffsart ist hier Denial-of-Service. Einem Rechner unter Windows NT werden z.B. Aufforderungen zum Zugriff auf ein freigegebenes Verzeichnis zugesandt, die einen überlangen Datei- oder Rechnernamen enthalten. Das NT-System reagiert sofort mit einem vollständigen Absturz.

Gezielte Angriffe auf die Interprozeßkommunikation (IPC) eines Windows-Rechners erlauben darüber hinaus den Zugriff auf Konfigurations- und Benutzerdaten. Eine detaillierte Darstellung findet sich in Kapitel 8.4.

7.17 X11

Das X-Window-System ist nicht nur in der UNIX-Welt weit verbreitet. Mittels des zugehörigen Protokolls werden u.a. Tastatureingaben eines Benutzer über ein TCP/IP-Netz zur verarbeitenden Maschine transferiert. Diese Tastatureingaben enthalten je nach Konfiguration auch Benutzerpaßwörter, die vom Angreifer mitgelesen werden können.

Gleichzeitig ist es möglich, bei einem ungeschützten System den Inhalt einzelner Eingabefenster der graphischen Oberfläche zu protokollieren. Darüber hinaus ist auch X-Windows, wie jedes graphische Benutzerinterface, anfällig für Angriffe, die Eingabemasken simulieren, um Benutzereingaben aufzuzeichnen (im Sinne eines trojanischen Pferdes, s.u.).

7.18 Weitere Angriffe

Neben den genannten Angriffsmethoden gibt es noch viele weitere, die weniger verbreitet sind und deshalb hier nicht dargestellt werden. Oftmals werden diese Methoden zur Vorbereitung anderer Angriffe eingesetzt oder begleiten diese.

Ein typischer Vertreter ist der Angriff auf das *Network Time Protocol* (NTP). Über NTP wird die Systemzeit mehrerer Rechner im Netz aufeinander abgestimmt. Viele Sicherheitsmechanismen sind auf eine exakte Zeiteinstellung angewiesen, um z.B. zu prüfen, ob Zertifikate und Benutzerpassworte abgelaufen sind oder um Replay-Angriffe (s.o.) wirksam zu unterbinden. Ein Angreifer, dem es gelingt, seine Maschine im Netz als Time-Server bekanntzumachen, hat die Möglichkeit, die

Sicherheitsmechanismen zu beeinflussen, um einen anderen Angriff durchführen zu können.

Die hier vorgestellten Angriffe auf die Internet-Kommunikation stehen immer in Zusammenhang mit Angriffen auf einzelne Rechner, die Knotenpunkte innerhalb der Kommunikationsverbindung darstellen. Diese Problematik wird im nächsten Abschnitt eingehender erläutert.

8 Angriffe auf Betriebssysteme

Angriffe auf Netzwerke sind immer Angriffe auf die darin befindlichen Rechner. So sind neben Netzwerkdiensten und Protokollen auch Betriebssysteme Ziel von Angriffen aus dem Internet. Betriebssysteme sind diesen Angriffen einerseits auf einzelne Systemen ausgesetzt, d.h. auf den Endpunkten einer Kommunikationsverbindung, andererseits auch auf Routern und Firewalls im Netz.

An dieser Stelle greifen Netzwerk- und Rechnersicherheit ineinander. Nur mit gesicherten Betriebssystemen auf den am Internet angeschlossenen Rechnern kann ein sicherer Netzbetrieb gewährleistet werden. Sind die Betriebssysteme von zentralen Rechnern im lokalen Netz oder im Internet kompromittiert, ist eine sichere Kommunikation nicht mehr möglich.

Die folgenden Abschnitte zeigen auf, nach welchem Schema reale Angriffe ablaufen und welche Ziele dabei verfolgt werden.

8.1 Vorgehensweise bei Angriffen

Der Angriff auf ein Kommunikationsnetz ist der Versuch, *Rechte* auf einzelnen Maschinen zu erlangen und auszubauen. Unter dem Begriff Rechte wird in diesem Zusammenhang die Kontrolle über den Ablauf von Software verstanden. Diese Kontrolle kann durch Ausnutzung von Sicherheitslücken und Manipulation an der Konfiguration eines Betriebssystems oder der darauf aktiven Dienste erlangt werden. Rechte in diesem Sinne schließen z.B. auch die unberechtigte Kenntnis von Paßwörtern oder das Vorhandensein einer manipulierten

Systemkomponente ein, die dem Angreifer unerlaubt Zugriff auf die jeweilige Maschine verschafft.

Der erste Schritt eines Angriffes besteht im Sammeln von Informationen über das Angriffsziel. Diese umfassen neben Maschinennamen und Netzwerkadresse auch die Art und Version des Betriebssystems und der gestarteten Netzdienste.

Im nächsten Schritt wird ein Angriff geplant und durchgeführt. Der Angreifer hat vorher die gesammelten Informationen ausgewertet und die Schwachstellen des Systems identifiziert. Oftmals ergibt sich aus der vorgefundenen Rechnerkonfiguration direkt ein Hinweis für den folgenden Angriff. Konnte z.B. festgestellt werden, daß der TFTP-Dienst unter UNIX nicht abgesichert ist, folgt der Versuch, die Paßwortdatei */etc/password* oder */etc/shadow* von diesem System auf den lokalen Rechner zu kopieren.

An diesem Punkt versucht der Angreifer, Rechte auf dem fremden System zu erwerben. Im angeführten Beispiel würde er versuchen, die gefundenen Paßwortinformationen zu entschlüsseln und auszutesten. Der Zweck dieses Schrittes besteht darin, den Zugriff auf das System zu einem späteren Zeitpunkt zu erleichtern, auch dann, wenn das ursprüngliche Sicherheitsloch nicht mehr besteht.

Eine weitere Möglichkeit besteht im Öffnen einer Hintertür im angegriffenen System, z.B. die unbemerkte Installation eines eigenen Netzdienstes, der dem Angreifer zukünftig jederzeit Zugriff ermöglicht.

Der gesamte Vorgang des Angriffs stellt sich oftmals als Spirale dar: Der Angreifer gewinnt Informationen, führt einen Angriff aus und erlangt Rechte im System. Über diese neuen Rechte, z.B. als regulärer User des Systems, kann er weitere Informationen gewinnen und Angriffe durchführen.

Zwei Vorgänge bilden das Rückgrat eines Angriffs: Erstens das Entschlüsseln von Paßwörtern und zweitens die unbemerkte Modifikation eines Systems zur Ausführung von selbst eingebrachten Programmen, sogenannten trojanischen Pferden. In den folgenden Abschnitten werden diese Mechanismen kurz erläutert.

8.2 Angriffe auf Paßwörter

Die Paßwort-Datenbank eines Systems ist ein bevorzugtes Ziel von Angriffen. Gelingt es einem Angreifer, mittels einer der oben angeführten Methoden diese Datei auszulesen, steht er im Anschluß vor dem Problem, die enthaltenen Einträge zu entschlüsseln.

Einträge der Paßwortdatei eines UNIX-Systems enthalten Benutzernamen, verschlüsseltes Paßwort, User- und Gruppen-ID, Home-Directory und Shell des Benutzers. Enthält ein Eintrag anstelle des verschlüsselten Paßworts das Zeichen „*“, so handelt es sich entweder um einen Systemaccount oder um einen Benutzer, für den kein Login möglich ist. Die letzte Zeile kennzeichnet einen NIS-Client. Abbildung 8.1 zeigt den gesamten Aufbau der Datei.

```
root:QGmukBIhTvyuP:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/usr/lib/news:
uucp:*:10:14:uucp:/var/spool/uucppublic:
operator:*:11:0:operator:/root:/bin/bash
games:*:12:100:games:/usr/games:
man:*:13:15:man:/usr/man:
nobody:*:-1:100:nobody:/dev/null:
ftp:*:404:1::/home/ftp:/bin/bash
guest:*:405:100:guest:/dev/null:/dev/null
test:hjiuuNBHdJiLp:500:100:Testuser:/home/test:/bin/bash
joerg:Pjk78JljklssL:501:100:Joerg Hartmann:/home/joerg:/bin/tcsh
+:*:0:0:::
```

Abbildung 8.1: Aufbau der Paßwort-Datei unter UNIX

Die Verschlüsselung der Paßwörter erfolgt üblicherweise nach dem Crypt - Verfahren, einer Variante des in Kapitel 3.2 dargestellten DES-Algorithmus. Crypt ist eine Einweg-Verschlüsselung, d.h. eine gegebene Zeichenfolge, z.B. das Paßwort eines Benutzers, wird so verschlüsselt, daß eine Umkehr des Verfahrens praktisch nicht möglich ist. Der verwendete Schlüssel entspricht dem eingegebenen Paßwort, die verschlüsselte Fassung wird abgespeichert, niemals der Klartext.

Meldet sich ein Benutzer am System an, wird das eingegebene Paßwort ebenfalls mit sich selbst verschlüsselt und das Ergebnis mit der abgespeicherten Zeichenfolge verglichen. Abhängig vom Ergebnis des Vergleiches erhält der Benutzer dann Zugriff auf das System oder wird abgewiesen.

Ein Angriff auf die Verschlüsselung wird über die Exhaustionsmethode (brute force attack) möglich, das Durchprobieren aller möglichen Schlüssel

innerhalb des Schlüsselraumes. Dieser Angriff führt theoretisch immer zum Ziel, ist jedoch in der Praxis durch Zeit und Hardware-Ressourcen begrenzt .

Bei dieser Methode werden Einträge einer Datenbank mit sich selbst verschlüsselt und das Ergebnis wird ebenfalls mit der abgespeicherten Zeichenfolge verglichen. Die Einträge der Textdatenbank enthalten einige in der EDV häufig als Paßwort verwendete Begriffe. Zusätzlich werden Variationen des Benutzernamens geprüft, z.B. Permutationen der einzelnen Zeichen des Namens (vgl. [Kyas 1998] S.134).

Führt die Verwendung vorgegebener Einträge aus einer Datenbank nicht zum Ziel, besteht die Möglichkeit, mittels eines Hilfsprogrammes Zeichenfolgen zu generieren und diese zu prüfen. Hier können alle auf einem Rechner darstellbaren Zeichen kombiniert werden, sodaß theoretisch das Paßwort des Benutzers auf jeden Fall ermittelt werden kann. Problematisch bei dieser Methode ist die erforderliche Rechenzeit bei Einsatz eines umfangreichen Zeichensatzes, der alle Buchstaben des Alphabets in Groß-/Kleinschrift, die Ziffern 0-9 sowie Satz- und Sonderzeichen enthält. Kurze Paßwörter mit bis zu fünf Zeichen werden jedoch auf einem Pentium II mit 400 MHZ innerhalb von 14 Tagen mit Sicherheit gefunden, selbst wenn beliebige Sonderzeichen verwendet worden sind.

8.3 Trojanische Pferde

Ziel eines Angreifers ist es, permanenten Zugriff auf ein System zu erlangen, selbst dann, wenn die ursprünglich zum Angriff verwendeten Sicherheitslücken geschlossen und die entschlüsselten Paßwörter geändert wurden. Aus diesem Grund erfolgt nach einem erfolgreichen Angriff auf ein

System der Versuch, dort Modifikationen vorzunehmen, die ein späteres erneutes Eindringen erleichtern.

Programme oder Programmteile, die diese Funktion erfüllen, werden als *trojanische Pferde* bezeichnet. Der Begriff kennzeichnete ursprünglich Anwendungsprogramme, die so modifiziert wurden, daß sie unbemerkt neben der eigentlichen Aufgabe noch eine zusätzliche übernehmen können, die dem Angreifer nutzt, z.B. das Ausspähen von Paßwörtern. Im Rahmen dieser Arbeit wird der Begriff etwas weiter gefaßt und dehnt sich auch auf modifizierte Systemdienste aus, die dem Angreifer erlauben, unbemerkt in ein System einzudringen, sowie auf Methoden zur Gewinnung sensitiver Informationen eines Systems (z.B. der Konfiguration von Sicherheitsmechanismen).

Trojanische Pferde werden auf verschiedene Arten eingesetzt. Eine Möglichkeit besteht in der geschickten Nachbildung einer Eingabeaufforderung, die ein Paßwort erfordert, z.B. dem Login-Prompt eines Systems. Der Anwender gibt sein Paßwort ein. Dieses Paßwort wird für den Angreifer gespeichert und im nächsten Schritt wird das ursprüngliche Programm aufgerufen, im Beispiel der Login-Prozeß, und die reguläre Funktion ausgeführt. Die Modifikation bleibt unbemerkt und der Angreifer ist stets im Besitz eines gültigen Paßworts für das System. Eine verfeinerte Variante des Angriffs besteht in der Modifikation von Betriebssystem-Bibliotheken, die beim Start des Login-Prozesses oder ähnlicher Dienste aufgerufen werden.

Die Modifikation eines Netzdienstes ist eine weitere Einsatzmöglichkeit für ein trojanisches Pferd. Nach einem erfolgreichen Angriff wird ein Netzdienst installiert, der dem Angreifer später ohne Eingabe eines Paßworts Zugriff auf das System gewährt. Hier kann z.B. ein abgewandelter Telnet-Daemon zum Einsatz kommen, der nicht auf dem vorgegebenen TCP-Port 23, sondern

zusätzlich zu dem unmodifizierten Dienst auf einem vom Angreifer gewählten Port abläuft. Über diesen Port ist dann eine Anmeldung ohne oder mit einem nicht in der Paßwort-Datei verzeichneten Paßwort möglich.

Durch komplexere Programme können ganze Systeme vom Angreifer ferngesteuert werden. Ein Beispiel für diese Technik sind die trojanischen Pferde *BackOrifice* und *NetBUS*, mit denen alle sicherheitsrelevanten Funktionen eines Systems unter Windows 95/98 bzw. Windows NT vom Angreifer übernommen werden können.

Im ersten Schritt wird ein Hintergrund-Programm als Server installiert, über das der Angreifer später eine Datenübertragung mit dem System initiieren kann. Die Installation des Servers erfolgt unbemerkt vom Benutzer, z.B. beim Aufruf einer präparierten HTML-Seite durch ein Java-Applet, welches Fehler in der Java VM ausnutzt (vgl. Kapitel 11.5.1). Der Server ist für verschiedenen TCP/IP-Ports konfigurierbar, wie in Abbildung 8.2 dargestellt, und kann so eingestellt werden, daß er unsichtbar im Hintergrund arbeitet.

Abbildung 8.2: Konfiguration des NetBUS-Servers

Der Zugriff auf den so präparierten Rechner erfolgt von einer beliebigen Stelle im Internet. Dem Angreifer steht eine graphische Oberfläche zur Verfügung, über die der fremde Rechner vollständig kontrolliert werden kann (vgl. Abbildung 8.3).

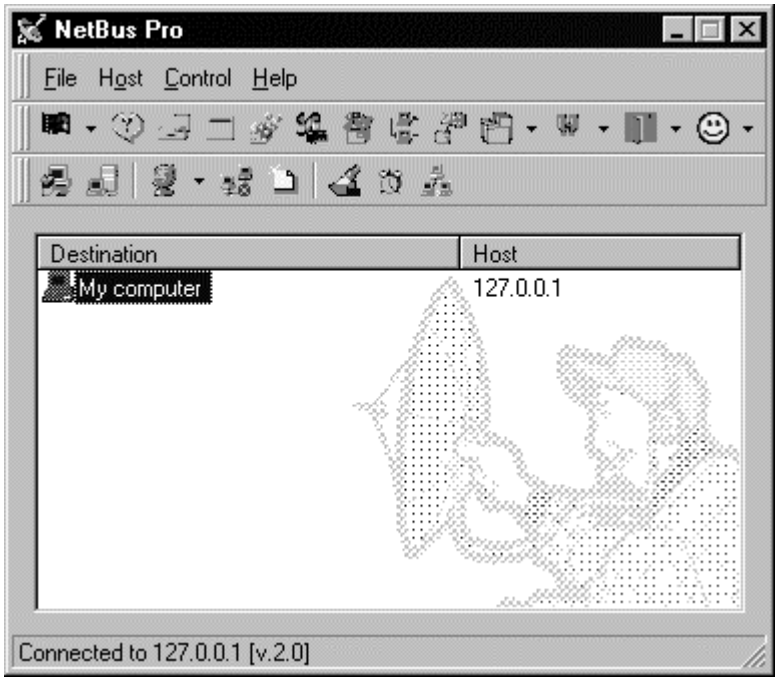


Abbildung 8.3: Oberfläche von NetBUS

Neben den Standardfunktionen der Systemadministration wie z.B. dem Zugriff auf das Filesystem oder dem Neustart des Rechners ist es auch möglich, einzelne TCP/IP-Ports auf den Rechner des Angreifers umzulenken, der dadurch eine falsche Identität annehmen kann. Ebenso ist auch die Umleitung der Ausgabe einzelner Programme auf eine definierte Portadresse möglich (vgl. Abbildung 8.4).

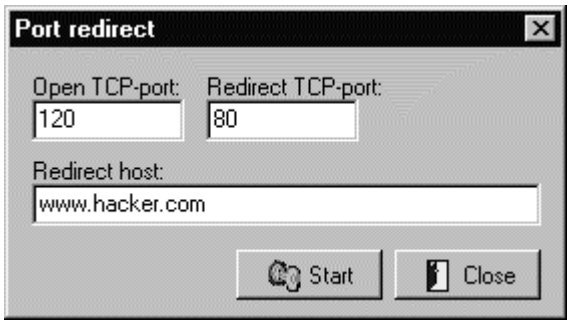


Abbildung 8.4: Umleitung von TCP/IP-Ports

Weitaus gefährlicher ist die Möglichkeit, alle Tastatureingaben auf dem fremden Rechner zu protokollieren. Da dieser Mechanismus unmittelbar auf Ebene des Betriebssystems abläuft, werden die Eingaben für alle aktiven Anwendungen aufgezeichnet, wie in Abbildung 8.5 dargestellt. Unter diesen Eingaben kann sich z.B. auch das Einwahl-Paßwort für den Internet-Provider, die beim Online-Banking verwendete Geheimzahl oder die PIN für eine Chipkarte befinden.

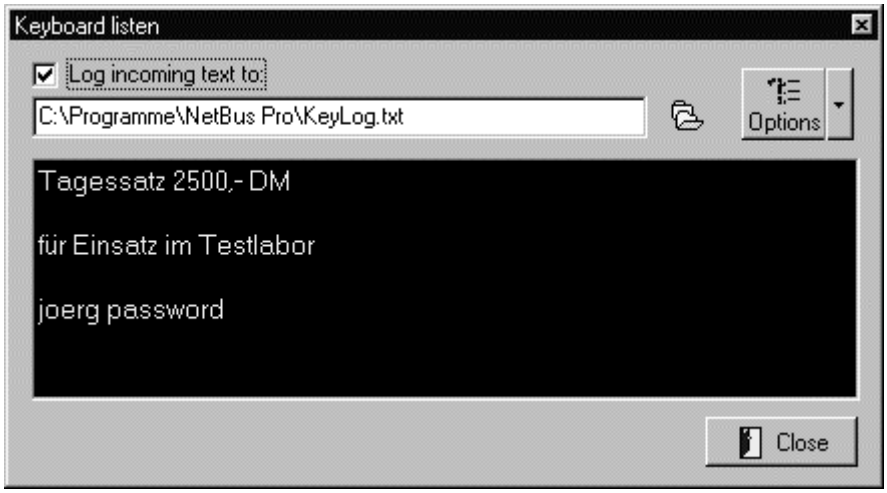


Abbildung 8.5: Aufzeichnung von Tastatureingaben

Die Brisanz solcher Trojaner wie NetBUS wird auch in den folgenden Szenarien deutlich, die den Ablauf eines realen Angriffs aus dem Internet gegen das Betriebssystem Windows NT darstellen. Ziel ist, einige

Sicherheitslücken von Windows NT aufzuzeigen, die im Bezug auf eine Internet-Anbindung dieses Systems besonders kritisch sind.

In Zusammenhang mit diesen Sicherheitslücken werde ich einige Abwehrmaßnahmen vorstellen, mit denen ein System gegen Angriffe aus dem Internet gesichert werden kann. Die hier angeführten Angriffsmethoden und Schutzmaßnahmen sind auf andere Netzwerkbetriebssysteme, z.B. UNIX oder Novell Netware, übertragbar. Weitere Maßnahmen zum unmittelbaren Schutz lokaler Rechner werden in Kapitel 10 dargestellt.

8.4 Ein Angriff auf Windows NT

Der erste Schritt eines Angriffs besteht, wie bereits erwähnt, im Sammeln von Informationen über das anzugreifende System. Wie gewinnt ein Angreifer diese Informationen? Die erste Annäherung an ein System erfolgt meist aufgrund der von einem Nameserver gelieferten Daten. Neben der IP-Adresse des fremden Systems können hier oftmals auch Informationen über den Aufbau des anzugreifenden Netzes gewonnen werden. Abbildung 8.6 stellt die Ausgabe des Nameservers dar, die in diesem Fall bereits Auskunft darüber gibt, welche Dienste von den Rechnern des fremden Netzes unterstützt werden. Im Beispiel läuft ein WWW-Server, der gleichzeitig als Dial-In-Server (Remote Access) und primärer Domänenkontroller (PDC) konfiguriert ist, sowie einige Rechner unter NT und eine Maschine mit Windows 95.

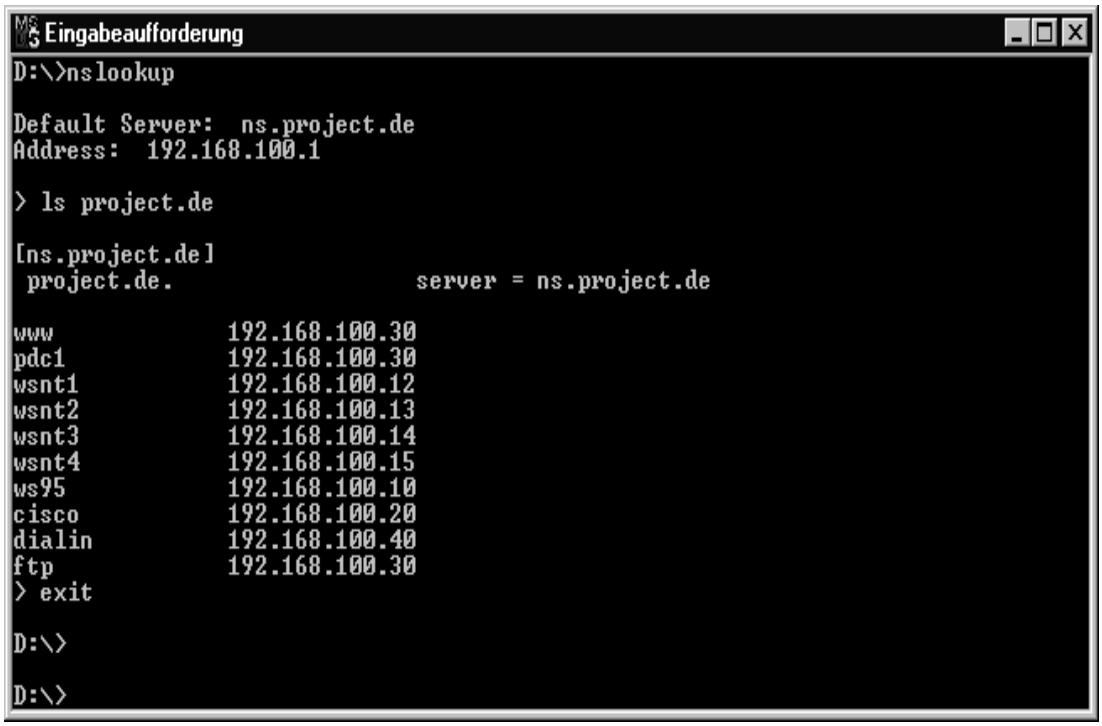


Abbildung 8.6: Informationsquelle DNS

Eine weitere Möglichkeit, an Informationen über Windows-Systeme zu gelangen, ist die NetBIOS-Schnittstelle. NetBIOS ist eine Softwareschnittstelle, über die Verbindungen zwischen Rechnern mit Microsoft-Betriebssystemen hergestellt werden, z.B. zwischen zwei NT-Rechnern. Auf diese Schnittstelle setzen NetBIOS-Anwendungen auf, die für solche Betriebssysteme spezifisch sind, z.B Datei- oder Druckdienste.

Microsoft liefert mit seinen Betriebssystemen einige Tools aus, die in erster Linie zu Diagnosezwecken in lokalen Netzen Verwendung finden. Diese Werkzeuge, z.B. das Programm *nbtstat,* arbeiten jedoch über jede TCP/IP-Verbindung und damit auch über das Internet. Mit Hilfe von *nbtstat* kann sich der Angreifer Informationen über angemeldete Benutzer (im Beispiel der Administrator) oder laufende Dienste (hier der Internet-Information-Server) verschaffen (vgl. Abbildung 8.7).

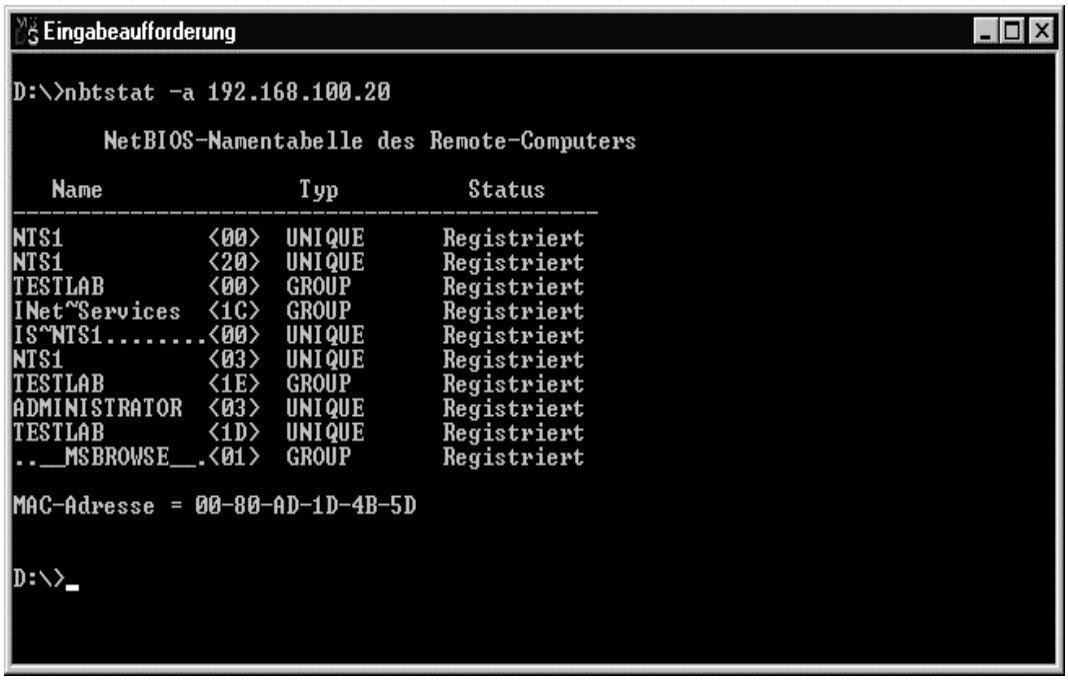
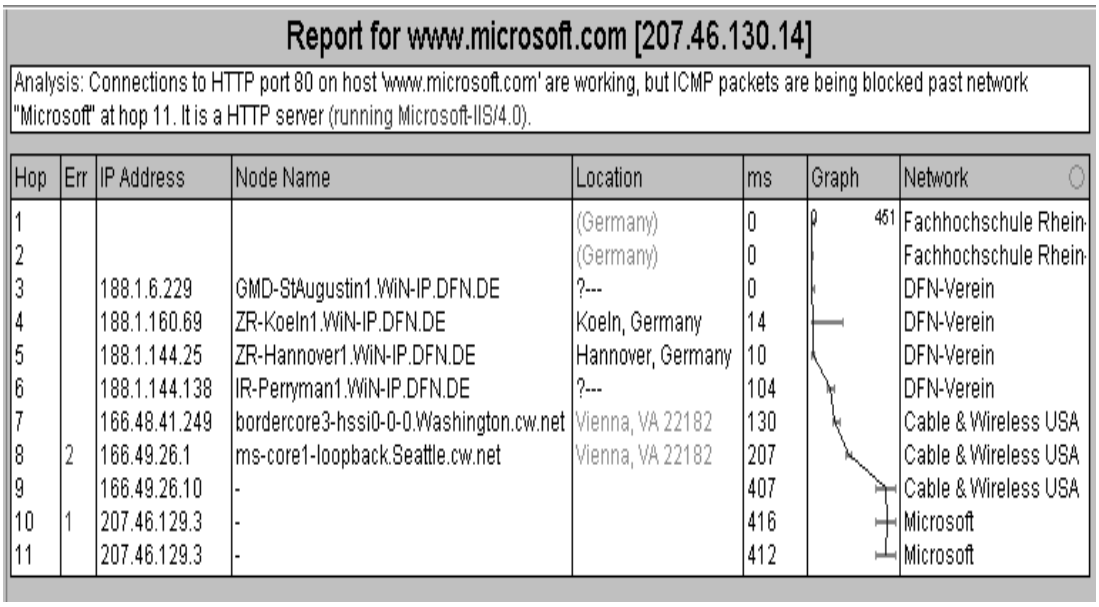


Abbildung 8.7: Informationsquelle NetBIOS

Wenn ein Angreifer festgestellt hat, daß bestimmte Dienste installiert sind, dann besteht der nächste Schritt darin herauszufinden, welche Version eines Dienstes aktiv ist, da jede Version charakteristische Fehler und Schwachstellen enthält. Eine Möglichkeit ist der Verbindungsaufbau zu diesem Dienst. Abbildung 8.8 zeigt in der dritten Zeile die Versionsnummer des FTP-Dienstes (Version 2.0). Aus dieser Information kann der Angreifer schließen, daß hier die veraltete Version 2 des Internet-Information-Server installiert ist, von deren Verwendung Microsoft aufgrund vorhandener Sicherheitsmängel abrät.

Abbildung 8.8: Informationsquelle Versionsmeldung

Eine weitere Informationsquelle können die Router zwischen dem lokalen Netz und dem Internet sein. In Kapitel 7.6 wurden bereits Schwächen des *Router Information Protokoll* (RIP) angesprochen, durch die Routing-Informationen und damit Informationen über die Struktur des lokalen Netzes zugreifbar sind. Auch ohne großen Aufwand kann die Wegwahl durch das Internet in ein fremdes Netz ermittelt werden, Abbildung 8.9 zeigt den Weg zu www.microsoft.com. In diesem Beispiel wird das Netz durch einen Gateway gesichert, der Network Address Translation (vgl. Kapitel 9.2.3) durchführt.



Report for www.microsoft.com [207.46.130.14]

Analysis: Connections to HTTP port 80 on host 'www.microsoft.com' are working, but ICMP packets are being blocked past network "Microsoft" at hop 11. It is a HTTP server (running Microsoft-IIS/4.0).

Hop	Err	IP Address	Node Name	Location	ms	Graph	Network
1				(Germany)	0	0 461	Fachhochschule Rhein
2				(Germany)	0		Fachhochschule Rhein
3		188.1.6.229	GMD-StAugustin1.WiN-IP.DFN.DE	?---	0		DFN-Verein
4		188.1.160.69	ZR-Koeln1.WiN-IP.DFN.DE	Koeln, Germany	14		DFN-Verein
5		188.1.144.25	ZR-Hannover1.WiN-IP.DFN.DE	Hannover, Germany	10		DFN-Verein
6		188.1.144.138	IR-Perryman1.WiN-IP.DFN.DE	?---	104		DFN-Verein
7		166.48.41.249	bordercore3-hssi0-0-0.Washington.cw.net	Vienna, VA 22182	130		Cable & Wireless USA
8	2	166.49.26.1	ms-core1-loopback.Seattle.cw.net	Vienna, VA 22182	207		Cable & Wireless USA
9		166.49.26.10	-		407		Cable & Wireless USA
10	1	207.46.129.3	-		416		Microsoft
11		207.46.129.3	-		412		Microsoft

Abbildung 8.9: Informationsquelle Router

Mit Hilfe von im Internet frei verfügbaren Werkzeugen kann ein Angreifer prüfen, ob bei einem oder mehreren Rechnern bestimmte TCP/IP-Ports ansprechbar sind. So kann ebenfalls festgestellt werden, welche Dienste unterstützt werden. Im Beispiel aus Abbildung 8.10 wird geprüft, ob die Ports 135-139 ansprechbar sind. Dies sind die TCP/IP-Ports für Remote-Procedure Calls und die NetBIOS-Schnittstelle.

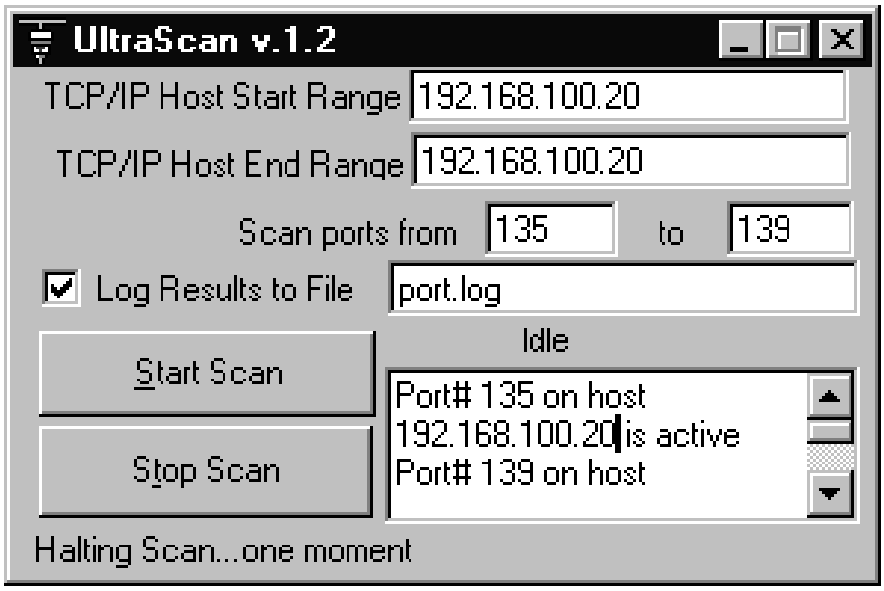


Abbildung 8.10: Informationsquelle Portscanner

Das verwendete Werkzeug, ein sog. *Portscanner* ist einfach aufgebaut. Komplexere Tools (in Abbildung 8.11 das Hackertool Legion) erlauben es, einen ganzen Netzwerkbereich, über den der Angreifer bereits aus dem Domain Name Service Kenntnis hat, auf die Existenz von NetBIOS-Diensten zu testen und freigegebene Verzeichnisse auf allen gefundenen Microsoft-Rechnern anzuzeigen. Ob der Einsatz eines Portscanners oder eines Tools wie Legion bereits als Angriff gewertet wird, hängt vom jeweiligen Administrator ab. Da aktiv Verbindungen zu einem fremden System aufgebaut werden, liegt es nahe, hier von einem Angriff zu sprechen.

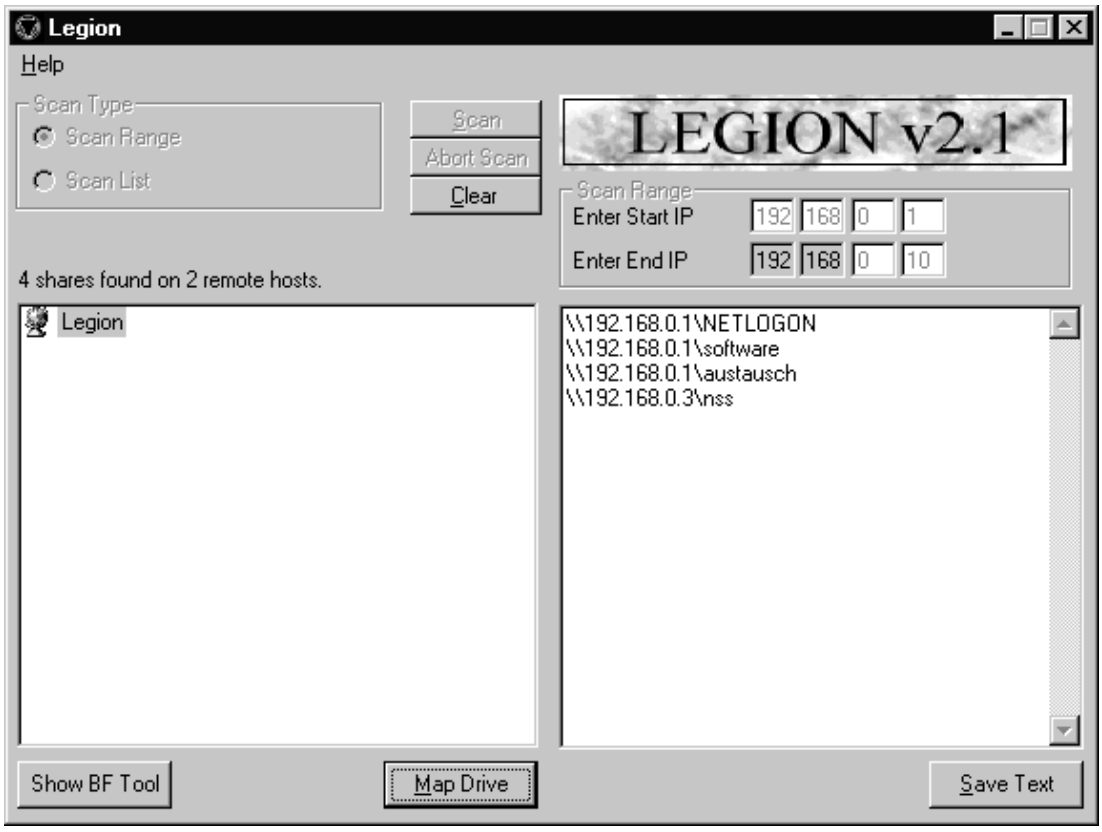


Abbildung 8.11: Informationsquelle Hackertools

Festzuhalten bleibt: Es gibt eine ganze Reihe von Tools, die im Internet frei verfügbar sind und die auch von Anwendern mit geringen Kenntnissen

eingesetzt werden können, um gezielt Informationen über fremde Rechner zu sammeln und Schwachstellen aufzudecken.

Sind genügend Informationen gewonnen und Schwachstellen identifiziert, erfolgt im nächsten Schritt der Angriff. Die hier gesteckten Ziele haben einen direkten Bezug zu den in Kapitel 2.2 angeführten Sachzielen der Informationssicherheit. Angriffe richten sich auf die *Vertraulichkeit*, d.h. die unberechtigte Kenntnisnahme des Inhalts von Dateien und Datenübertragungen, sowie auf die *Integrität*, d.h. die Manipulation von Dateien oder Konfigurationen. Über eine Denial-of-Service-Attack werden darüber hinaus Angriffe auf die *Verfügbarkeit* einzelner NT-Dienste oder ganzer NT-Systeme durchgeführt. Die Häufigkeit dieser Art von Angriffen hat in letzter Zeit stark zugenommen. Sie werden über einen direkten Angriff auf den TCP/IP-Stack von Windows NT realisiert.

Was ein Angreifer bereits mit einfachen Mitteln erreichen kann, läßt sich anschaulich an dem Hackertool NT-InfoScan verdeutlichen. Mit diesem Werkzeug können nicht nur angemeldete Benutzer (wie mit nbtstat), sondern alle auf einer Maschine angelegten Benutzer angezeigt werden. Zusätzlich wird die Zuordnung zwischen dem Namen privilegierter Benutzer und ihrer User-ID geprüft, wie in Abbildung 8.12 dargestellt.

So findet der Angreifer heraus, welche Namen für den Administrator (UID 500) oder den Guest-Account (UID 501) verwendet wurden. Schutzmaßnahmen durch Umbenennen von Accounts werden so unwirksam gemacht. Der Angreifer kann gezielt das Kennwort eines Accounts angreifen.

Abbildung 8.12: Accounts mit ID in NT-InfoScan

Alle vorhandenen Benutzeraccounts werden auf das Fehlen eines Paßworts und auf die Verwendung des Benutzernamens als Paßwort geprüft. Weiterhin kann ein Angriff direkt auf einen bestimmten Account erfolgen. Der Angreifer gibt eine Paßwortliste an und alle darin enthaltenen Einträge werden als Paßwort für den angegebenen Benutzer durchprobiert. Abbildung 8.13 zeigt den Angriff auf das Kennwort des Benutzers *test*.

Von den in Kapitel 8.2 dargestellten Angriffen auf Paßwörter unterscheidet sich dieses Vorgehen dadurch, daß der Angreifer hier nicht zuerst die Windows NT Benutzerdatenbank in seinen Besitz bringen muß, sondern direkt über das Internet angreifen kann. Nachteilig für den Angreifer wirkt sich aus, daß ein Test über das Netz deutlich länger dauert, als der Zugriff auf eine lokal vorliegende Datenbank.

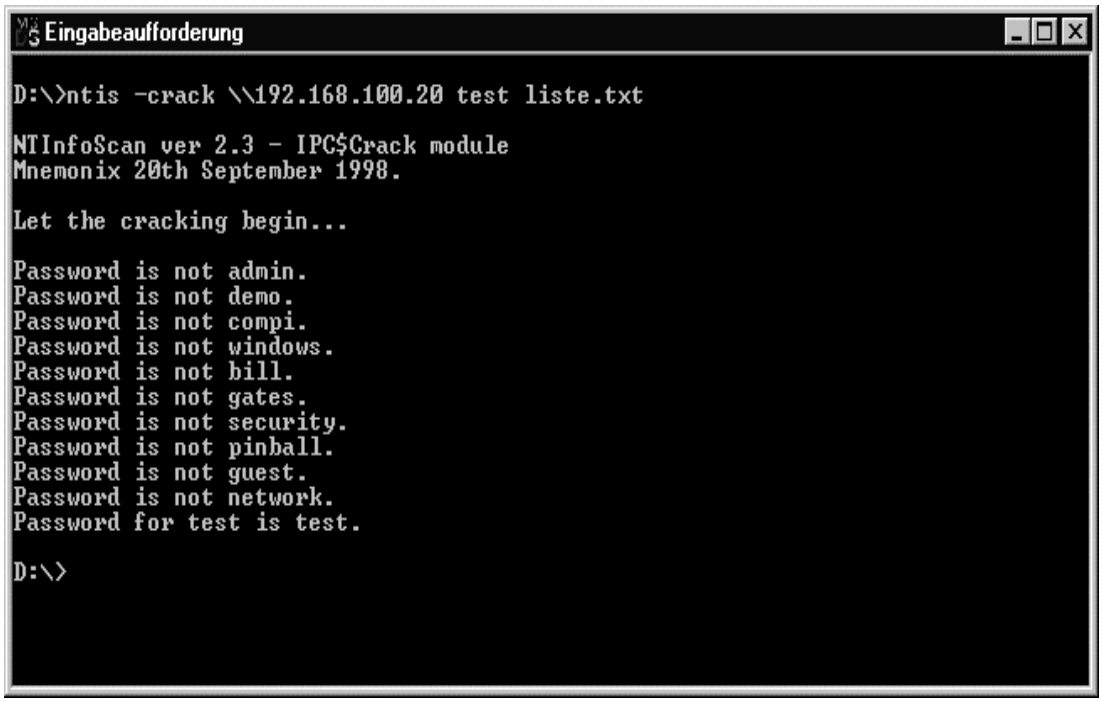


Abbildung 8.13: Wörterbuchangriff auf Windows NT über das Internet

Wenn der Angreifer mit seiner Taktik Erfolg hat, wird er im nächsten Schritt versuchen, Rechte auf dem fremden System zu erwerben. Dies kann schon durch Kenntnis des Paßworts eines privilegierten Accounts mittels NTInfoScan geschehen sein oder nachträglich durch Veränderungen an Systemdateien erfolgen.

Die Veränderungen erstrecken sich auf Dateien, die weitere Informationen über das System offenlegen. Dies sind z.B. ausführbare Dateien, die beim Ändern eines Paßworts durch einen Benutzer ausgeführt werden, oder Dateien, in denen Dienstekonfigurationen gespeichert sind. Hier kann der Angreifer durch Installation eigener Dienste bzw. Abänderung bereits installierter Dienste später leichter Zugriff zum System erhalten. In diesen Bereich fällt auch das in Kapitel 8.3 angesprochene Tool NetBUS zur Fernsteuerung eines Rechners unter Windows NT.

Wenn ein Angriff wie der oben beschriebene ohne Aufwand möglich ist, könnte man daraus schließen, daß Windows NT ein besonders unsicheres

Betriebssystem ist. Tatsächlich ist NT ein relativ junges System, welches sich in stärkerem Maße weiterentwickelt als andere Systeme, wie z.B. UNIX. Mit aktuell ca. 20 Millionen Zeilen Quellcode (Windows NT 5.0, Stand April 1999) ist es darüber hinaus hochkomplex. Sowohl die ständigen Neuerungen als auch die Komplexität und die Abwärtskompatibilität zu älteren und unsicheren Netzdiensten (z.B. Lan Manager) wirkt sich negativ auf die Stabilität aus. Darüber hinaus liegt auch bei Windows NT, wie bei den Vorläufern Windows 3.X und Windows 95, der Fokus auf Benutzerfreundlichkeit, nicht auf erhöhter Sicherheit.

Dies darf jedoch nicht darüber hinwegtäuschen, daß auch über lange Zeit eingeführte Betriebssysteme wie UNIX schwerwiegende Sicherheitsmängel aufweisen. Diese wurden teilweise schon in vorangegangenen Kapiteln angesprochen. Hier wird ein Angreifer mit zu NT-InfoScan vergleichbaren Werkzeugen ähnliche Ergebnisse erzielen.

Im folgenden Abschnitt werden die Schwachstellen von Windows NT kritisch betrachtet und Maßnahmen zum Schutz gegen einen Angriff aus dem Internet erörtert. Die grundlegenden Mechanismen zum Schutz von Kommunikationskanälen werden hier um eine Darstellung zur Sicherung eines Endpunktes der Kommunikationsverbindung im Internet ergänzt. Sie sind zu großen Teilen von Windows NT auf andere Betriebssysteme übertragbar.

8.5 Sicherung von Windows NT

Bei der Installation von Windows NT werden Netzdienste installiert, die gar nicht auf jedem System notwendig sind. So wird beispielsweise der NetBIOS-Serverdienst auf einer NT-Workstation installiert, der nur benötigt wird, wenn Verzeichnisse freigegeben werden sollen. Die Freigabe von Verzeichnissen erfolgt jedoch in der Regel über einen Windows NT Server, nicht über die Workstation. Viele Angriffe erfolgen direkt auf den Server-Dienst. Darunter fällt z.B. auch der Einsatz des oben erwähnten Tools Legion, über das ein Angreifer versuchen kann, unberechtigt auf freigegebene Verzeichnisse zuzugreifen.

Eine einfache Möglichkeit der Sicherung besteht im Lösen der Bindungen zwischen dem Server-Dienst und der TCP/IP-Schnittstelle, wie in Abbildung 8.14 dargestellt. Diese Maßnahme schützt den Serverdienst, jedoch muß ein Angriff nicht unmittelbar auf den einzelnen Dienst erfolgen, sondern kann auch auf bestimmte Schwachstellen der NetBIOS-Architektur gerichtet sein. NT-InfoScan nutzt beispielsweise die Inter-Prozeß-Kommunikation (IPC), um eine sogenannte Null-Session zu etablieren, bei der eine Kontrollverbindung zwischen Rechnern mit NT-Betriebssystem über einen anonymen Account eingerichtet wird.

Abbildung 8.14: Bindungen für NetBIOS-Dienste unter Windows NT

Besseren Schutz bietet eine Unterbrechung von TCP/IP-Verbindungen auf einer tieferen Ebene des Protokollstacks. So kann z.B. die gesamte Bindung der NetBIOS-Schnittstelle über TCP/IP gelöst werden. Diese Maßnahme ist sinnvoll auf Rechnern, die direkt mit dem Internet Kontakt haben, z.B. WWW-Server auf Basis von NT. Verfügt das System über zwei Netzwerk-Interfaces, so kann auf der externen Schnittstelle mit Verbindung zum Internet NetBIOS deaktiviert werden, auf der internen mit Verbindung zum lokalen Netz jedoch weiterhin eine Verbindung über TCP/IP erfolgen (vgl. Abbildung 8.15).

Um den Zugriff auf lokale NetBIOS-Dienste aus dem Internet zu unterbinden, kann die NetBIOS-Scope-ID gesetzt werden. Jedem NetBIOS-Paket im Netz wird dann eine ID hinzugefügt, die zwischen allen Rechnern, die miteinander kommunizieren wollen, einheitlich sein muß. Ein Angreifer aus dem Internet kennt diese ID nicht und kann daher auch keinen Angriff über NetBIOS starten.

Abbildung 8.15: Sperrung von NetBIOS über TCP/IP

Eine weitere Schwachstelle eines Systems mit Internet-Anbindung bilden die TCP/IP-Dienste selbst, z.B. der *Internet Information Server* (IIS) oder ein DHCP-Server. Selbst einfache Dienste wie *chargen*, *echo*, *daytime* und *quote-of-the-day* sind anfällig für Denial-of-Service-Angriffe, werden aber im

normalen Netzbetrieb selten gebraucht und können deaktiviert werden. Ein zusätzliches Problem vieler TCP/IP-Dienste besteht in der Übertragung sensibler Informationen im Klartext, z.B. von Paßwörtern. NetBIOS-Dienste vor der Einführung des Service-Pack 3 für Windows NT übertragen Daten grundsätzlich im Klartext.

Hinzu kommt, daß die Benutzerrechte bei NT nach einer Installation sehr offen eingestellt sind, z.B. erhält jeder Benutzer das Recht, auf eine Maschine im Netz zuzugreifen. Diese Benutzerrechte müssen vom Administrator explizit eingeschränkt werden (vgl. Abbildung 8.16). Dies gilt auch für die Rechte im Dateisystem NTFS. Einige Accounts, z.B. der Guest-Account können auf NT-Workstations vollständig gesperrt werden.

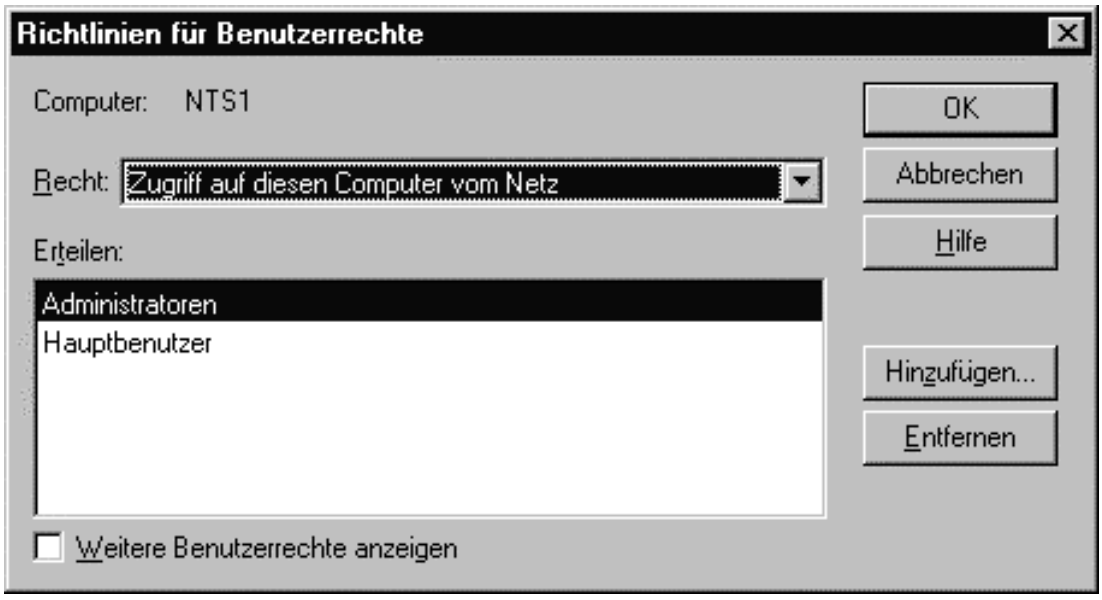


Abbildung 8.16: Benutzerrechte einschränken

Auf die interne Systemdatenbank (Registry) eines NT-Systems ist von externen Rechnern Zugriff möglich, sofern keine besonderen Maßnahmen ergriffen werden. Hier kann ein Angreifer Informationen über die Sicherheitskonfiguration erlangen, die später genutzt werden können. Dies können z.B. die Einstellungen für die minimale Paßwortlänge oder die

Konfiguration von Systemdiensten sein. Die Paßwörter für den Start von Diensten werden ebenfalls unverschlüsselt gespeichert. Eine Sicherung der Registry erfolgt über den Eintrag *SecurePipeServers*, wie in Abbildung 8.17 dargestellt, durch den der Zugriff unterbunden wird.

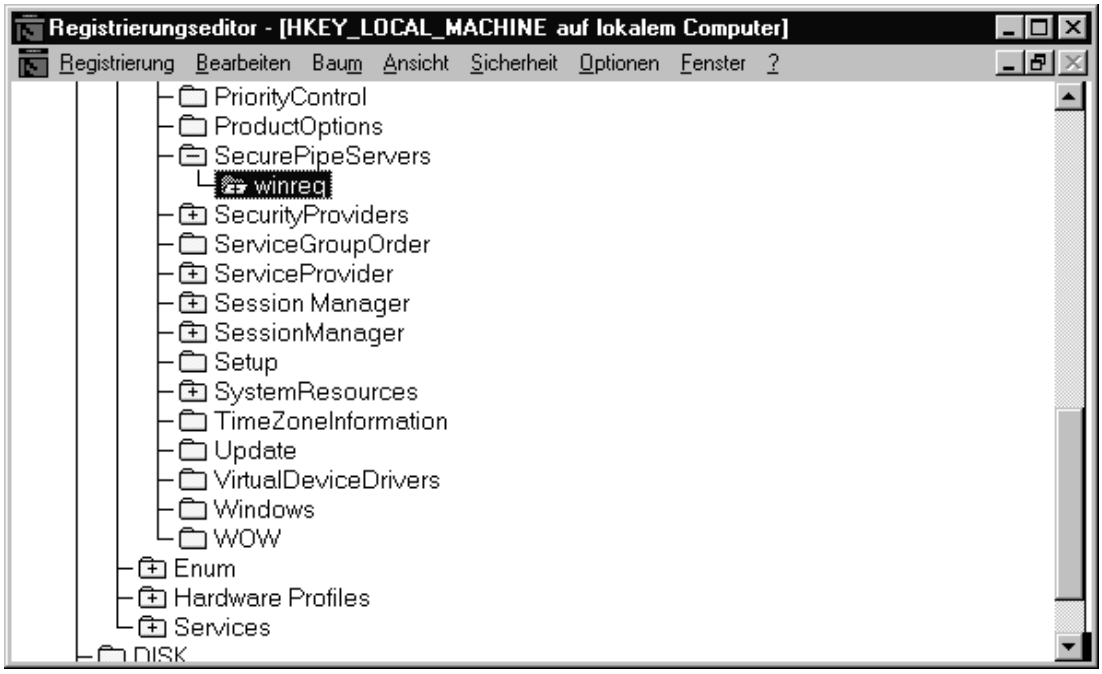


Abbildung 8.17: Remote-Zugriff auf die Registry

Komplexe Serverdienste laufen nach der Installation oftmals ungesichert ab. So ist beispielsweise der Zugriff durch anonyme Benutzer erlaubt, und Paßwortübertragungen erfolgen im Klartext. Einige sicherheitskritische Dienste sind so konfiguriert, daß sie keinen Schutz gegen unerlaubten Zugriff bieten, so ist z.B. bei SNMP als Community-Name *public* eingetragen, so daß alle Systeme im Netz Lesezugriff auf die Konfiguration dieses Rechners haben. Wird diese Einstellung beibehalten, können von anderen Rechnern aus sensible Informationen des eigenen Netzes abgefragt werden. Eine Sicherung des Dienstes erfolgt durch Verwendung eines neuen Community-Namens, wie in Abbildung 8.18 dargestellt. Zusätzlich sollten die Rechner angegeben werden, mit denen kommuniziert werden darf.

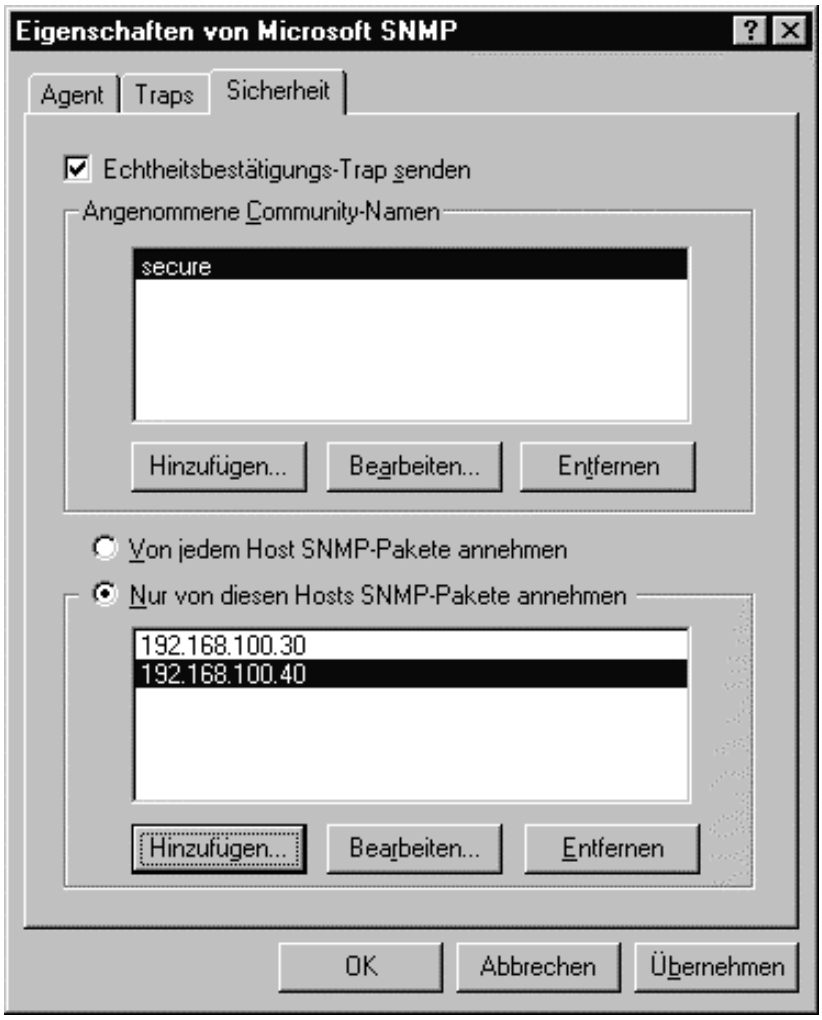


Abbildung 8.18: Sicherung von SNMP

Eine ähnliche Problematik zeigt sich auch bei Anwendungsprogrammen, z.B. dem Internet Explorer. Hier gibt es unüberschaubar viele Konfigurationsmöglichkeiten, und die Voreinstellungen nach der Installation entsprechen dem niedrigsten Sicherheitsniveau. Da der Anwender in der Regel von dieser Vielfalt überfordert wird, kann der Administrator bei einer automatischen Installation einige Parameter voreingstellen.

Zusätzlicher Schutz für Maschinen mit Internet-Anbindung wird durch Sperrung nicht benötigter TCP/IP-Ports erreicht. Wird auf einer Maschine nur der WWW-Server ausgeführt, können alle Ports bis auf Port 80 (HTTP)

gesperrt werden. Windows NT Systeme enthalten einfache Paketfilter, über die dies ohne Zusatzsoftware erreicht werden kann, jedoch enthalten diese Filter nur wenige Konfigurationsmöglichkeiten. In Abbildung 8.19 ist eine Konfiguration dargestellt, mit der alle Ports bis auf Port 80 blockiert werden. Eine weitere Möglichkeit besteht in der Sperrung sicherheitsrelevanter Ports, z.B. der NetBIOS-Ports, während der Zugriff auf alle anderen Ports erlaubt wird.

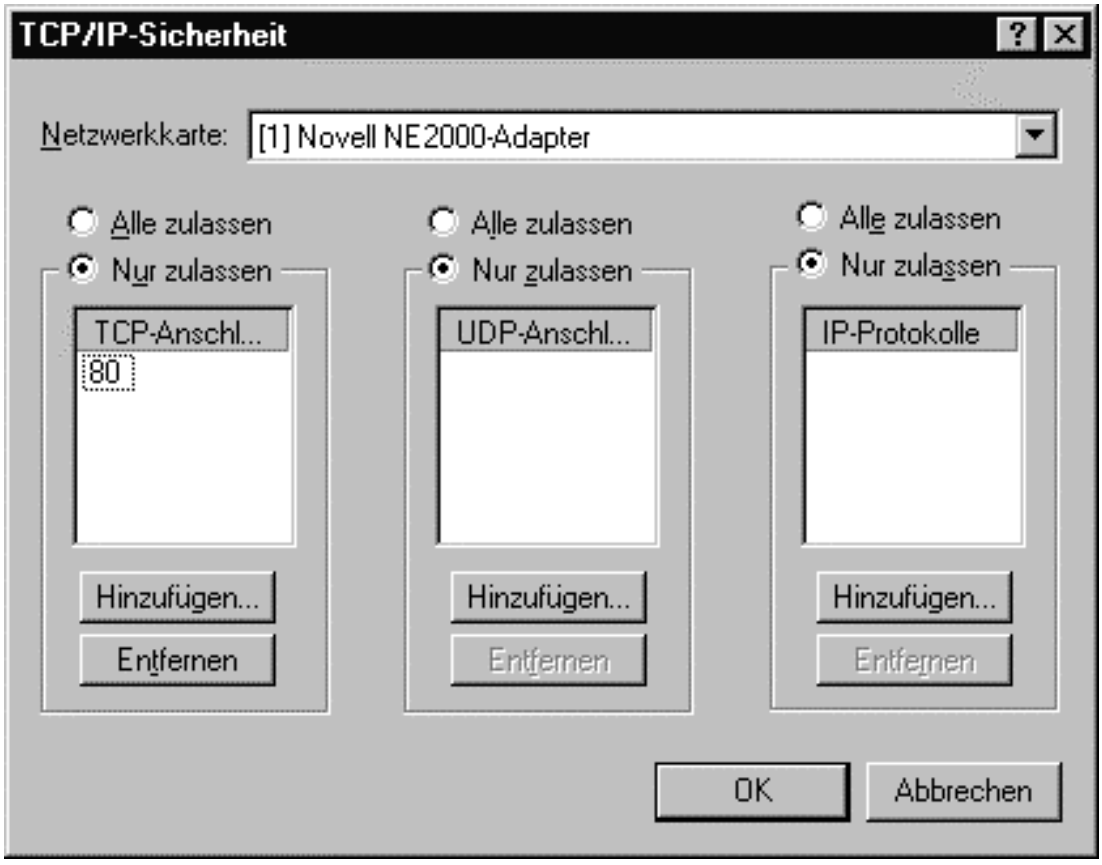


Abbildung 8.19: Sperrung nicht verwendeter TCP/IP-Ports

Deutlich bessere Konfigurationsmöglichkeiten für Windows NT Serversysteme bietet der *Routing and Remote Access Service*, der aus dem ehemaligen Multiprotokoll-Router Steelhead entstanden ist und kostenlos von Microsoft bezogen werden kann.

Mit dem Mechanismus der Paketfilterung können auch Routing-Informationen des lokalen Netzes geschützt werden. Das Router Information Protokoll (RIP) arbeitet auf UDP Port 520. Wird dieser Port ausgehend gesperrt, kann der eigene Router nur Informationen empfangen, jedoch keine weitergeben. Dieses Verhalten ist natürlich nur unter bestimmten Bedingungen wünschenswert. Sinnvoller ist die Einschränkung auf bestimmte Router, wie in Abbildung 8.20 dargestellt. Wenn mit externen Routern kommuniziert wird, dann sollten diese Router explizit angegeben werden.

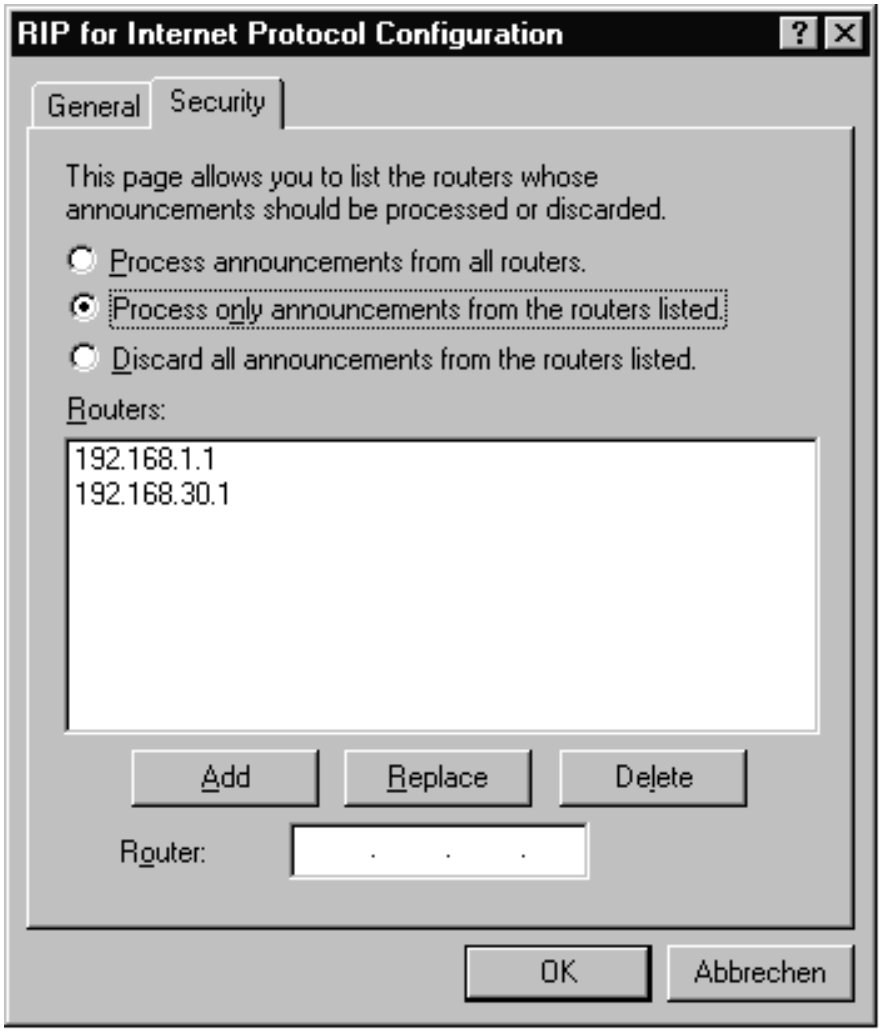


Abbildung 8.20: Routing-Konfiguration sichern

Über die genannten einfachen Maßnahmen hinaus sollten besonders sensible Systeminformationen, z.B. die Benutzer-Account-Datenbank zusätzlich gesichert werden. Service Pack 3 enthält z.B. das Utility SYSKEY, mit dem die Datenbank vollständig verschlüsselt werden kann. Ein Angreifer, der auf diese Datenbank Zugriff erlangt, hat es dann erheblich schwerer, die in der Datenbank enthaltenen Informationen zu verwerten.

Weiterhin ist eine Einstellung der sicherheitsrelevanten Parameter von NT sinnvoll. Mit dem *Security Configuration Editor* (Service Pack 4) können diese Parameter geprüft und eingestellt werden, z.B. der Zugriff auf die Registry oder das SMB-Signing, eine digitale Signatur für SMB-Pakete, wie in Abbildung 8.21 gezeigt.

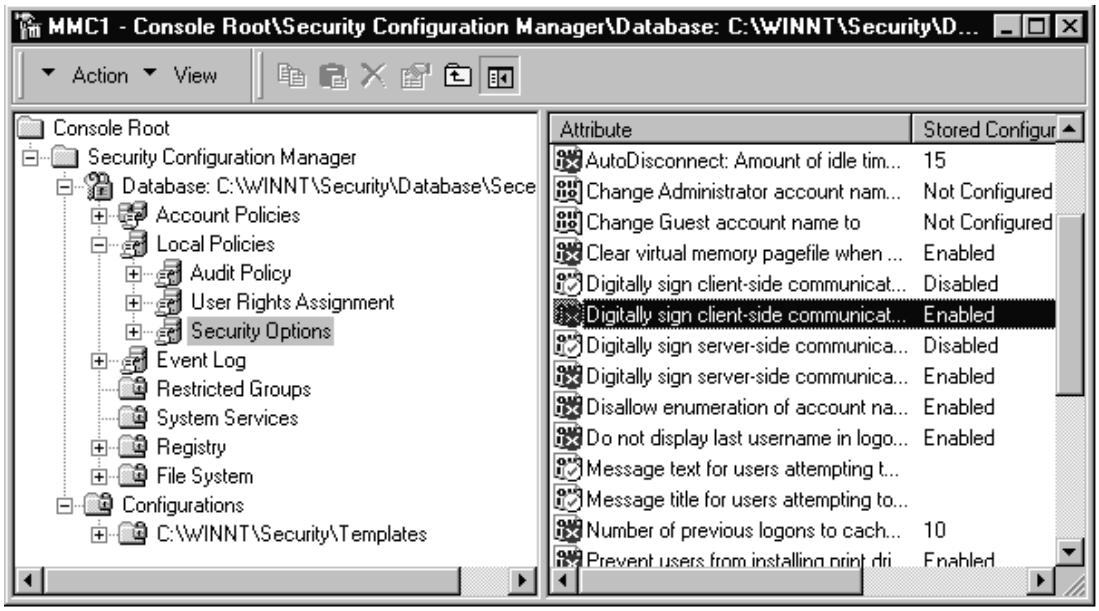


Abbildung 8.21: Sicherheitseinstellungen prüfen

Ein weiterer wichtiger Schritt zu einen sicheren System besteht in der Protokollierung von sicherheitskritischen Operationen. Einige Verletzungen der Sicherheitsrichtlinien können über das Ereignisprotokoll erfaßt werden, z.B. mehrfache Login-Versuche mit falschem Paßwort, andere über externe Tools. Ein Angriff über NT-InfoScan wäre an dieser Stelle schon bemerkt

worden, da für den Benutzer *test* mehrfach ein ungültiges Kennwort ausprobiert wurde.

Um einen Angriff und seine Auswirkungen festzustellen, empfiehlt es sich, die Systemintegrität in regelmäßigen Abständen zu prüfen. Dies schließt insbesondere sicherheitsrelevante Dateien ein, die auf Veränderungen geprüft werden sollten. Einige Dateien, z.B. solche, die ausgeführt werden, wenn ein Benutzer sein Paßwort ändert, sind ein bevorzugtes Angriffsziel für Hacker und werden oft durch trojanische Pferde ersetzt, die für den Angreifer Paßwörter mitlesen und speichern. An dieser Stelle müssen externe Tools eingesetzt werden, da gängige Betriebssysteme die erforderlichen Mechanismen nicht enthalten.

Durch Prüfung und Protokollierung kann ein Angriff zwar nicht verhindert, jedoch bemerkt werden. Sind zusätzlich einige oder alle der angeführten Schutzmaßnahmen aktiv, wird ein Angriff über NetBIOS, z.B. der NT-InfoScan-Angriff, abgeblockt. Abbildung 8.22 stellt ein System dar, bei dem die Bindungen zwischen NetBIOS und TCP/IP gelöst wurden. Hier können noch Internet-Dienste wie z.B. FTP oder WWW ausgeführt werden, der Zugriff auf einen NetBIOS-Dienst schlägt aber fehl und daher auch der Angriff mittels NT-InfoScan.

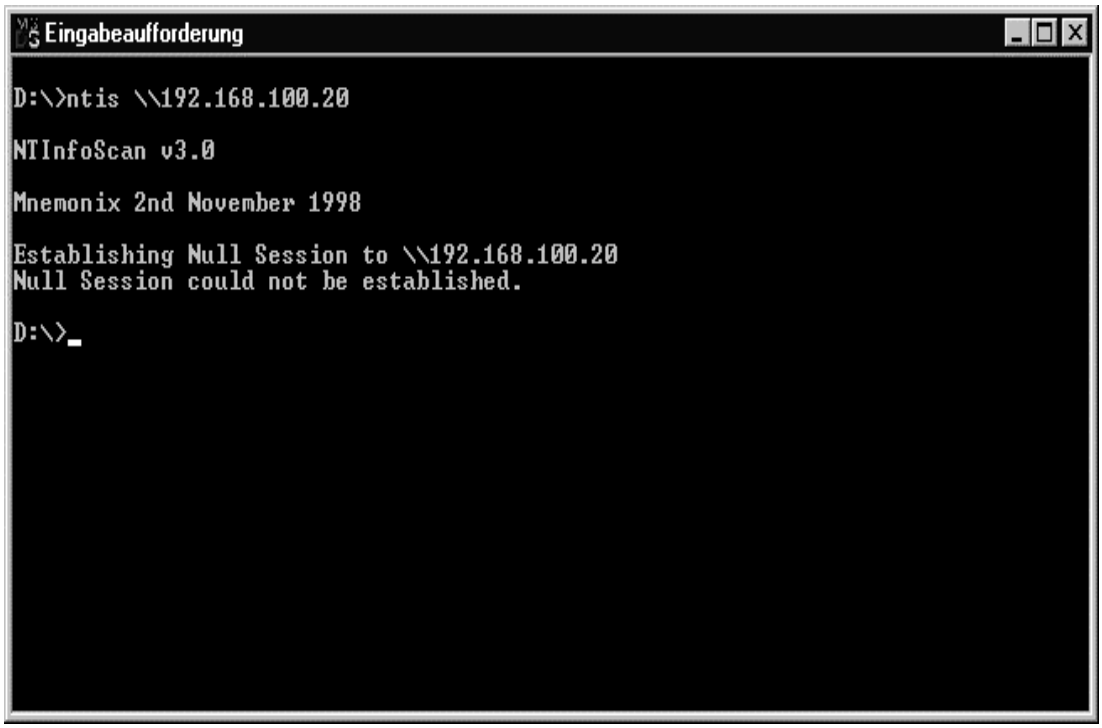


Abbildung 8.22: Abblocken von NT-InfoScan nach der Sicherung

Die hier geschilderten Mechanismen zur Absicherung einzelner Rechner müssen um Mechanismen zur Sicherung von Kommunikationsverbindungen zwischen diesen Rechnern ergänzt werden. Im nächsten Kapitel folgt daher eine systematische Darstellung von Maßnahmen zur Sicherung der Kommunikation.

9 Sicherung der Kommunikation

Maßnahmen zum Schutz der Internet-Kommunikation basieren zu einem großen Teil auf Verschlüsselung und digitaler Signatur. Verschlüsselung gewährleistet Vertraulichkeit der Kommunikation, unberechtigten Dritten bleibt der Inhalt der Kommunikation verborgen. Digitale Signatur sichert die Integrität und Authentizität der übertragenen Informationen, Veränderungen werden bemerkt und Sender sicher identifiziert. Neben Verschlüsselung und Signatur sichern weitere Mechanismen das Umfeld der Kommunikation, d.h. Rechner, lokale Netze, Vermittlungsstellen und einzelne Internet-Dienste.

Die folgenden Abschnitte liefern einen vollständigen Überblick über gängige Verfahren und Mechanismen zur Sicherung der Internet-Kommunikation. Auch hier folge ich der Systematik vorangehender Kapitel und gliedere anhand der verschiedenen Schichten des OSI-Modells.

9.1 Netzwerkzugriff (OSI-Schicht 1 und 2)

In Kapitel 5.1 wurde bereits angesprochen, daß die Schichten 1 und 2 des OSI-Modells in TCP/IP nicht realisiert sind. Schutzmechanismen auf dieser Ebene fallen also nur bedingt in den Bereich der Internet-Sicherheit.

Die Datenübertragung auf den beiden untersten Ebenen des OSI-Modells wird heute über die Netzwerk-Hardware realisiert. Moderne Netzwerkkarten sind in der Lage, die notwendigen Zugriffsverfahren wie z.B. CSMA/CD, selbständig durchzuführen. Verschlüsselungsverfahren auf Schicht 2 können entweder durch spezielle Hardware im Endgerät ausgeführt werden, oder durch Hardware, die auf dieser Schicht Vermittlungsaufgaben zwischen Netzen durchführt, z.B. ein Switch oder eine Bridge.

Die Vielzahl möglicher Medien zur Datenübertragung, darunter Glasfaser, Richtfunk, Infrarot oder Kupferkabel hat zur Folge, daß Schutzmechanismen auf dieser Ebene nicht standardisiert sein können.

9.2 Vermittlung (OSI-Schicht 3 und 4)

9.2.1 IPSEC

Schutzmechanismen auf IP-Ebene werden im vorgeschlagenen Standard IPSEC (Internet Protocol Security) definiert. Eine zentrale Rolle kommt dabei den Erweiterungen der derzeit aktuellen Version 4 des Internet Protokolls zu, die auch in die kommende Version 6 Einzug halten werden. Als Erweiterung im Bereich der Sicherheit werden der *Authenticication Header* (AH) und der *Encapsulation Security Payload* (ESP) eingeführt.

Über den *Authenticication Header* wird die Integrität und Authentizität eines Datenpaketes sichergestellt. Zu diesem Zweck wird eine digitale Signatur (vgl. Kapitel 3.6) über das Datenpaket erstellt, die vom Empfänger geprüft werden kann. Als Hashfunktion findet MD5 Verwendung, die Signatur kann über verschiedene Verfahren erfolgen.

Der *Encapsulation Security Payload* sichert zusätzlich zum AH auch die Vertraulichkeit der übertragenen Daten. Dies geschieht entweder im *Transportmodus*, bei dem nur die Nutzdaten verschlüsselt werden, oder im *Tunnelmodus*, bei dem ein Datenpaket vollständig verschlüsselt wird, inklusive der Adreß- und Absenderinformationen. Abbildung 9.1 verdeutlicht die beiden Modi der Sicherung.

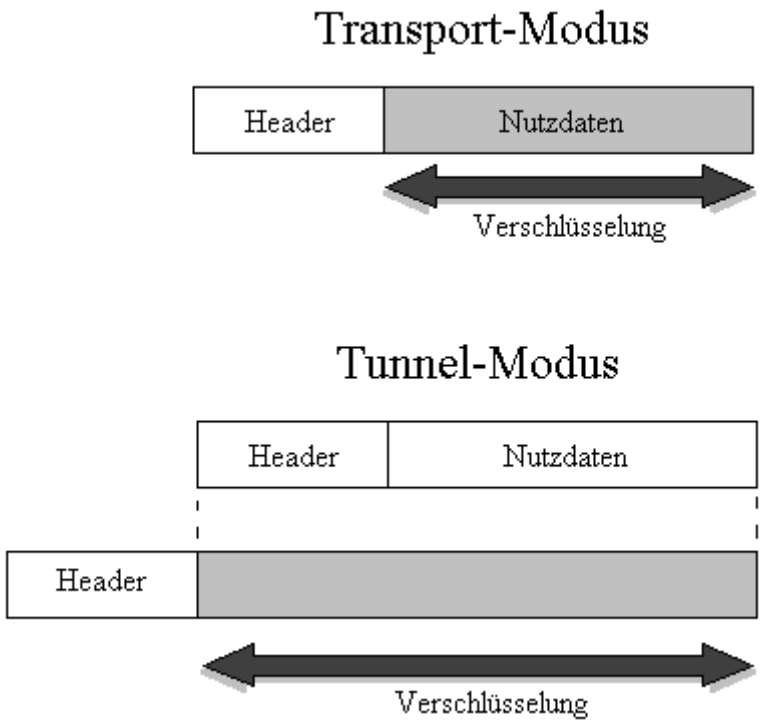


Abbildung 9.1: Transport- und Tunnelmodus

Verschiedene Verfahren, die die an der Kommunikation teilnehmenden Maschinen miteinander aushandeln, können zur Verschlüsselung eingesetzt werden, darunter auch DES oder IDEA. Die benötigten Schlüssel werden über ein eigenes Austauschprotokoll vereinbart. Dieses Austauschprotokoll wird durch ein weiteres Protokoll zur Verwaltung von Schlüsseln ergänzt. Aktuell ist hier das Protokoll ISAKMP (Internet Security Association and Key Management Protocol) definiert. ISAKMP implementiert die Generierung von Schlüsseln, nicht jedoch den Schlüsselaustausch. Verschiedene Austauschprotokolle sind über ISAKMP möglich, im Mittelpunkt der Diskussion steht momentan das auf dem Diffie-Hellman-Verfahren (vgl. Kapitel 3.3) basierende OAKLEY-Protokoll.

IPSEC kann Vertraulichkeit, Integrität und Authentizität der übertragenen Daten gewährleisten. Es hat jedoch noch einige Schwächen, z.B. die fehlende Public Key Infrastruktur, ohne die eine Prüfung der eingesetzten öffentlichen Schlüssel nicht möglich ist.

Ein deutlicher Vorteil ist der offene Standard: IPSEC kann von verschiedenen Herstellern implementiert werden, so daß die unterschiedlichen im Internet eingesetzten Produkte miteinander kombiniert werden können. Durch seinen modularen Aufbau ist IPSEC sehr flexibel und kann leicht an verschiedene Sicherheitsanforderungen, z.B. neue Verschlüsselungsalgorithmen angepaßt werden.

9.2.2 VPNs und Server Gated Cryptography

Ein *Virtual Private Network* (VPN) sichert die Kommunikation zwischen Rechnern über ein unsicheres öffentliches Netz wie z.B. das Internet. Durch ein VPN werden bestehende öffentliche Netze zur privaten Kommunikation genutzt. Da sowohl öffentliche als auch private Daten über ein Medium transportiert werden sollen, müssen die privaten Daten durch Verschlüsselung geschützt werden.

Zur Verschlüsselung von Nutzdaten können verschiedene Standards implementiert werden, z.B. IPSEC oder das von Microsoft unterstützte Point-to-Point Tunneling Protocol (PPTP). Häufig werden jedoch von einzelnen Herstellern proprietäre Verfahren eingesetzt. Generell wird an dieser Stelle zwischen dem Einsatz einer Block- oder Stromchiffre (vgl. Kapitel 3.1) unterschieden. Werden einzelne Pakete des Datenstromes als Block verschlüsselt, ergibt sich eine höhere Ausfallsicherheit, jedoch auch eine größere Anfälligkeit gegen Replay-Angriffe (vgl. Kapitel 6.7). Die Verwendung von Stromchiffren erzeugt einen kontinuierlichen Datenstrom mit höherem Widerstand gegen kryptographische Analysen, z.B. durch einen Known-Plaintext Angriff.

Auch bei einem VPN stellt sich die Frage, ob lediglich die Nutzdaten verschlüsselt werden sollen, oder ein vollständiges IP-Paket inklusive Protokollheader. Die vollständige Verschlüsselung ist etwas aufwendiger, stellt aber sicher, daß nicht erkennbar ist, welche Maschine innerhalb des Zielnetzwerkes angesprochen werden soll.

Der Aufbau eines VPN kann über unterschiedliche Topologien erfolgen. Bei einer *Site-Site* Verbindung sind zwei lokale Netze miteinander über ein als Gateway arbeitendes Paar von VPN-Servern verbunden. Die Verschlüsselung der Kommunikation erfolgt ausschließlich zwischen den Gateways, daher spricht man hier auch von Server Gated Cryptography. Bei diesem Modell sind keine Änderungen an den einzelnen Rechnern des lokalen Netzes notwendig, allerdings läuft die Datenübertragung innerhalb des Netzes weiterhin ungeschützt ab.

Der Site-Site Architektur steht die direkte Verbindung zwischen zwei Rechnern (*Host-Host*) gegenüber. Hier erfolgt die Verschlüsselung auf jedem an der Kommunikation beteiligten Rechner. Der dadurch entstehende höhere Verwaltungsaufwand wird durch den Vorteil der sicheren Kommunikation auch innerhalb des lokalen Netzes ausgeglichen. Eine Mischform stellt das *Host-Site-Modell* dar. Hier erfolgt die Verschlüsselung zwischen einem VPN-Gateway und einem einzelnen Rechner innerhalb eines lokalen Netzes.

Die Vorteile eines VPN bestehen in der Nutzung vorhandener Netzanbindungen und der Transparenz des Verfahrens gegenüber Anwendungen, die mittels TCP/IP kommunizieren.

Die Kosten einer Internet-Verbindung liegen deutlich unter den Kosten einer permanenten Verbindung, z.B. zwischen zwei Firmenstandorten. Andererseits stellen öffentliche Netze keine garantierten Bandbreiten zur

Verfügung, und der Zugriff auf diese Netze ist nicht kontrollierbar. Der durch die Verschlüsselung erzeugte Overhead bewirkt eine Verzögerung bei der Übermittlung der Daten, insbesondere bei Verwendung des Tunnelmodus.

9.2.3 Firewalls und Proxies

Als *Internet-Firewall* wird ein zwischen dem Internet und dem lokalen Netz plazierter Rechner bezeichnet, der den eingehenden und ausgehenden Datenstrom nach festgelegten Kriterien filtert. Die *Filterkriterien* bestimmen einerseits, auf welche Internet-Dienste ein Teilnehmer innerhalb des lokalen Netzes zugreifen darf und andererseits, auf welche internen TCP/IP-Dienste aus dem Internet Zugriff möglich sein soll.

Firewalls werden nach ihrer Funktionsweise unterschieden. Einfache Firewalls auf der Basis von Paketfiltern kontrollieren den Datenstrom anhand statischer Regeln, die festlegen, welche IP-Pakete passieren dürfen. Die Filterung erfolgt aufgrund der Absender- und Zieladresse sowie der ein- und ausgehenden Portadressen. Firewalls dieser Art arbeiten zumeist auf den OSI-Schichten 3 und 4. Eine Filterung aufgrund von Hardware-Adressen (Schicht 2) ist jedoch ebenfalls möglich. Da Filterregeln statisch sind, einige Internet-Dienste (z.B. RPC) jedoch mit dynamischen Portzuweisungen arbeiten, reicht diese Maßnahme allein noch nicht zur Sicherung eines Netzes aus. Darüber ergibt sich in größeren Netzen eine unübersichtlich große Zahl von Filterregeln.

Firewalls auf Schicht 5 werden als *Circuit Gateway* bezeichnet. Hier werden Daten aus dem lokalen Netz an einen bestimmten Port des Gateway gesendet, der die Anfrage entgegennimmt und prüft. Im nächsten Schritt wird eine Verbindung zum Zielsystem im Internet mittels einer neuen Portadresse hergestellt. Der Gateway übernimmt die Vermittlung zwischen

ein- und ausgehender Verbindung. Da keine direkte Verbindung zwischen lokalen Rechnern und Rechnern im Internet besteht, ist diese Methode sicherer als eine einfache Paketfilterung. Nachteilig wirkt sich die Tatsache aus, daß zusätzliche Software zur Umsetzung der Anfragen auf den Rechnern des lokalen Netzes installiert werden muß.

Die Umsetzung von IP- und Portadressen wird als *Network Adress Translation* (NAT) bezeichnet und kann *dynamisch* oder *statisch* erfolgen. Die dynamische Umsetzung erfolgt in der Regel, wenn die Zahl der im lokalen Netz zur Verfügung stehenden IP-Adressen nicht ausreicht. Hier wird nur eine registrierte IP-Adresse benötigt, über die der Gateway mit dem Internet verbunden ist. Rechner des lokalen Netzes erhalten inoffizielle, nicht registrierte Adressen aus einem Bereich, der im Internet nicht verwendet wird. Bei jedem Zugriff auf das Internet wird die Anfrage des lokalen Client umgesetzt und über die registrierte IP-Adresse des Gateway weitergeleitet. Die Antwort eines Servers aus dem Internet geht den umgekehrten Weg. Der Gateway verwaltet die zum Verbindungsaufbau benötigten Informationen über eine Tabelle, die eine eingehende Portadresse der IP-Adresse eines Client zuweist. Der Server kann so eingehende Verbindungen aufgrund der Portadresse an den richtigen lokalen Rechner vermitteln.

Eine statische Adreßumsetzung wird verwendet, um lokale TCP/IP-Dienste vor dem direkten Zugriff aus dem Internet zu schützen. Um diesen Modus einzusetzen, werden dem Gateway mehrere registrierte IP-Adressen zugewiesen, die dieser über eine Tabelle einzelnen Servern im lokalen Netz zuordnet. Interne Dienste sind so aus dem Internet nur über den Gateway ansprechbar, ein direkter Verbindungsaufbau wird abgelehnt.

Eine Erweiterung des Circuit Gateway stellt der *Application Level Gateway* dar. Während bei Einsatz des Circuit Gateway noch Veränderungen am TCP/IP-Stack des Clients nötig sind, ist der Application Level Gateway für

die einzelnen Clients nicht von einem Server im Internet zu unterscheiden. Daher sind auch keine Änderungen am Client notwendig. Dieser Gateway arbeitet auf OSI-Schicht 7, die Schichten 3 und 4 sind für den Client transparent.

Firewalls, die auf den OSI-Schichten 5 und höher arbeiten, d.h. Circuit- und Application Level Gateways, werden in der Regel als *Proxy-Systeme* bezeichnet, da sie stellvertretend für den eigentlichen Server die Anfragen von Client-Rechnern entgegennehmen.

Neben der Aufgabe der Umsetzung von IP- und Portadressen führen Proxy-Systeme häufig noch zusätzliche Funktionen aus. Eine weitere zentrale Funktion von WWW-Proxies besteht z.B. in der Zwischenspeicherung von einmal abgerufenen HTML-Dokumenten, die so bei einer erneuten Anfrage schneller zugestellt werden können.

Anhand ihrer Positionierung im Netz werden weitere Unterteilungen im Bereich der Firewalls vorgenommen. Im einfachsten Fall wird der Firewall direkt zwischen Internet und lokalem Netz plaziert (*dual-homed host*) und erfüllt im wesentlichen Routing-Funktionen mit Paketfilterung zwischen den Netzen. Der Einsatz eines oder mehrerer Proxy-Dienste auf dem Firewall ist möglich. Ein erfolgreicher Angriff auf den Firewall öffnet jedoch direkt den Weg in das lokale Netz.

Eine Verbesserung ergibt sich durch den Einsatz eines *Bastion-Host*, d.h. eines Rechners im lokalen Netz, der stellvertretend für anderer Rechner des Netzes TCP/IP-Dienste im Internet anbietet. Die Absicherung dieses Rechners erfolgt ebenfalls über einen Firewall. Bei dieser Lösung ist nach einem erfolgreichen Angriff auf den Firewall noch nicht das gesamte Netz schutzlos, sondern zunächst nur der Bastion-Host.

Eine Erweiterung des Konzeptes bildet der Einsatz eines überwachten Teilnetzes (*screened subnet*). Bei diesem Konzept wird eine sogenannte *demilitarisierte Zone* zwischen Internet und lokalem Netz eingerichtet. Bei diesem Konzept befinden sich einige weniger wichtige Server des lokalen Netzes zwischen Firewall und Bastion-Host. Im Idealfall wird ein zweites Netz zwischen diesen beiden Rechnern mit einem weiteren Bastion-Host aufgebaut, so daß der Angreifer weitere Anstrengungen unternehmen muß, um in sensible Bereiche im lokalen Netz vorzustoßen.

9.3 Anwendung (OSI-Schichten 5 bis 7)

9.3.1 SSL

Die *Secure Sockets Layer* (SSL), eine Entwicklung der Firma Netscape, arbeiten auf OSI-Schicht 5 und sind weit verbreitet. SSL sichert die Verbindung zwischen Client und Server im Internet über Verschlüsselung. Unterstützt werden verschiedene symmetrische Algorithmen. Die verwendeten Schlüssel werden über asymmetrische Verfahren ausgetauscht, SSL ist daher ein hybrides Verfahren (vgl. Kapitel 3.4).

Eine Datenübertragung mittels SSL erfolgt über zwei Teilprotokolle. Über das Handshake-Protokoll wird zu Beginn einer Verbindung zwischen Client und Server vereinbart, welche Algorithmen genutzt werden sollen. Hier findet auch die Authentifizierung der Kommunikationspartner statt. SSL unterstützt die Authentifizierung des Servers gegenüber dem Client und ab Version 3 auch die Authentifizierung des Client gegenüber dem Server. Über das zweite Protokoll (Record Layer Protocol) wird der Aufbau der zu übermittelnden Daten beschrieben.

Eine SSL-Verbindung erfolgt in mehreren Schritten. Nach der Vereinbarung über die zu verwendenden Algorithmen werden Zertifikate zwischen Client und Server ausgetauscht. In der Regel sendet nur der Server ein Zertifikat, welches ihn eindeutig identifiziert. Der Browser prüft die im Zertifikat enthaltenen Informationen auf Übereinstimmung zwischen DNS-Namen und *public key* des Servers. Dann generiert der Client einen nur für diese Sitzung gültigen Schlüssel (*session key*), der mit dem *public key* des Servers verschlüsselt an diesen übermittelt wird. Die weitere Kommunikation wird mit dem vereinbarten Algorithmus und Schlüssel gesichert.

Da zu jeder Sitzung ein neuer *session key* verwendet wird, kann ein Angreifer zunächst nur eine Datenübertragung entschlüsseln. In der Vergangenheit wurden Angriffe gegen SSL bekannt, die auf unsicheren Zufallszahlen beruhten. Ein Angreifer kann in diesem Fall die Tatsache nutzen, daß der zu durchsuchende Schlüsselraum kleiner ist als ursprünglich vorgesehen.

Weiterhin stand zur Nutzung außerhalb der USA lange Zeit nur eine Version von SSL zur Verfügung, die für die symmetrische Verschlüsselung eine Schlüssellänge von 40 statt 128 Bit verwendete.

9.3.2 S-HTTP

Das von der Firma Terisa Systems entwickelte Secure HTTP (S-HTTP) verfolgt die selben Ziele wie die oben beschriebenen Secure Sockets Layer. Auch hier wird die Übertragung zwischen Client und Server mittels HTTP auf Anwendungsebene gesichert. Unterschiede zwischen beiden Verfahren liegen in den zusätzlichen Konfigurationsmöglichkeiten von S-HTTP.

Im Gegensatz zu SSL kann bei S-HTTP die Auswahl von bestimmten Algorithmen während des Handshake zwischen Client und Server erzwungen werden. Darüber hinaus kann mittels S-HTTP der Einsatz einer digitalen Signatur vorgeschrieben werden. Ebenso ist aber auch eine Übertragung ohne den Einsatz von Schutzmechanismen möglich.

Der entscheidende Vorteil von S-HTTP besteht daher in der besseren Anpassung an die Sicherheitsanforderungen während der Kommunikation. Wie bei SSL ist der Einsatz von S-HTTP für höhere Protokollschichten transparent, die Änderungen an vorhandenen Anwendungsprogrammen sind minimal.

9.3.3 SSH

Die *Secure Shell* (SSH) ist eine Universitätsentwicklung, die inzwischen kommerziell von der Firma Data Fellows vertrieben wird. Mit SSH werden auf Anwendungsebene Verbindungen verschlüsselt, z.B. bei Anmeldung an einer fremden Maschine zu Beginn einer Telnet-Sitzung.

SSH sichert u.a. die Dienste Telnet, FTP und RCP durch Einsatz von Verschlüsselung. Eingesetzt werden DES, Triple-DES und IDEA, die verwendeten *session keys* werden mittels RSA oder Diffie-Hellman ausgetauscht, d.h. SSH ist ein hybrides Verfahren. Durch die Verwendung asymmetrischer Algorithmen ist ein Authentifizierung der Kommunikationspartner möglich, es existiert aber zur Zeit noch keine Infrastruktur zur Verwaltung und Zertifizierung öffentlicher Schlüssel.

Um SSH anwenden zu können, müssen sowohl auf dem Client als auch auf dem Server die entsprechenden Dienste installiert sein. Bei Verbindungen über das Internet ist dies jedoch selten der Fall. Zur Absicherung im lokalen

Netz kann SSH sinnvoll eingesetzt werden, um Sniffing-Angriffe zu verhindern.

9.3.4 Kerberos

Das ursprünglich vom MIT entwickelte System *Kerberos* dient zur Authentifizierung von Benutzern im Netz. Meldet sich ein Benutzer an seiner Maschine an, so erhält er von einem *Key Distribution Center* (KDC) im Netz einen für diese Sitzung gültigen *session key*, der Zugriff auf Ressourcen im Netz ermöglicht.

Kerberos setzt eine Umgebung voraus, in der sich ein Benutzer über ein Paßwort authentifiziert. Das KDC vereinbart mit dem angemeldeten Benutzer einen *master key*, in dessen Berechnung das Anmeldepaßwort eingeht. Versuchen zwei Partner zu kommunizieren, stellt das KDC einen *session key* bereit, der mit dem zugehörigen *master key* verschlüsselt den Kommunikationspartnern zugestellt wird. Die Gesamtheit von *session key* und Verwaltungsinformationen wird als *ticket* bezeichnet und dient zur gegenseitigen Authentifizierung der Partner und zur Verschlüsselung der übertragenen Daten.

Neuere Versionen von Kerberos erweitern den Authentifizierungsprozeß durch Einsatz eines *Ticket-Granting-Server* (TGS), um die Kommunikation mit verschiedenen Partnern zu erleichtern. Durch den TGS meldet sich ein Benutzer nur einmal an (*Single Sign On*) und kann dann auf alle im Netz bereitgestellten Ressourcen zugreifen, auch wenn diese über verschiedene Server verteilt sind.

9.3.5 PGP und PEM

Zur Verschlüsselung von E-Mails hat sich *Pretty Good Privacy* (PGP) von Phil Zimmermann zum Standard entwickelt. PGP wendet ein hybrides Verfahren (vgl. Kapitel 3.4) an. Zunächst erfolgt eine symmetrische Verschlüsselung mittels der Algorithmen DES, Triple-DES, IDEA oder CAST, dann wird der hier verwendete *session key* nach RSA oder Diffie-Hellman verschlüsselt. Aktuelle Versionen erlauben nur noch den Einsatz von Diffie-Hellman, da vermehrt Sicherheitslücken in RSA bekannt wurden (vgl. [Schneier 1996, S.538ff]).

Aufgrund von Exportbeschränkungen der USA für starke Kryptographie existieren zwei Versionen von PGP, eine internationale Version und eine Version für den exklusiven Einsatz in den USA. Beide Versionen unterscheiden sich in ihrer Funktion jedoch nicht voneinander. Für alle gängigen E-Mail-Clients gibt es einen Plug-In, der die Anwendung vereinfacht. PGP unterstützt neben der Verschlüsselung von E-Mails auch die Dateiverschlüsselung.

Die Zertifizierung öffentlicher Schlüssel ist in PGP nicht durch Einsatz einer zentralen Zertifizierungsstelle gelöst, sondern durch ein sogenanntes *Web of Trust*. Bei diesem Modell werden öffentliche Schlüssel durch eine digitale Unterschrift von einem oder mehreren Anwendern zertifiziert. Diese haben sich vorher davon überzeugt, daß ein Schlüssel tatsächlich zu einer bestimmten Person gehört, z.B. durch Vergleich des MD5-Hash eines *public key* mit dem von der Person genannten Wert.

Das Modell des Web of Trust führt jedoch bei einer großen Zahl von Kommunikationspartnern zu einer unüberschaubaren Menge von Zertifikaten. Um die Kommunikation mit neuen Partnern einfacher zu gestalten, wurden im Internet PGP-Keyserver eingerichtet, auf denen

öffentliche Schlüssel gespeichert werden können. Es erfolgt hier jedoch keine weitere Sicherheitsprüfung.

Dieses Problem umgeht der PEM-Standard (*Privacy Enhanced Mail*) durch eine hierarchische Anordnung von Zertifizierungsstellen. PEM unterscheidet sich von PGP durch die Einschränkung auf bestimmte Verschlüsselungsalgorithmen (z.B. DES) und die getrennte Übertragung des MD5-Hashwertes. PEM ist zur Zeit weit weniger verbreitet als PGP.

9.3.6 S-MIME

Mit der von der Firma RSA entwickelten *Secure Multipurpose Internet Mail Extension* (S-MIME) ist eine Sicherung des MIME-Standards (vgl. Kapitel 7.7) zur Übertragung von E-Mail möglich. Um dies zu erreichen, wird in den Standard ein hybrides Verschlüsselungsverfahren analog zu PGP integriert. E-Mails werden symmetrisch verschlüsselt (DES, Triple-DES u.a.), der verwendetet *session key* wird durch den RSA-Algorithmus geschützt.

S-MIME fügt sich nahtlos in den vorhandenen MIME-Standard ein. MIME definiert, welche Anwendung abhängig vom Inhalt der E-Mail auf dem empfangenden Rechner gestartet werden soll. Einzelne Tags (*MIME-Types*) spezifizieren die Anwendung. S-MIME fügt den vorhandenen einen neuen MIME-Type hinzu, der den Umgang mit Verschlüsselung und Signatur regelt.

Ein entscheidender Vorteil gegenüber PGP liegt in der Verwendung des X.509 Standards zum Einsatz von Zertifikaten, der Schwächen des *Web of Trust* umgeht. Dem gegenüber wirkt sich die Beschränkung auf den RSA-Algorithmus negativ aus.

9.3.7 Weitere Standards

Einige weitere Standards betreffen nicht unmittelbar die sichere Kommunikation, sondern definieren lediglich Sicherheitsmechanismen im Umfeld der Datenübertragung im Internet. Zu diesen erweiterten Standards gehört z.B. *Secure Electronic Transaction (SET)* und das *Homebanking Computer Interface* (HBCI). Da hier lediglich bekannte kryptographische Verfahren neu kombiniert und mit Anweisungen zur Durchführung von Transaktionen versehen werden, sind diese Verfahren nicht Teil der vorliegenden Arbeit. Eine Übersicht über gängige Verfahren im elektronischen Zahlungsverkehr liefert [Raepple 1998, Kap. 5.5].

9.3.8 Übersicht Einordnung OSI

Die meisten der bisher beschriebenen Schutzmechanismen können unmittelbar in das OSI-Modell eingeordnet werden. Abbildung 9.2 zeigt eine Zuordnung der verschiedenen Verfahren zu einzelnen Schichten des Modells.

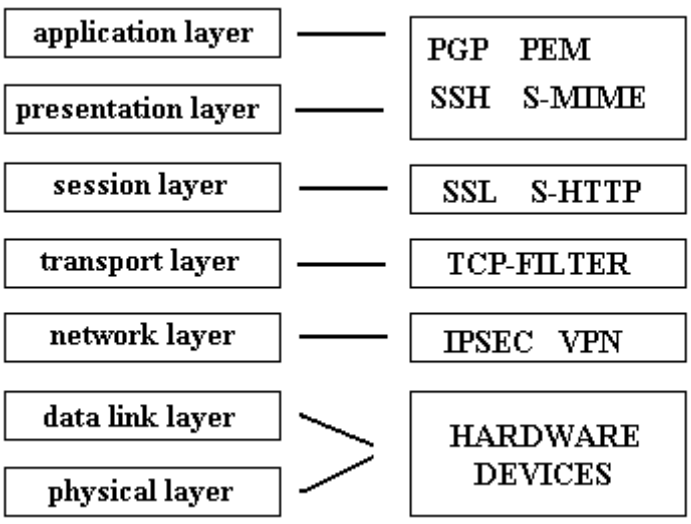


Abbildung 9.2: Zuordnung der Sicherheitsmechanismen zum OSI-Modell

Diese Mechanismen sichern die Übertragung von Daten zwischen Rechnern, bieten jedoch keinen Schutz gegen einen direkten Angriff auf einen einzelnen an der Kommunikation beteiligten Rechner. Mit dieser Problematik beschäftigt sich der folgende Abschnitt.

10 Schutzmaßnahmen für Betriebssysteme

10.1 Grundlagen

Die vorangegangenen Abschnitte haben gezeigt, daß Kryptographie eine starke Waffe im Kampf um die sichere Kommunikation ist. Doch auch die beste Verschlüsselung bleibt nutzlos, wenn der Rechner, auf dem sie durchgeführt wird, manipuliert worden ist. Der einzelne Arbeitsplatzrechner ist nicht nur die Schnittstelle zum Benutzer, sondern auch der Endpunkt eines Kommunikationskanals im Internet.

Benutzer in Netzwerken werden zunächst über den Rechner identifiziert, an dem sie sich anmelden. Weitere Rechte im Netz, z.B. der Zugriff auf freigegebene Ressourcen, werden oftmals aufgrund der Vertrauensstellung zwischen Rechnern gewährt, d.h. ein auf einer Maschine angemeldeter Benutzer erhält auch Zugriff auf die Ressourcen anderer Rechner. Sichere Kommunikation in einem Netzwerk beginnt daher immer mit der Zugriffskontrolle auf den lokalen Rechner. Dieser Rechner ermöglicht den Zugriff auf das Netz. Wer die Ressourcen eines Rechners kontrolliert, hat ein Sprungbrett zum lokalen Netz und zum Internet gefunden.

Unter Zugriffskontrolle wird die Kontrolle des Zugriffs auf alle Informationen eines Computersystems im weiteren Sinne verstanden, d.h. Zugriff auf Datenbestände, Prozesse, Kommunikationskanäle etc. (vgl. [Pohl/Weck 1993,S. 22ff]). Zugriffskontrolle wird in aktuellen Systemen durch Paßwörter, Magnet-/Chipkarten und biometrische Verfahren ausgeübt.

Sichere Systeme sind nicht allein durch Zugriffskontrolle und kryptographische Verfahren zu realisieren, sie erfordern auch Maßnahmen, die außerhalb des Bereichs der technischen Sicherheit liegen. Dazu zählt die

physische Zugangskontrolle oder materielle Sicherheit im weiteren Sinne sowie die personelle Sicherheit im Unternehmen. Diese Gebiete werden im Rahmen der Arbeit nicht behandelt, einen guten Überblick liefern hier [Kersten 1991] und [Pohl/Weck 1993].

Der Zugriffskontrolle kommt in vernetzten Systemen eine besondere Bedeutung zu, daher werden die verschiedenen Mechanismen in den folgenden Abschnitten kurz vorgestellt.

Hier ist zu beachten, daß zwischen der Behauptung einer Identität (*Identifikation*) und dem Nachweis derselben (*Authentifikation*) unterschieden werden muß. Den Nachweis seiner Identität erbringt ein Benutzer zumeist durch Paßwörter, Chipkarten oder biometrische Verfahren.

10.2 Paßwörter

Die Realisierung der Zugriffskontrolle über ein Paßwort ist weit verbreitet. Allerdings bietet dieses Verfahren viele Angriffspunkte, da es auf der Geheimhaltung einer Zeichenfolge durch den Benutzer beruht. Die Zeichenfolge wird bei Einrichtung des Benutzeraccount vereinbart und vom System verschlüsselt abgelegt. Bei jeder Anmeldung wird das eingegebene Paßwort ebenfalls verschlüsselt und mit der gespeicherten Version verglichen. Stimmen beide Versionen überein, erhält der Benutzer Zugriff auf das System. In Abschnitt 3.10 wurde bereits erläutert, daß die gespeicherte Version des Paßworts Angriffen ausgesetzt ist, die durch systematische Verschlüsselung ausgewählter Zeichenfolgen eine Übereinstimmung zu finden versuchen.

Weiterhin werden Paßwörter auch zur Authentifikation des Benutzers gegenüber Netzdiensten eingesetzt und in der Regel im Klartext übertragen,

so daß ein Angreifer durch einen Sniffing-Angriff (vgl. Kapitel 6.1) gezielt Paßwörter ausspähen kann. Hier können sowohl Werkzeuge zur Netzdiagnose eingesetzt werden, die einzelne Pakete aus dem Datenstrom zwischen zwei Maschinen ausfiltern, als auch spezielle Hacker-Tools. Abbildung 10.1 stellt dar, wie über das Tool *sniffit* das Paßwort einer FTP-Sitzung ermittelt wird. Der Angreifer spezifiziert durch Angabe der IP-Adresse von Sender und Empfänger sowie den für bestimmte Dienste charakteristischen TCP/IP-Port, welche Datenpakete ausgefiltert werden sollen. Im Beispiel werden alle Datenpakete ausgefiltert, die an den Rechner mit der IP-Adresse 192.168.100.10 gerichtet sind und Port 21 verwenden, der dem Dienst FTP zugeordnet ist. Sniffit ermittelt daraufhin automatisch das verwendete Paßwort und stellt es dar (im Beispiel die Zeichenfolge *test*). Von diesem Angriff sind alle Netzdienste betroffen, die Klartext übertragen, darunter Telnet, FTP, HTTP, und POP3.

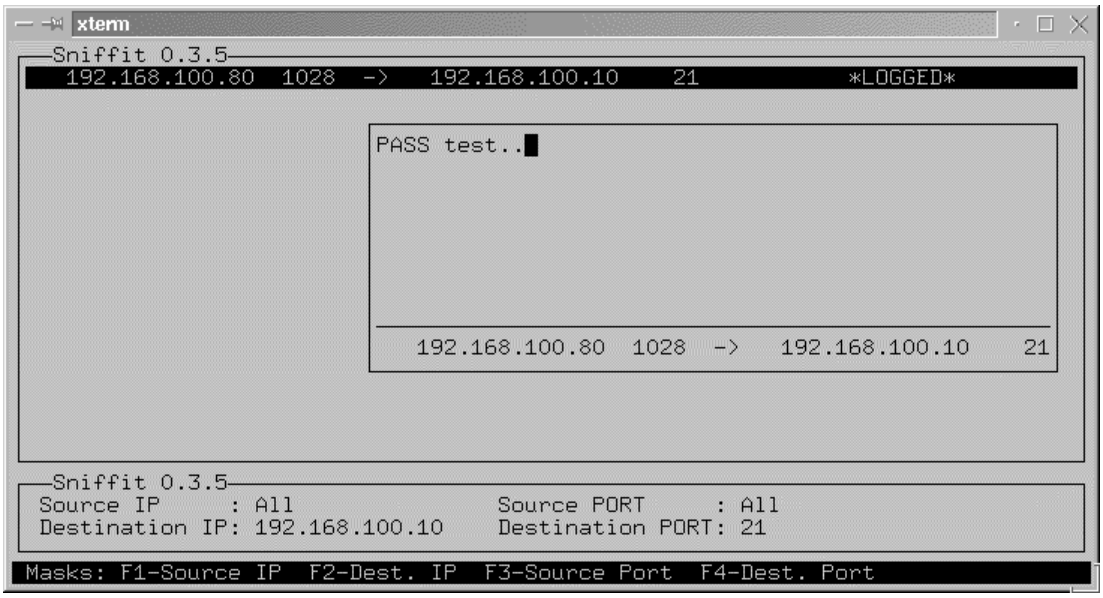


Abbildung 10.1: Das Tool SNIFFIT

Auch der lokale Rechner bietet Angriffspunkte. Hintergrundprozesse, die alle Tastatureingaben eines Benutzers protokollieren, können eingesetzt werden, um die Eingabe eines Paßworts abzufangen. Darüber hinaus findet die

Verarbeitung von eingegebenen Paßwörtern im Hauptspeicher des Rechners statt, der bei ungenügendem Schutz einzelner Speicherbereiche (z.B. beim kooperativen Multitasking von Windows 3.11 oder bei der gemeinsam Nutzung einer VM unter Windows 95) ausgelesen werden kann.

Problematisch ist auch die vom Benutzer unbemerkte Speicherung des Paßworts durch das System, z.B. in der Auslagerungsdatei (Swapspace) oder in Konfigurationsdateien für Online-Dienste (z.B. T-Online). Der Remote Access Dienst von Windows NT speichert beispielsweise Paßwörter auch dann, wenn der Benutzer dies explizit nicht erlaubt (vgl. Abbildung 10.2).

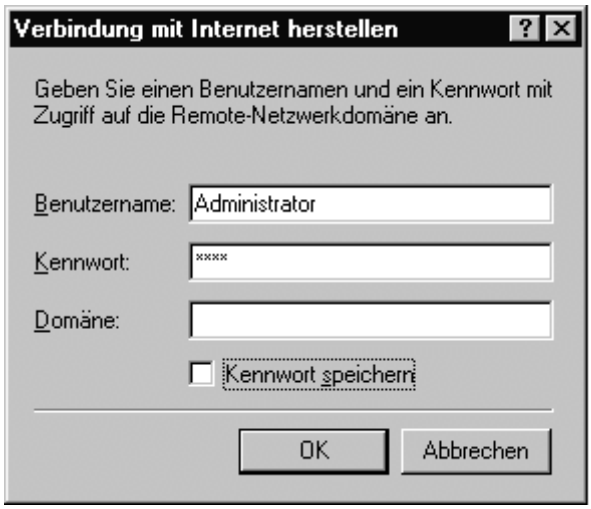


Abbildung 10.2: Speicherung auch ohne Angabe der Option *Kennwort speichern*

Dateien, die Paßwörter enthalten, können gezielt durchsucht werden. Dies gilt auch für Dateien, die nicht auf dem Rechner des Benutzers gespeichert sind. Die Backup-Bänder eines UNIX-Systems beinhalten ebenfalls die Paßwortdateien */etc/passwd* oder */etc/shadow* und können, da die Sicherheitsmechanismen des Systems sich nicht auf externe Speichermedien erstrecken, ohne Schwierigkeiten von diesen Medien eingelesen werden. Viele Systeme, wie z.B. Windows NT, erlauben die Erstellung einer Emergency Recovery Disk, auf der die Systemkonfiguration

gespeichert ist. Damit hat der Benutzer die Möglichkeit, bei schwerwiegenden Hard- und Softwarefehlern das System wieder herzustellen. Auch hier sind die vollständigen Paßwortdateien des Systems enthalten.

Hat der Angreifer physischen Zugriff auf die Hardware eines Systems, sind weitere Angriffe möglich. Zu diesen Angriffen zählt der Neustart des Systems im Single-User-Mode (z.B. Solaris), bei dem nach dem Start eine Shell mit Superuser-Rechten zur Verfügung steht, oder das Booten über eigene Datenträger (z.B. Startdisketten oder externe angeschlossene SCSI-Festplatten), die später einen Zugriff auf das Dateisystem des fremden Rechners und damit auch auf sensible Daten erlauben.

Schutzmechanismen für Paßwörter sind vielfältig und beginnen bereits bei der Auswahl der verwendeten Zeichenfolge. Bei der Generierung des Paßworts kann bereits geprüft werden, ob dieses Paßwort genügend lang ist und ob es eine Kombination aus alphanumerischen Zeichen und Sonderzeichen enthält, die nicht aus Wörterbucheinträgen oder Benutzernamen konstruiert ist. Diese Maßnahmen erschweren einen Angriff über systematisches Austesten.

Weiteren Schutz bildet das *password aging*, bei dem der Benutzer in regelmäßigen Abständen aufgefordert wird, das Paßwort zu ändern. Zusammen mit einer Sperrung des Accounts bei mehrmaliger falscher Paßworteingabe kann eine deutliche Erhöhung des Sicherheitsniveaus erreicht werden.

Die sichere Speicherung von Paßwörtern kann auf Chipkarten erfolgen (s.u.) oder durch Verschlüsselung von Festplatte und externen Speichermedien. Nur wenige Betriebssysteme bieten eine starke Verschlüsselung, jedoch existieren zahlreiche Zusatzprodukte, die diese Funktionen bereitstellen (z.B.

Utimaco Safeguard, Norman Access Control, Aliroo PrivaSuite u.a.). Unter Windows NT ab Service Pack 3 besteht die Möglichkeit, die Paßwort-Datenbank zu verschlüsseln und beim Start des Systems wieder freizugeben. Bei Auslagerung auf externe Medien bleibt die Verschlüsselung erhalten (vgl. Abbildung 10.3).

Abbildung 10.3: Verschlüsselung sensibler Informationen unter Windows NT

Paßwörter sind auch Angriffen bei der Übertragung im lokalen Netz und im Internet ausgesetzt. Hier ist die Verwendung der in Abschnitt 9.3 angeführten Sicherheitsmaßnahmen sinnvoll oder der Verzicht auf die Nutzung bestimmter Netzdienste wie Telnet oder (nicht anonymes) FTP, die Paßwörter im Klartext über eine Internet-Verbindung übertragen.

Eine deutlich höhere Sicherheitsstufe der Zugriffskontrolle wird durch den Einsatz von Chipkarten erreicht, wie der nächste Abschnitt zeigt.

10.3 Chipkarten

10.3.1 Grundlagen

Mit dem Aufkommen der ersten Kreditkarten (z.B. Diners Club) war die Idee geboren worden, Informationen auf Plastikkarten aufzubringen. Auf diesen ersten Kreditkarten wurden der Name des Besitzers, Kreditkartennummer und Ablaufdatum aufgebracht. Die Karten wurden durch schwer reproduzierbare Merkmale, z.B. Hochprägung oder Hologramme, vor unerlaubter Vervielfältigung geschützt.

Wenig später folgten die ersten Magnetstreifenkarten, auf denen Informationen elektronisch lesbar kodiert waren. Die auf dem Magnetstreifen aufgebrachte Information war jedoch einfach auslesbar und mit etwas größerem Aufwand (Kartenlesegerät mit motorischem Einzug) auch veränderbar. Darüber hinaus war die Speicherkapazität von Magnetstreifenkarten sehr begrenzt (bis max. 256 Byte).

Die Nachteile anderer Karten konnten von Chipkarten größtenteils überwunden werden. Bei einer Chipkarte wird ein Halbleiterchip mit dem Kartenkörper vergossen. Die auf dem Chip enthaltenen Informationen können mittels eines Kartenterminals ausgelesen werden. Einfache Speicherchipkarten erlauben die ungeschützte Veränderung der aufgebrachten Informationen (z.B. die deutsche Krankenversichertenkarte), bei Speicherchipkarten mit Sicherheitslogik kann die Veränderung unterbunden oder an bestimmte Bedingungen (z.B. Eingabe einer PIN) geknüpft werden.

Prozessorchipkarten enthalten im Gegensatz zu Speicherchipkarten nicht nur einen statischen oder dynamischen Speicher, sondern eine vollständige

Recheneinheit mit Mikroprozessor, ROM, EPROM oder EEPROM und RAM. Im ROM befindet sich das Betriebssystem des Chips, das EPROM oder EEPROM dient als nichtflüchtiger Speicher für Programmcode und Daten, der Programmcode wird im RAM ausgeführt.

Die Formate und physikalischen Eigenschaften von Karte und Chip sind in der ISO-Norm 7816 beschrieben.

10.3.2 Mikroprozessorkarten

Mikroprozessorkarten sind im Hinblick auf die Informationssicherheit besonders interessant. Dieser Kartentyp kann gespeicherte Daten wirksam gegen unerlaubte Kenntnisnahme und Veränderung schützen. Für einzelne Speicherbereiche kann festgelegt werden, welche Art des Zugriff erlaubt ist. Einige Bereiche sind z.B. nur lesbar, andere unter bestimmten Bedingungen (Eingabe einer PIN) auch schreibbar. Der Zugriff wird über das Betriebssystem der Chipkarte gesteuert, die Datenein- und ausgabe erfolgt nur über eine genau definierte serielle Schnittstelle. So ist es auch möglich, Daten auf die Karte zu laden, die nur vom Betriebssystem der Chipkarte, nicht jedoch vom Anwender lesbar sind. Vertrauliche Daten dieser Art können z.B. Signaturschlüssel und Schlüssel zur eindeutigen Identifikation der Karte sein. Über eine Chipkarte ist dann eine digitale Signatur durchführbar, ohne daß der Anwender direkt den Signaturschlüssel einsehen oder verändern kann; der Karte werden die zu signierenden Daten übergeben, der Signaturvorgang wird auf der Karte durchgeführt und die entstandene Signatur wird zurückgeliefert (vgl. [Rankl/Effing 1998] Kapitel 8).

Eine Mikroprozessor-Chipkarte stellt ein in sich geschlossenes System dar, einen eigenständigen Rechner mit integrierten Sicherheitsfunktionen. Diese

Eigenschaft hat entscheidend zur Verbreitung der Chipkarten als einem vergleichsweise preiswerten Sicherheitswerkzeug geführt.

10.3.3 Sicherheitsaspekte

Das System Chipkarte ist wie jedes andere technische System angreifbar. Angriffe auf der physikalischen Ebene sind in der Regel sehr aufwendig, da teueres Spezialwerkzeug benötigt wird, um die auf dem Halbleiter gespeicherten Informationen auszuwerten. Dies könnte z.B. durch Feinabschleifen des Halbleiters und Auswertung der Schalterlogik mittels Mikroskop und Prozeßrechner geschehen.

Angriffe auf logischer Ebene sind hingegen mit geringerem finanziellen Aufwand durchführbar. Bei dieser Art des Angriffs wird häufig die Kommunikation zwischen Chipkarte und Chipkartenterminal protokolliert, indem mittels eines Chipkartenadapters der Datenstrom zwischen Karte und Terminal belauscht wird. Der Angriff kann erweitert werden, indem eine Chipkarte simuliert wird, d.h. eine Schaltung mit Mikroprozessor wird so aufgebaut, daß sie sich gegenüber einem Chipkarten-Terminal wie eine echte Chipkarte verhält. So kann die Kommunikation zwischen Terminal und Chipkarte nicht nur protokolliert, sondern auch gezielt durch Einstreuen falscher Datenpakete manipuliert werden.

Den besten Schutz vor Angriffen auf logischer Ebene bietet eine Verschlüsselung des Datenstromes zwischen Chipkarte und Terminal. Hierbei verfügen Karte und Terminal in der Regel über die Möglichkeit, eine symmetrische Verschlüsselung durchzuführen. Neuere Chipkarten verfügen oftmals auch über einen kryptographischen Koprozessor und sind daher in der Lage, asymmetrische Verschlüsselung anzuwenden (z.B. RSA).

Verschlüsselungsverfahren bilden die Grundlage für eine Authentisierung von Chipkarte und Terminal. Mittels Authentisierung kann der Terminal die Echtheit der Chipkarte und die Chipkarte die Echtheit des Terminals prüfen, um Manipulationen an einer der beiden Komponenten auszuschließen. Grundlage der Authentisierung ist die Prüfung einer geheimen Bitfolge, die sowohl dem Terminal als auch der Chipkarte bekannt ist. Bei der Prüfung einer Chipkarte durch das Terminal (*Internal Authenticate*) erhält die Chipkarte eine Zufallszahl, verschlüsselt diese Zahl unter Verwendung der geheimen Bitfolge und sendet das Ergebnis zum Terminal. Der Terminal entschlüsselt mit der gespeicherten Bitfolge und prüft das Ergebnis auf Übereinstimmung. Im Falle der Übereinstimmung ist die Chipkarte authentisch. Die Prüfung des Terminals durch die Chipkarte (*External Authenticate*) erfolgt nach demselben Prinzip. Durch Verwendung kartenindividueller Schlüssel und asymmetrischer Verfahren kann die Authentisierung weiter abgesichert werden. Zur Vereinfachung des Authentisierungsverfahrens kann die Echtheit von Karte und Terminal in einem einzigen Vorgang (*Mutual Authenticate)* geprüft werden (vgl. [Rankl/Effing 1996, S.237ff]).

Eine weitere Absicherung des Systems Chipkarte kann über Fehlbedienungszähler (PIN-Eingabe und Authentisierung), kryptographisch sichere Zufallszahlen-Generatoren, physikalische Sicherungen (z.B. Passivierungsschichten) und Sicherung des Kartenkörpers vor unerlaubter Vervielfältigung durch Aufbringen von fälschungssicheren Merkmalen (z.B. Hologrammen) erfolgen.

10.4 Biometrische Verfahren

Biometrische Verfahren verifizieren die Identität einer Person durch eindeutige Merkmale, die nur dieser Person zugeordnet werden. Dies können physiologische Merkmale sein, z.B. das charakteristische Muster eines Fingerabdruckes, oder Merkmale, die das Verhalten der Person erfassen, z.B. den Schreibrhythmus durch Messung der Zeitintervalle zwischen mehreren Tastaturanschlägen.

Während bisherige Verfahren auf den Prinzipien Wissen (Paßwort) oder Besitz (Chipkarte) beruhten, liegt der entscheidende Vorteil biometrischer Verfahren darin begründet, daß sie eine Person direkt identifizieren können. Wenn der Besitz einer Karte oder die Kenntnis eines Paßworts entfällt, ist dies aus der Perspektive der Sicherheit ein Vorteil, da die in den vorangegangenen Abschnitten genannten Probleme nicht auftreten.

Zu den wichtigsten momentan verfügbaren Verfahren gehören die verschiedenen Mechanismen der Stimmerkennung, der Netzhaut-Scan, die Messung von Gesichtsfeld und Handgeometrie, die Fingerabdruckmessung und die Messung des Tastaturanschlages oder der Dynamik einer Unterschrift. Viele weitere Verfahren sind in der Entwicklung, z.B. die Erkennung von Wärmefeldern im menschlichen Körper, jedoch noch nicht für den praktischen Einsatz geeignet.

Problematisch sind die verglichen mit herkömmlichen Verfahren höheren Kosten einer biometrischen Zugriffskontrolle. Hinzu kommt die geringere Akzeptanz der Benutzer und eine relativ hohe Fehlerrate bei vielen Systemen, die im Verfahren selbst begründet liegt. Während beim Einsatz von Paßwörtern und Chipkarten eine einfache Entscheidung getroffen werden kann, ob die Kriterien der Identifikation erfüllt sind, arbeiten

biometrische Verfahren mit Toleranzen. Zunächst wird ein physiologisches oder ein verhaltensbasierendes Merkmal gemessen und ausgewertet. Dann werden bei jeder Identifikation die gespeicherten Werte mit den ursprünglichen Referenzwerten verglichen. Durch die komplizierten Meßverfahren und die möglichen Abweichungen durch körperliche Veränderungen, z.B. das Muster der Stimme nach einer Erkältung, kann nur verglichen werden, ob die aktuell gemessenen Werte innerhalb einer definierten Toleranz zu den Referenzwerten liegen. Ein direkter Vergleich ist nicht möglich.

Die vorgestellten Verfahren Paßwort, Chipkarte und Biometrie sind Mechanismen der Zugriffskontrolle, über die der Zugriff auf lokale Rechner gesichert werden kann. Weitere Maßnahmen, die den lokale Rechner schützen, werden im nächsten Abschnitt erläutert.

10.5 Schutzmaßnahmen für lokale Rechner

Neben der Zugriffskontrolle erlauben weitere Maßnahmen die direkte Sicherung eines Systems gegen Angriffe. Hier sind insbesondere Datenverschlüsselung, Kontrolle der Schnittstellen und Protokollführung zu nennen.

Über die vollständige oder teilweise Verschlüsselung der Festplatte wird die Vertraulichkeit der gespeicherten Informationen gewahrt. Kryptographische Verfahren werden auf die gesamte Festplatte, auf einzelne Partitionen oder auf Ebene von Verzeichnissen und Dateien angewandt, um sensible Daten (z.B. Paßwort-Dateien) abzusichern. Die zusätzliche Verschlüsselung von Disketten und Backup-Medien schützt diese Informationen auch außerhalb des Systems.

Die Kontrolle der Schnittstellen unterbindet den unkontrollierten ein- und ausgehenden Datentransfer. Das Einschleusen von Viren und trojanischen Pferden wird ebenso unterbunden wie der Export sicherheitskritischer Informationen. Über serielle und parallele Schnittstellen können ebenso wie über Disketten und Wechselmedien Informationen transportiert werden. Problematisch sind auch Durchführungen des Datenbusses (z.B. SCSI) aus dem Gehäuse eines Rechners, da hier externe Geräte mit großen Speicherkapazitäten angeschlossen werden können und bei entsprechender Konfiguration auch die Möglichkeit besteht, das System von einer externen Festplatte zu starten und auf die Daten der lokalen Platte zuzugreifen.

Da, wo Angriffe auf Systeme nicht vermieden werden können, kann durch das Führen eines Sicherheitsprotokolls (auditing) der Zugriff aufgezeichnet werden. Protokolle enthalten neben Systeminformationen (z.B. Zeitpunkt eines Neustarts) auch Benutzeraktionen (z.B. An- und Abmeldung oder Zugriff auf Dateien und Geräte). Erweiterte Prüfverfahren testen die Integrität durch Erstellung von Prüfsummen über einzelne Systemdateien, so daß Modifikationen (z.B. Viren und trojanische Pferde) bemerkt werden. Die Protokolldateien selbst müssen auch gegen Modifikation geschützt werden. Abbildung 10.4 zeigt den Schutz für einzelne Dateien durch die Software *Safeguard Advanced Security* der Firma Utimaco. Hier kann auch spezifiziert werden, welche Aktion nach der Feststellung einer Veränderung ausgeführt werden soll, z.B. eine vollständige Blockierung des Systems.

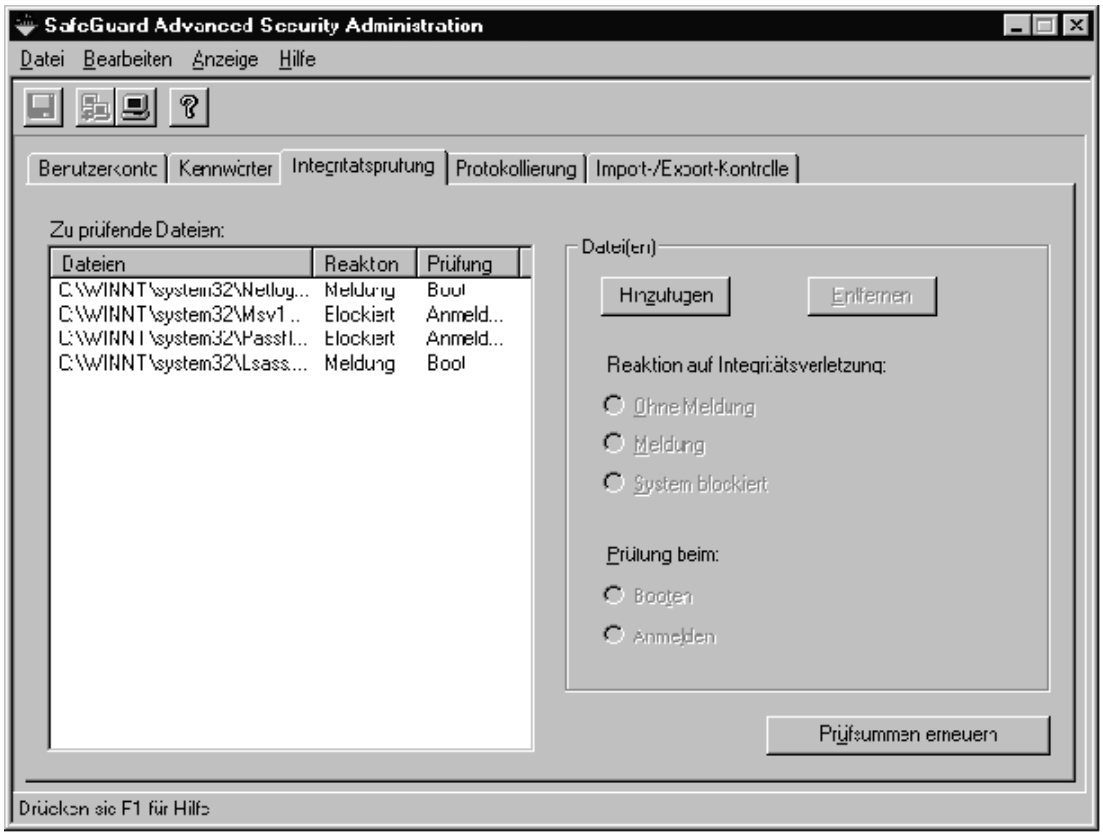


Abbildung 10.4: Integritätsprüfung durch Safeguard

Weitere Sicherheitsmechanismen des Betriebssystems erhöhen die Resistenz gegen Angriffe. Dies kann z.B. eine abgesicherte Umgebung (*Trusted Path*) zur Anmeldung eines Benutzers sein, die Veränderungen an Paßwort-Eingabemasken verhindert, oder die *Object Reuse Protection*, die Objekte (Speicherblöcke, Dateien etc.) vor dem Zugriff durch einen Benutzerprozeß initialisiert, um die zuvor von anderen Benutzern eingebrachte Informationen zu schützen.

Die Unterstützung verschiedener Rollen für administrative Funktionen (*Trusted Facility Management*) bewirkt, daß Privilegien z.B. an einzelne Arbeitsgruppen-Administratoren, Verwalter von Druckern oder Backup-Operatoren vergeben werden, ohne daß ein voller Systemzugriff erlaubt wird. Privilegien sind abgestufte Benutzerrechte zur Durchführung systemrelevanter Aktionen, z.B. den Zugriff auf Gerätetreiber oder Rechte

zum Shutdown des Systems. Es sollten stets nur soviele Rechte vergeben werden, wie zur Durchführung der Aufgabe benötigt werden.

In modernen Betriebssystemen, z.B. Windows NT werden Ressourcen und Zugriffsrechte eindeutig über sog. *Security Identifier* (SID) identifiziert. An Benutzer und Systemprozesse werden *Access Token* vergeben, die vollständige Beschreibungen inklusive SID und Privilegien enthalten. Sie werden vor dem Zugriff auf ein Objekt mit den Informationen verglichen, welche die Zugriffsrechte auf ein Objekt regeln (*Security Descriptor*).

Formale Vorgaben und Einstufungen für die Sicherheit von lokalen Rechnern finden sich in verschiedenen Katalogen zur Bewertung der Sicherheit informationstechnischer Systeme (*Orange Book/TCSEC, ITSEC, Common Criteria*). Dort werden funktionale Anforderungen in Klassen eingeteilt, in die vorhandene Systeme eingeordnet werden können. In der Klasse C2 von ITSEC wird z.B. die oben erwähnte Object Reuse Protection sowie individuell steuerbare Kontrolle und Protokollierung gefordert.

Auch wenn formal eine bestimmte Sicherheitsstufe erreicht wird, ist die Sicherheit des Netzbetriebes noch nicht gewährleistet. Windows NT ist z.B. unter bestimmten Bedingungen nach ITSEC C2 zertifiziert, diese Zertifizierung gilt jedoch nach der Einrichtung von Netzverbindungen nicht mehr. Für den Einsatz im Internet sind die in Kapitel 9 beschriebenen zusätzlichen Maßnahmen notwendig.

Nachdem nun allgemeine Schutzmaßnahmen für die Internet-Kommunikation und für lokale Rechner dargestellt wurden, wird im nächsten Abschnitt kurz der Stand der Technik zu Online-Datenbanken skizziert, um das neu entstandene Modell für die Bonner Wortdatenbank in einen größeren Zusammenhang einzuordnen.

11 Online-Datenbanken im Internet

Ausgewählt wurden *WordNet*, *COSMAS*, *WordbanksOnline* und *BLISNet*, die bisherige Abfragekomponente zur Bonner Wortdatenbank. Die Auswahl der Systeme orientiert sich hierbei nicht an linguistischen Fragestellungen, sondern verfolgt das Ziel, die gängigen Sicherheitsmechanismen der Datenbanken abgestuft darstellen zu können. Da eine vollständige Beschreibung der Systeme aus linguistischer Sicht über den Rahmen dieser Arbeit hinausgehen würde, erfolgt nur eine kurze Charakterisierung, an die sich eine Diskussion der eingesetzten Sicherheitsmechanismen und ihrer Wirksamkeit anschließt.

11.1 WordNet

Wordnet ist ein lexikalisches Referenzsystem, das an der Universität von Princeton entwickelt wurde und Wortformen (Nomina, Verben, Adjektive und Adverbien) der englischen Sprache explizit in ihren Relationen zueinander beschreibt. In der Datenbank sind die verschiedenen Bedeutungen durch lexikalisch-semantische Relationen (z.B. Synonymie oder Hyponymie) miteinander zu einem semantischen Netz verknüpft

Die gesamte Datenbasis von *Wordnet* ist auf CD-ROM und zum Download im Internet verfügbar, zusätzlich ist eine Online-Abfrage möglich. Der Online-Zugriff ist über Frames realisiert und verwendet aktuell (Stand November 2001) die Datenbasis der *WordNet*-Version 1.6. (s. Abbildung 11.1). Die Nachfolge-Version 1.7 liegt derzeit nur zur lokalen Installation unter UNIX vor.

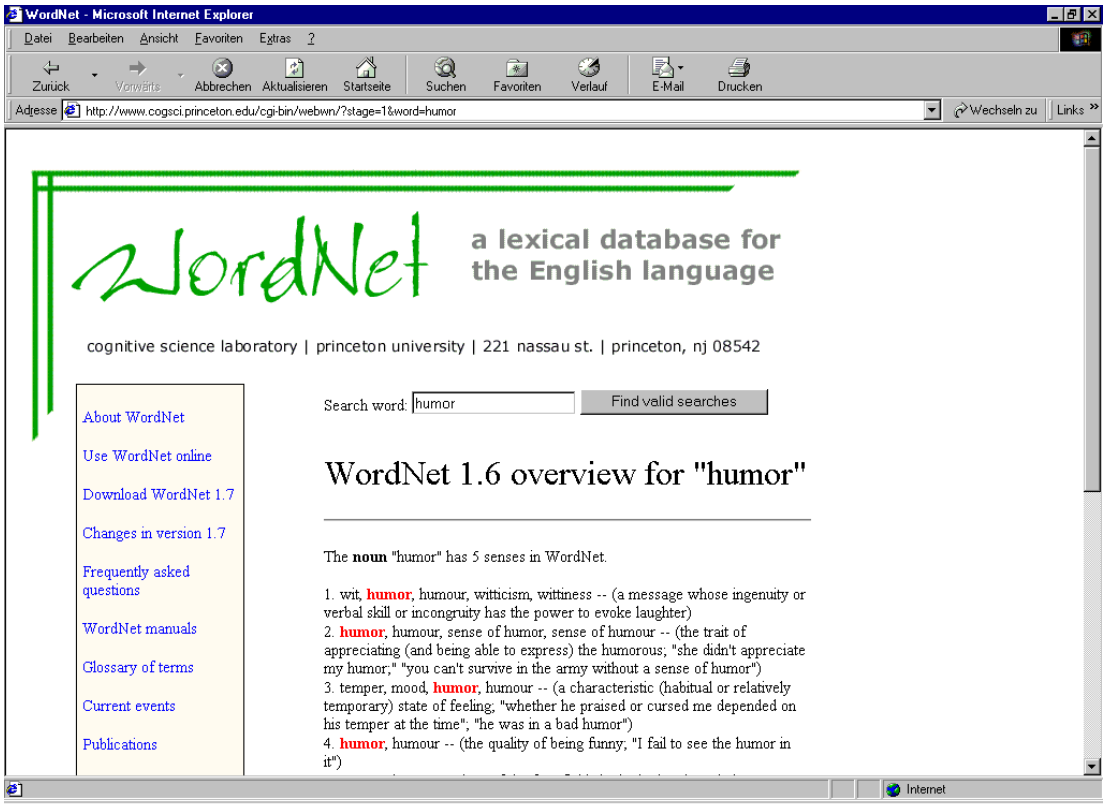


Abbildung 11.1: Einfaches WWW-Interface zu WordNet

Aufgrund des großen Erfolges von *WordNet* gibt es verschiedene Bestrebungen, das zugrundeliegende Konzept des lexikalisch-semantischen Netzes auf andere Sprachen zu übertragen. Der deutschsprachige Ansatz der Universität Tübingen, das *GermaNet* ist Teil dieser Bestrebungen. Hier sind über 40.000 Synsets zu deutschen Nomina, Verben und Adjektiven enthalten. (Stand September 2001). Im Projekt *EuroWordNet* wurden die verschiedenen Ansätze miteinander in einem Gesamtsystem mit Hilfe einer Interlingua verbunden. Aktuell sind hier neben Englisch und Deutsch die Sprachen Niederländisch, Italienisch Spanisch, Französisch, Estnisch und Tschechisch enthalten (Stand Oktober 2001).

In zahlreichen Projekten wurden weitere Oberflächen und Client/Server-Systeme zu *WordNet* entworfen, mehr als 20 dieser Projekte werden alleine auf den Webseiten der Princeton-Universität aufgelistet. Hier sind alle

gängigen Programmier- und Scriptsprachen (Java, C/C++, Prolog, LISP, TCL, Perl, Python etc.) vertreten.

Im WWW-Interface zu *WordNet* sind aktuell keine Sicherheitsmechanismen implementiert, der Transport der abgefragten Informationen erfolgt unverschlüsselt über das Internet. Ein schreibender Zugriff auf den Datenbestand ist nicht möglich. Dies gilt auch für die zusätzlichen Schnittstellen.

Eine Bewertung von *WordNet* scheint einfach - der Datenbestand ist frei verfügbar und die Kommunikation aus diesem Grund ohne Sicherung. Da kein schreibender Zugriff aus dem Internet auf die Datenbasis möglich ist, besteht für *WordNet* auf den ersten Blick auch keine Notwendigkeit, Sicherheitsmechanismen zu implementieren.

Auf den zweiten Blick stellt sich der Sachverhalt anders dar. Auch wenn der Betreiber auf Sicherheitsmechanismen verzichten kann, ist dieser Verzicht unter Umständen ein Risiko für den Anwender. Dies betrifft zum einen die fehlende Identifikationsmöglichkeit des Servers und zum anderen den Zugriff über eine der vielen zusätzlichen Abfragekomponenten.

Bei jedem Verbindungsaufbau zum Server der Princeton-University geht der Anwender davon aus, genau mit diesem Server verbunden zu sein. Durch Angriffe innerhalb des lokalen Netzes, z.B. durch Manipulation auf Ebene des Adress Resolution Protocol (vgl. Kapitel 7.1) oder im Internet durch Angriffe auf Router (vgl. Kapitel 7.6) oder DNS-Server (vgl. Kapitel 7.13) wird der Anwender jedoch auf den Server eines Angreifers umgelenkt.

Über Steuerelemente auf seinem eigenen Server, z.B. JavaScript oder ActiveX versucht der Angreifer dann, den Rechner des Anwenders zu manipulieren (vgl. [Kyas 1998] S.203ff). Erweitert wird diese Problematik, da

vom offiziellen Webserver der Princeton-Universität auch Benutzerschnittstellen zur lokalen Installation heruntergeladen werden können. Werden diese vom Angreifer manipuliert, erhält er volle Kontrolle über den Rechner des Anwenders.

Eine ähnliche Problematik zeigt sich bei der Verwendung einer der vielen Schnittstellen, die in Projekten rund um *WordNet* entstanden sind. Diese Komponenten laufen ebenfalls lokal auf dem Rechner des Anwenders ab, der in der Regel nicht in der Lage ist, das Gefahrenpotential abzuschätzen. Selbst in den Fällen, wo der Quellcode vollständig offengelegt ist, bleibt das Problem der Analyse dieses Quellcodes, die ein hohes Maß an Sachkenntnis und einen hohen Zeitaufwand erfordert. Ist der Quellcode nicht verfügbar, d.h. es wird nur ein ausführbares Programm zum Download zur Verfügung gestellt, hat der Anwender keine Möglichkeit der Prüfung.

Durch den Einsatz von Server-Zertifikaten und digitalen Signaturen (vgl. Kapitel 3.7) zur Sicherstellung der Integrität von Anwendungen könnte diese Situation verbessert werden. Diese Möglichkeiten werden jedoch in *WordNet* und den damit verbundenen Projekten bisher noch nicht genutzt.

11.2 COSMAS

COSMAS ist die Abfragekomponente zur Korpussammlung des Instituts für Deutsche Sprache in Mannheim. Die Sammlung enthält natürlichsprachliche Texte zur Volltextrecherche, die zum Teil automatisch morphosyntaktisch analysiert und annotiert sind. Aktuell beträgt die Textmenge über 900 Millionen laufende Wortformen (Stand November 2001) und resultiert zum größten Teil aus der Auswertung von Zeitungstexten.

Mittels eines einfachen Web-Interface ist die Recherche über die verschiedenen Korpora möglich (s. Abbildung 12.2), die Kommunikation erfolgt dabei unverschlüsselt.

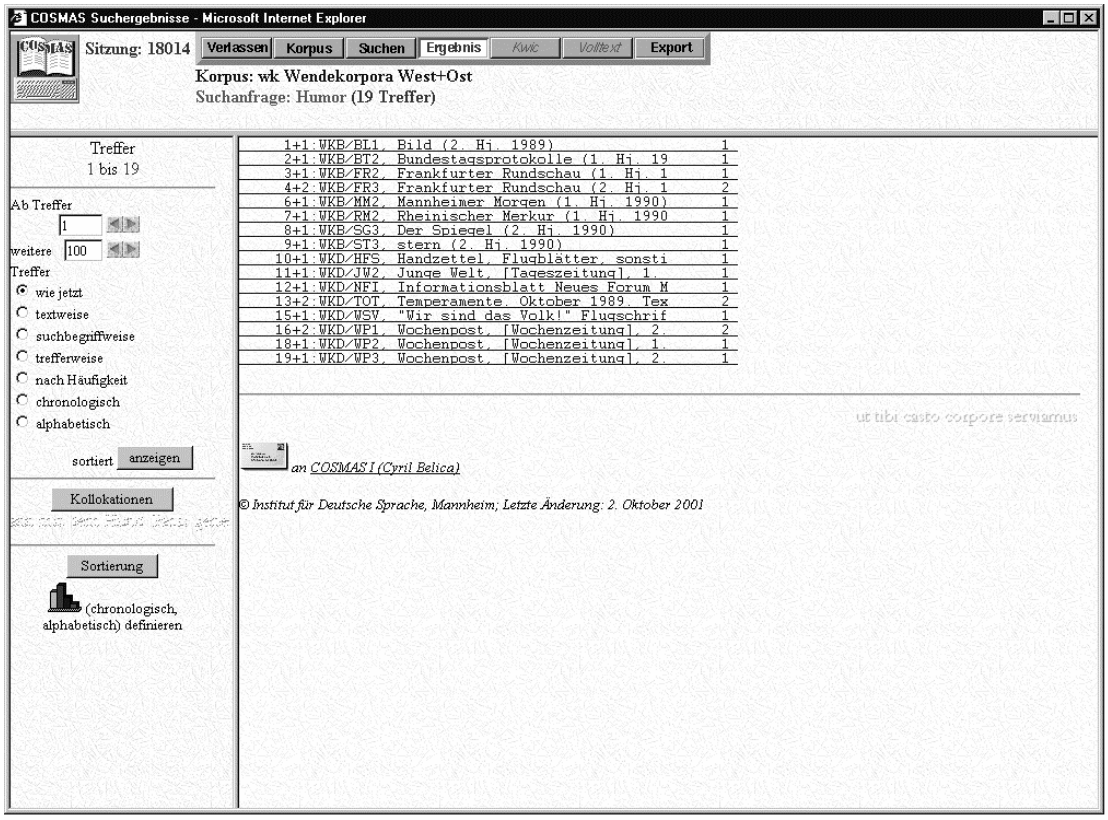


Abbildung: 11.2 Recherche im Wendekorpus

Die kommerzielle Nutzung der öffentlich zugänglichen Informationen ist untersagt, und der Benutzer wird beim Zugriff auf *COSMAS* darauf hingewiesen. Ein schreibender Zugriff auf den Datenbestand ist über das Internet nicht möglich.

Der Zugriff auf andere als die öffentlich zugänglichen Informationen ist nur intern im IDS möglich. Dies betrifft Korpora, die Rechte Dritter enthalten und daher nicht im Internet zur Verfügung gestellt werden können. Die Trennung des Zugriffs erfolgt über die Identifikation der IP-Adresse der zugreifenden Maschine. Wird ein Adressbereich identifiziert, der nicht dem IDS zugeordnet

ist, so wird der Zugriff verweigert. Weitere Details zu diesem Verfahren wurden vom IDS mit Hinweis auf Sicherheitsbedenken nicht zur Verfügung gestellt.

Der Ansatz, Details zu sicherheitsrelevanten Mechanismen nicht zu veröffentlichen, ist natürlich nachvollziehbar. Im Bereich der Kryptographie hat sich jedoch gezeigt, daß der Versuch der Geheimhaltung in der Regel fehlschlägt (vgl. Kapitel 3.1). Das Prinzip der Geheimhaltung, auch als „Security by Obscurity“ bezeichnet, führt dazu, daß eine Prüfung der Verschlüsselungsverfahren in größerem Umfeld nicht stattfinden kann und Sicherheitsmängel nicht behoben werden.

Grundsätzlich ist das dargestellte Verfahren der Identifikation von IP-Adressen angreifbar. Über das in Kapitel 7.6 angesprochene Loose Source Routing kann z.B. der Weg eines IP-Paketes vom Empfänger zum Ziel vorgegeben werden. Handelt es sich dabei um ein Datenpaket mit falscher IP-Adresse würde der absendende Rechner unabhängig von seinem tatsächlichen Standort als interner Rechner des IDS erkannt. Auch die Angriffe über das Routing Information Protocol könnten erfolgreich sein, über das tatsächliche Sicherheitsniveau im IDS kann - aufgrund fehlender Informationen - jedoch keine Aussage getroffen werden.

11.3 WordbanksOnline

Collins *WordbanksOnline* ist die Schnittstelle zu einem 56 Millionen Wortformen umfassenden Textkorpus der geschriebenen und gesprochenen englischen Sprache. Die Inhalte von *WordbanksOnline* wurden größtenteils aus der Bank of English übernommen und ergänzt um Transkripte von Radiosendungen und anderen formellen und informellen

Sprachaufzeichnungen. Auf diese Datenbasis kann entweder mittels eines Java-Applet (vgl. Abbildung 11.3) oder über Telnet zugegriffen werden.

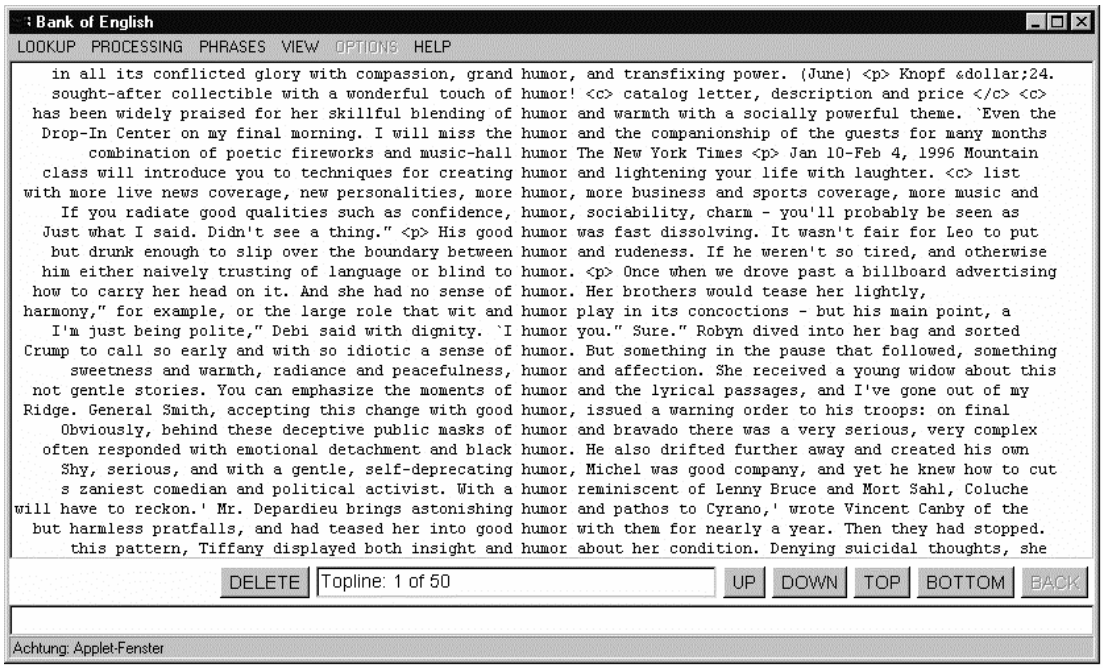


Abbildung 11.3: Java-Interface zu WordbanksOnline

Für den Zugriff über Java gilt die schon für *WordNet* skizzierte Problematik, zumal auch hier das Applet nicht signiert ist. Die Verschlüsselung der Kommunikation mittels Secure Socket Layer (vgl. Kapitel 9.5.1) und der Einsatz von Zertifikaten würden die Problematik entschärfen.

Für die Abfrage von *WordbanksOnline* über Telnet ist eine Authentifizierung mittels Paßwort erforderlich. Die Übertragung des Paßworts erfolgt unverschlüsselt. Dadurch ist ein Mitlesen des Paßworts durch einen Angreifer problemlos möglich, wie in Kapitel 10.2 gezeigt wurde. Darüber hinaus ist die Übernahme der Verbindung nach Authentifizierung möglich (vgl. Kapitel 6.5). Hier würde der Einsatz einer Secure Shell die Situation verbessern, d.h. die Verschlüsselung der übertragenen Daten oberhalb von OSI-Layer 3, z.B. mittels des oben genannten Secure Socket Layer (vgl. Kapitel 9.3.1 und zur Einordnung in ein Schichtenmodell Kapitel 9.3.8).

Da eine direkte Anmeldung am UNIX-Betriebssystem erfolgt (vgl. Abbildung 11.4), sind auch die in Kapitel 8 dargestellten Sicherheitsrisiken relevant, z.B. der Angriff auf falsch implementierte Shell-Übergänge (im Beispiel wird ein Shellskript zur Ermittlung des Terminal-Typs eingesetzt). Erschwerend kommt hinzu, daß der eingesetzte Server weitgehend ungesichert ist. Eine Übertragung der Datei /etc/passwd (vgl. Kapitel 8.2) über den anonymen FTP-Zugang ist problemlos möglich. Auch wenn die eigentlichen Paßworte hier nicht hinterlegt sind, erhält ein Angreifer so wertvolle Hinweise auf angelegte Benutzer und die ID von Systemprozessen.

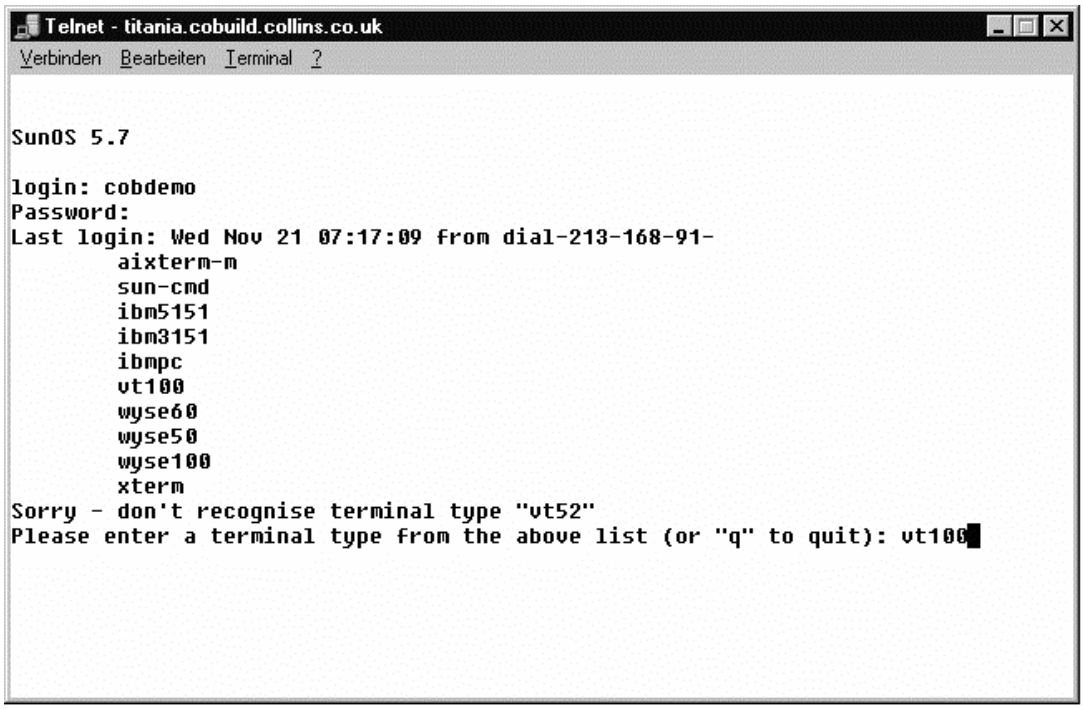


Abbildung 11.4: Login auf der Telnet-Schnittstelle

WordbanksOnline wird kommerziell genutzt, es steht aber ein Demo-Account zur unentgeldlichen Nutzung Verfügung, der die Zahl der angezeigten Ergebnisse reduziert oder den Suchstring auf Wortformen einschränkt, die mit einen bestimmten Buchstaben beginnen.

11.4 BLISNet

BLISNet ist die Abfragekomponente zur Bonner Wortdatenbank, die als kumulierte Wortdatenbank der deutschen Sprache Grund- und Vollformen mit morphologischen, morpho-syntaktischen, syntaktisch-semantischen Angaben sowie Angaben zur Pragmatik enthält. Detaillierte Informationen über Aufbau und Struktur der Datenbank sind in [Brustkern 1992], [Elsen 1996] sowie [Messelken 1997] enthalten. In [Hartmann 1997] findet sich eine ausführliche Beschreibung der *BLISNet*-Komponente bezüglich Design, Implementierung und Funktion.

In der ersten Version war *BLISNet* als reine Client/Serveranwendung in den Programmiersprachen C und TCL/TK realisiert. Die neuere Version 2 unterscheidet sich durch den Einsatz der Internet- Standardtechnologien, insbesondere des World Wide Web und der Programmiersprachen Java und Javascript. Zu einem Server der Version 2 kann über jeden WWW-Browser eine Verbindung aufgebaut werden. Der Server stellt eine Verbindung zur Datenbasis her, bereitet die gefundenen Informationen auf und erzeugt dynamisch eine HTML-Seite, die diese Informationen im Browser des Benutzers zur Bearbeitung darstellt. Abbildung 11.5 zeigt eine Abfrage über dynamisch erzeugte HTML-Seiten in *BLISNet* Version 2.

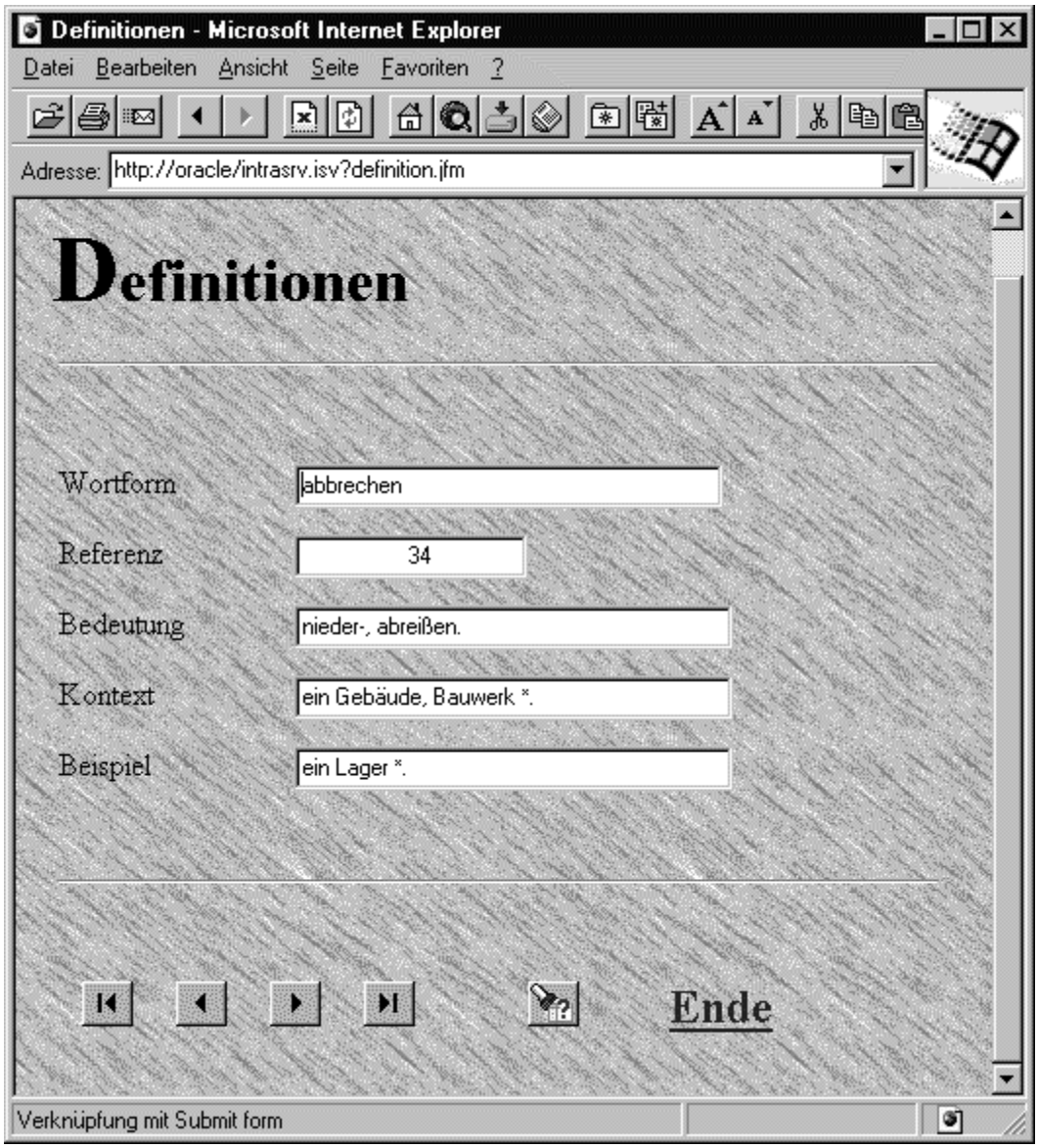


Abbildung 11.5: Abfrage mit dem BLISNet-Interface

Die bisherigen Sicherheitsmechanismen von *BLISNet* umfaßten eine Paßwortabfrage und die Prüfung der IP-Adresse, von der aus ein Zugriff erfolgte. Zugriff auf die Datenbasis wurde erteilt, wenn Benutzername und Paßwort korrekt eingegeben wurden und der Rechner des Benutzers in der Liste zugriffsberechtigter Maschinen eingetragen war.

Ohne weitere Maßnahmen sind diese einfachen Sicherheitsmechanismen jedoch angreifbar. In den Kapiteln 7.1 und 7.2 wurde dargestellt, daß ein Angreifer über verschiedene Arten des *Spoofing* auf unterschiedlichen

Protokollebenen eine falsche Identität annehmen kann. Mechanismen, die auf der Prüfung von IP-Adressen beruhen, können durch Vortäuschen einer falschen Zuordnung zwischen IP-Adresse und Hardware-Adresse angegriffen werden (ARP-Spoofing). Das angegriffene System sendet dann alle Pakete an das System des Angreifers. Schließt die Identifikation von Systemen die Angabe eines DNS-Namens ein, so wird ein Angriff mittels DNS-Spoofing möglich.

Die Übertragung von Paßwort-Informationen birgt das Risiko eines *Sniffing*-Angriffes, sofern nicht starke Verschlüsselung zum Einsatz kommt. In *BLISNet* Version 2 wurde der *Base64-Algorithmus* eingeführt, um ein Mitlesen des Benutzerpaßwortes zu erschweren. Einem gezielten Angriff hält diese Methode jedoch nicht stand, da Base64 ein einfacher, umkehrbarer Algorithmus ist, der auch im Hypertext Transfer Protokoll (HTTP) eingesetzt wird, um die Existenz eines Paßwortes zu verbergen. Verfügt ein Angreifer über detaillierte Kenntnisse des TCP/IP-Protokollstack, so kann das Paßwort protokolliert und in Klartext umgesetzt werden.

Über einen *Hijacking*-Angriff kann eine einmal etablierte Verbindung zur Bonner Wortdatenbank auf den Rechner des Angreifers umgelenkt werden. Dieser Angriff wird zu einem Zeitpunkt durchgeführt, an dem die Prüfung der Identität des Benutzers bereits abgeschlossen ist. Der Angreifer arbeitet dann mit den Rechten des regulären Benutzers weiter.

11.5 Sicherheitsanforderungen an die Bonner Wortdatenbank

Die dargestellten Systeme zeigen ein abgestuftes Sicherheitsniveau. WordNet (auch GermaNet und EuroNet) arbeitet ohne Schutzmechanismen, COSMAS sicherert den Zugriff durch Identifikation der IP-Absendeadresse, WordBanksOnline arbeitet mit Passwort-Abfrage auf der Textoberfläche und BlisNet nutzt sowohl die Adressprüfung als auch die Passwortabfrage.

Beim Design von *BLISNet* standen Funktionalität und einfacher Aufbau mittels Standarddiensten im Vordergrund, nicht jedoch erhöhte Sicherheitsanforderungen. Darüber hinaus bestand die Annahme, daß die Inhalte der Bonner Wortdatenbank nicht mit dem gleichen Aufwand zu schützen seien wie z.B. die Übertragung von Kreditkarteninformationen im Internet, d.h. daß der Widerstandswert des Systems nicht vergleichbar sein muß mit kommerziellen Systemen, die hochsensible Informationen transportieren.

Diese Annahme wurde revidiert aufgrund der oben geschilderten Angriffsmöglichkeiten und folgender Überlegungen, die davon ausgehen, daß die Kommunikation mit der Wortdatenbank durch stärkere Mechanismen als Paßworte und Adreßprüfungen geschützt werden muß.

Zunächst ist sichere Kommunikation notwendig, da über Identifizierung und Authentifizierung eines Benutzers Zugriffsrechte auf die Datenbasis vergeben werden, die auch das Recht zur Modifikation oder Löschung von Datensätzen beinhalten können. Ein Angreifer, der sich unberechtigt Schreibzugriff auf das System verschafft, ist so in der Lage, die Datenbasis in einen inkonsistenten Zustand zu versetzen. Bei den anderen hier vorgestellten Systemen ist der schreibende Zugriff durch authentifizierte Benutzer nicht vorgesehen, daher tritt das Problem z.B. bei *WordNet* nicht auf. Im Sinne einer Zusammenarbeit von Linguisten über das Internet ist diese Funktion aber durchaus sinnvoll und nützlich. Die Erstellung und Plege einer Wortdatenbank ist ein zeitintensiver Prozess, der durch Nutzung des Internet und der damit verbundenen Ressourcenbündelung erheblich verbessert werden kann. Eine sichere Identifikation der beteiligten Wissenschaftler ist dafür jedoch unbedingt erforderlich.

Auch im Hinblick auf eine mögliche kommerzielle Nutzung der Wortdatenbank in der Zukunft ist eine sichere Identifizierung einzelner Benutzer erforderlich. Einerseits muß sichergestellt werden, daß kein Zugriff auf Informationen erfolgt, deren Bereitstellung gebührenpflichtig ist, andererseits muß auch gewährleistet sein, daß nur Leistungen abgerechnet werden, die tatsächlich von einem bestimmten Benutzer in Anspruch genommen wurden.

An ein System mit den geschilderten Möglichkeiten müssen auch höhere Anforderungen im Bezug auf Sicherheit gestellt werden. Diese Annahme führte zu einem Modell der sicheren Kommunikation mit der Wortdatenbank, das in den folgenden Abschnitten vorgestellt wird.

12 Ein Modell für die Bonner Wortdatenbank

12.1 Anforderungen

In Kapitel 2.4 wurden die zu erreichenden Sachziele im Zusammenhang mit der sicheren Kommunikation über ein unsicheres Medium dargestellt. Unter vertrauenswürdiger Kommunikation wird im Rahmen dieser Arbeit die vertrauliche und integere Übermittlung von Inhalten an einen Kommunikationspartner verstanden, der zuvor sicher identifiziert wurde. Um der durch das Medium hervorgerufenen Anonymität entgegenzuwirken, ist die Zuordnung einer Person zu ihrer behaupteten Identität notwendig sowie der Schutz der übermittelten Inhalte gegen unberechtigte Kenntnisnahme und Manipulation. Die Primärziele *Vertraulichkeit, Integrität* und *Authentizität* bilden somit die Anforderungen an ein Modell der sicheren Kommunikation für die Bonner Wortdatenbank.

12.2 Das Modell

Die gestellten Anforderungen führten zum Entwurf eines Modells zur Sicherung der Kommunikation mit der Bonner Wortdatenbank. Dieses Modell stellt zugleich einen neuen Ansatz zur Integration von Rechner- und Netzwerksicherheit dar. Basierend auf den Möglichkeiten starker Kryptographie wird hier gezeigt, wie ein real existierendes System abgesichert werden kann. Die verwendeten Komponenten und Techniken werden in den folgenden Abschnitten detailliert dargestellt.

Betrachten wir zunächst den Grundaufbau des Modells. Im einfachsten Fall läuft die Interaktion eines Benutzers mit der Bonner Wortdatenbank über eine einfache Client/Server- Verbindung ab, wie in Abb. 12.3 dargestellt.

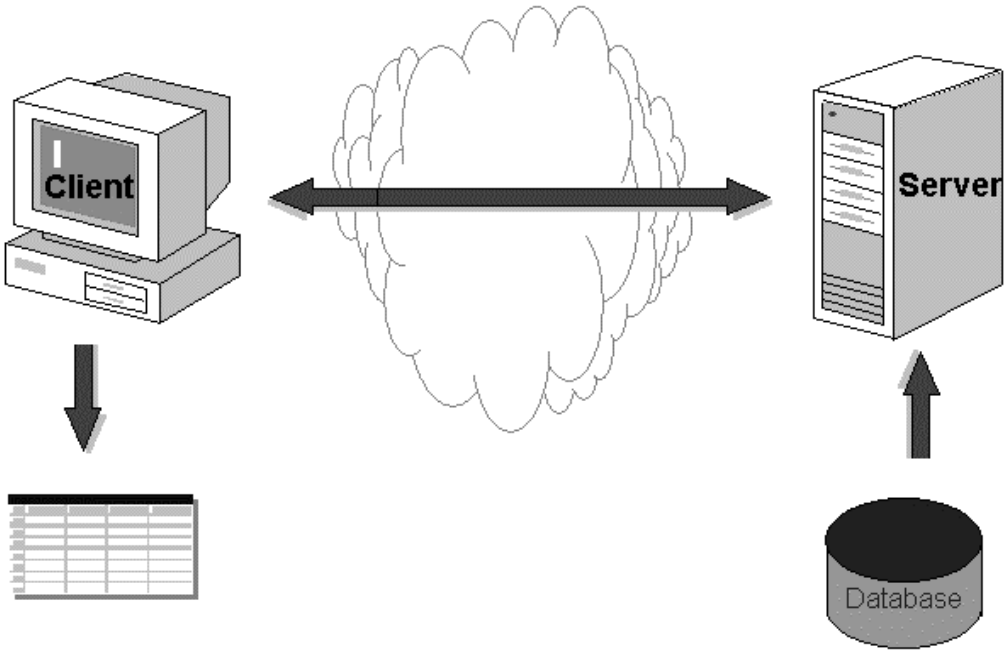


Abbildung 12.3: Einfaches Client/Server-Modell

Bisher erfolgte die Übertragung zwischen Client und Server unverschlüsselt, auch das Benutzerpaßwort lag in einem Format vor, das bei entsprechendem Kenntnisstand in Klartext überführt werden konnte. Der Client wurde vom Server über Prüfung der IP-Absendeadresse authentifiziert.

In einem Modell mit höherem Widerstandswert muß zunächst die Verbindung zwischen Client und Server verschlüsselt werden, um das Mitlesen der übertragenen Nutzdaten und des verwendeten Paßwortes zu verhindern. Zur Verschlüsselung wird ein symmetrisches Verfahren (vgl. Kapitel 3.2) eingesetzt. Der Schlüssel wechselt bei jeder neuen Verbindung. Der Einsatz eines symmetrischen Verfahrens mit *session key* wird in Abbildung 12.4 dargestellt.

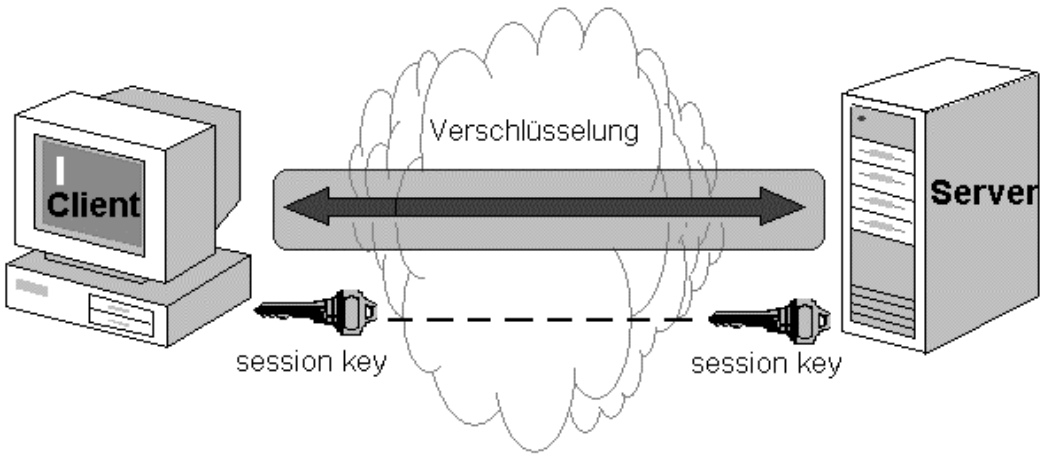


Abbildung 12.4: Sicherung der Vertraulichkeit durch Verschlüsselung

Die Verschlüsselung garantiert *Vertraulichkeit* und kann auf verschiedenen Ebenen des OSI-Modells realisiert werden. Denkbar wäre z.B. der Einsatz der in Kapitel 9.3.1 beschriebenen *Secure Sockets Layer* (OSI-Schicht 5) oder eine Verschlüsselung auf Anwendungsebene (OSI-Schicht 7). Gegen den Einsatz von SSL spricht, daß dieser Mechanismus nicht die Authentifizierung von Benutzern einschließt und das Problem der Verwaltung von öffentlichen Schlüsseln nicht zufriedenstellend gelöst ist. Mittels SSL kann eine gegenseitige Authentifizierung zwischen Client und Server erfolgen, nicht jedoch eine Authentifizierung zwischen Benutzer und Server. Die Verschlüsselung auf Anwendungsebene kann den Benutzer mit einbeziehen und ist darüber hinaus flexibler bei der Schlüsselverwaltung.

Für die Bonner Wortdatenbank wurde ein Konzept entwickelt, über welches Benutzer eindeutig gegenüber dem Server authentifiziert werden und in dem die Verschlüsselung der Kommunikation durch starke Kryptographie erfolgt. In diesem Konzept werden digitale Signaturen (vgl. Kapitel 3.6) eingesetzt, die einerseits zur Authentifizierung und andererseits zum Austausch der verwendeten *session keys* dienen. Während eine einfache Verschlüsselung nur die Vertraulichkeit der Kommunikation zwischen Client und Server

sicherstellt, bietet der Einsatz von Signaturen die Möglichkeit, eine Vertrauensstellung zwischen Benutzer und Server aufzubauen.

Der Benutzer ist im Besitz eines *Signaturschlüssels*, d.h. der privaten Komponente eines asymmetrischen Schlüsselpaares. Mittels dieses Schlüssels kann er vom Server gesendete Daten signieren, die der Server mittels der öffentlichen Komponente des Schlüssels prüft. Stimmen die signierten Daten mit den ursprünglich gesendeten überein, so ist der Benutzer gegenüber dem Server authentifiziert.

Die Einschränkung auf bestimmte, zum Verbindungsaufbau mit dem Server berechtigte Maschinen entfällt an dieser Stelle, da die sichere Identifikation des Benutzers die Prüfung einer IP-Absendeadresse überflüssig macht. Abbildung 12.5 stellt die geschilderten Mechanismen dar.

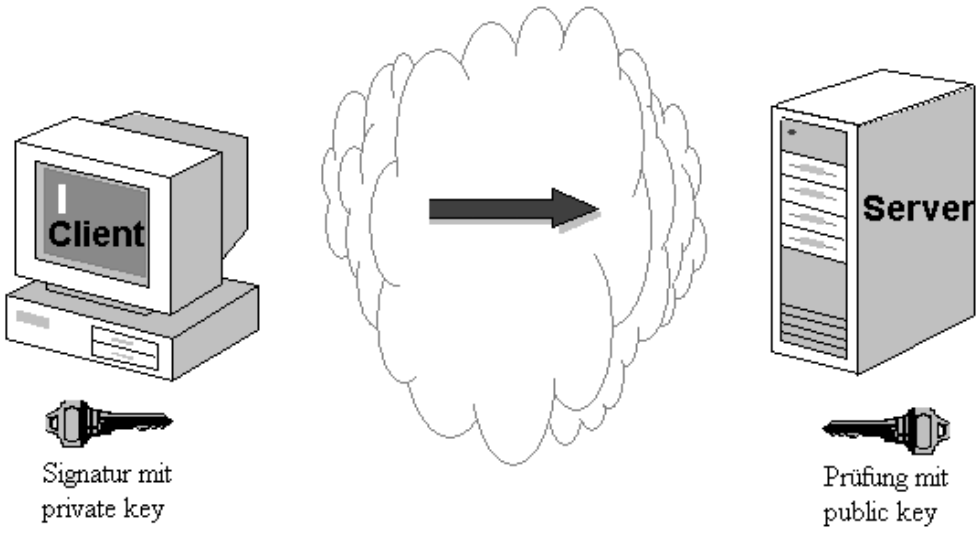


Abbildung 12.5: Authentifizierung durch Signatur

Der Einsatz digitaler Signaturen erfordert die Einführung von Komponenten zur Generierung, Distribution, Prüfung und sicheren Speicherung dieser Schlüssel. Die Generierung von Schlüsseln erfolgt über eine neue

Komponente des Systems, welche Teilfunktionen einer Zertifizierungsstelle ausführt (vgl. Kapitel 3.7) und daher vereinfachend auch als *Certification Authority* (CA) bezeichnet wird. Über diese Komponente werden Benutzer angelegt und verwaltet, sowie Schlüssel erzeugt und Benutzern zugewiesen.

Die Distribution und sichere Speicherung von Signaturschlüsseln erfolgt über den Einsatz von Chipkarten. Öffentliche Schlüssel werden zusammen mit den zugehörigen Benutzerinformationen in einem Datenbanksystem gespeichert. Das Gesamtsystem besteht damit aus Client, Server, CA, Datenbank und Chipkarte. Die Funktionen der einzelnen Komponenten werden in den folgenden Abschnitten vorgestellt.

12.2.1 Client und Server

Der *Server* nimmt Verbindungen von Clients an, führt die notwendige Authentifizierung durch und bearbeitet die gestellte Anfrage. Er arbeitet als paralleler Server, d.h. mehrere Clients können gleichzeitig eine Verbindung aufbauen. Die zur Authentifizierung der Clients benötigten Informationen (z.B. den zur Prüfung der Signatur erforderlichen *public key*) erhält der Server aus der angeschlossenen Datenbank.

Der *Client* baut die Verbindung zum Server auf, identifiziert und authentifiziert sich und sendet Anfragen, die der Server bearbeitet. Da die Architektur plattformunabhängig implementiert werden soll, kommt der Einsatz von Standardsoftware nicht in Frage. Darüber hinaus erfüllt ein einfacher WWW-Server nicht die gestellten Anforderungen hinsichtlich Verschlüsselung und digitaler Signatur auf Anwendungsebene. Aktuell erfolgt daher eine Realisierung mittels der Programmiersprache Java. Die Sicherheitsaspekte von Java werden in Abschnitt 12.5.1 dargestellt.

Im Verlauf der Ausarbeitung des Modells wurde ein Authentifizierungs-Protokoll entwickelt, welches auch zum Schlüsselaustausch eingesetzt werden kann. In seiner einfachsten Fassung sieht es die Erzeugung einer Zufallszahl durch den Server vor, die vom Client signiert wird. Der Server prüft im nächsten Schritt die Signatur mittels des gespeicherten öffentlichen Schlüssels. Bei Übereinstimmung mit der gesendeten Zufallszahl erzeugt der Server einen *session key* und sendet diesen, verschlüsselt mit dem *public key* des Client, an den Server zurück. Abbildung 12.6 verdeutlicht das Verfahren.

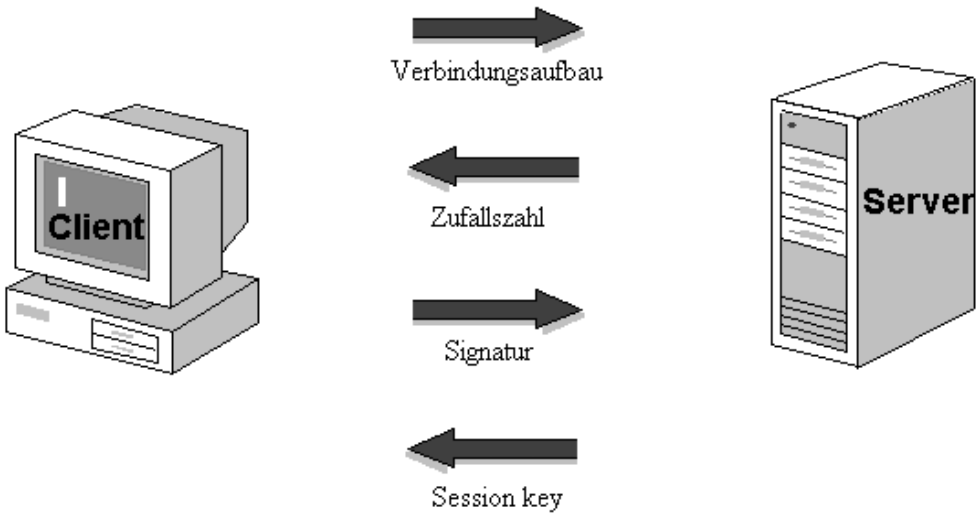


Abbildung 12.6: Authentifizierung und Schlüsselaustausch

Eine Erweiterung dieses Protokolls umfaßt die Prüfung der Integrität der Java Virtual Maschine auf dem Client und fügt in die Signatur einen MD5-Hash ein (mittels einer XOR-Verknüpfung).

Wird zwischen Client und Server nach der Authentifizierung ein sicherer Kanal über eine neue Verbindung etabliert, kann die Antwort des Servers erweitert werden. In diesem Falle wird nicht nur der *session key* übermittelt, sondern auch ein TCP-Serverport zum Aufbau der neuen Verbindung.

Anhand einer eingefügten Zeichenkette prüft der Client, ob auf der Seite des Servers mit dem korrekten *public key* verschlüsselt wurde, da die Richtigkeit von *session key* und TCP-Port erst bei einem tatsächlichen Verbindungsaufbau geprüft werden kann. An dieser Stelle kann in die Antwort auch die vom Server erzeugte Zufallszahl eingehen, die der Client ja bereits entschlüsselt hat. Auf diese Weise kann Replay-Angriffen wirksam vorgebeugt werden. Eine vollständige Antwort enthält einen zusätzlichen Header, der aus einer XOR-Verknüpfung von Zufallszahl und Zeichenkette gebildet wurde, wie in Abbildung 12.7 dargestellt.

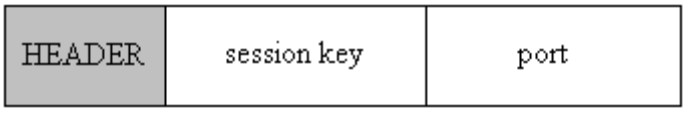


Abbildung 12.7: Aufbau der Antwort des Servers

Die in BLISNet eingeführte Prüfung der IP-Absendeadresse kann zwar aufgrund des Einsatzes von digitalen Signaturen übergangen werden, ist jedoch weiterhin möglich. Die bisher erforderlichen manuellen Änderungen können im neuen Modell vermieden werden. Ist die Änderung der IP-Adresse eines Rechners erforderlich, kann die neue IP-Adresse dem Server mitgeteilt werden. Dazu wird die neue Adresse und der zugehörige Maschinenname signiert und an den Server übertragen. Anhand der Signatur prüft der Server, ob die Änderung berechtigt ist. Bei Distribution der Client-Software durch einen FTP-Server ist eine Integritätsprüfung durch den Client möglich, sofern die Software digital signiert wurde.

12.2.2 CA und Datenbank

Mit Hilfe der CA-Komponente werden Benutzer des Systems verwaltet. Bei Erstellung eines neuen Benutzers werden Name und E-Mail-Adresse gespeichert und ein RSA-Schlüsselpaar generiert. Der private Teil des Schlüssels wird auf einer Chipkarte (s.u.), der öffentliche Teil in einer Datenbank gespeichert, auf die später auch vom Server zugegriffen werden kann, um die Signatur des Client zu prüfen.

Da der öffentliche Schlüssel zur Identifikation einzelner Clients und auch zur Übermittlung von Informationen (*session key*, TCP-Port) dient, wird er verschlüsselt in der Datenbank abgelegt. Zum Start der CA-Komponente ist

ein Paßwort für diese symmetrische Verschlüsselung anzugeben, so daß auch der Zugriff auf den öffentlichen Schlüssel geschützt abläuft.

Um Daten zwischen CA und Datenbank zu übertragen, kann die *Java Database Connection* (JDBC) verwendet werden, da die oben angeführte Verschlüsselung bereits vor der Datenübertragung stattfindet, die Daten daher nie im Klartext übertragen oder abgelegt werden. Optional können die vollständigen Benutzerdaten verschlüsselt werden.

Durch den Einsatz von JDBC kann nahezu jede auf dem Markt verfügbare Datenbank in das System integriert werden.

12.2.3 Chipkarte

Der private Schlüssel des Benutzers muß besonders geschützt werden, da er die einzige Möglichkeit zur sicheren Identifizierung eines rechtmäßigen Zugriffs auf die Bonner Wortdatenbank darstellt. Microprozessor-Chipkarten sind für diese Zwecke ideal, da sie ein in sich geschlossenes, sicheres System zur Speicherung sensibler Daten bilden. Auf diese Weise wird der *private key* eines Benutzers erst nach Prüfung einer PIN zugreifbar und ist deshalb sicher gespeichert.

Problematisch ist hier einerseits die Vielzahl der am Markt befindlichen Kartenleser und Chipkarten, andererseits die Realisierung des betriebssystem-unabhängigen Zugriffs auf die Chipkarten. Es existieren verschiedenen Bestrebungen, eine einheitliche Schnittstelle zur Chipkarte zu implementieren. Die wichtigsten sind das unter Federführung der Telekom entwickelte *Card Terminal API* (CTAPI) und das *PC-Smartcard Interface* (PC/SC) von Microsoft. Letzteres ist inzwischen auch auf verschiedene

UNIX-Plattformen portiert worden (Linux, Solaris, AIX) und durch den Einsatz in Windows-Produkten weit verbreitet.

Das von namenhaften Herstellern (IBM, VISA, SUN, Netscape u.a.) gegründete *OpenCard Framework* (OCF) bietet die Möglichkeit, aus Java-Programmen unabhängig vom verwendeten Betriebssystem Chipkarten-Terminals anzusprechen. Hier werden nicht nur einige weit verbreitete Terminals unterstützt, sondern auch alle Kartenterminals, die der PC/SC-Spezifikation genügen. Durch den kombinierten Einsatz eines PC/SC-kompatiblen Kartenlesers und des OpenCard Framework wird der Chipkarten-Zugriff unter diversen Betriebssystemen möglich.

12.2.4 Systemübersicht

Durch eine Unterteilung zwischen den administrativen Komponenten CA und Chipkarte einerseits und den operationalen Komponenten Client und Server andererseits, die beide gemeinsam auf die Datenbank zugreifen, ergibt sich ein zweiteiliger Aufbau des Systems. Bevor eine Kommunikation mit der Wortdatenbank möglich ist, muß zunächst ein Benutzer angelegt und ein zugehöriges Schlüsselpaar generiert werden. Der öffentliche Schlüssel wird in der Datenbank abgelegt, der private Schlüssel sicher auf der Chipkarte gespeichert, wie in Abbildung 12.8 dargestellt.

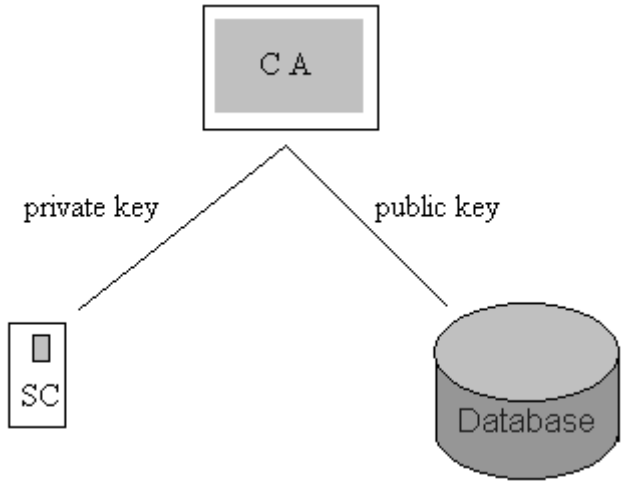


Abbildung 12.8: Generierung und Verteilung eines Schlüsselpaares

Die Chipkarte wird im nächsten Schritt zur Authentifizierung des Client (die zu Beginn einer Verbindung von Client und Server abläuft) eingesetzt, wie Abbildung 12.9 zeigt. Nun kann der eigentliche Prozeß der Informationsabfrage erfolgen.

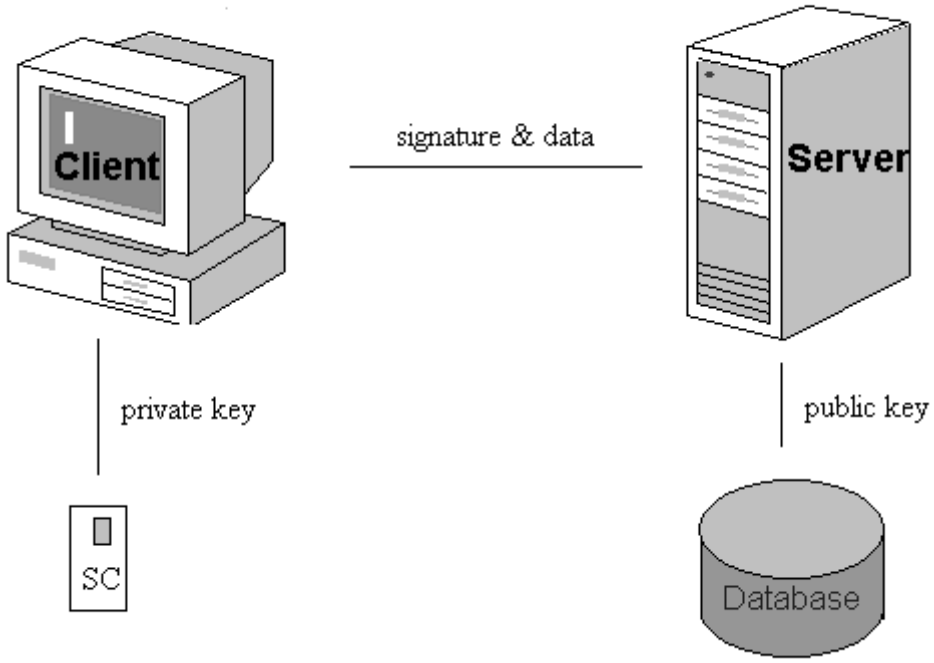


Abbildung 12.9: Authentifizierung mittels Signatur

An dieser Stelle wird gewährleistet, daß der Benutzer über die Eingabe seiner PIN gegenüber der Chipkarte authentifiziert ist und die Chipkarte im nächsten Schritt den Signaturschlüssel freigibt, der für die Authentifizierung des Client gegenüber dem Server notwendig ist.

An den Prozeß der Authentifizierung schließt sich die Übergabe eines *session key* an, der zur symmetrischen Verschlüsselung verwendet wird. Durch den Einsatz dieses Schlüssel wird die Vertraulichkeit der Kommunikation sichergestellt.

12.3 Sicherheit des Systems

Die vorgestellten Komponenten bilden ein Gesamtsystem zur sicheren Kommunikation im Internet. Die Inhalte der Bonner Wortdatenbank können über dieses System sicher abgefragt und verändert werden, Vertraulichkeit wird durch Verschlüsselung realisiert, Integrität und Authentizität durch Einsatz digitaler Signaturen. Die einzelnen Mechanismen werden in den folgenden Abschnitten dargestellt.

12.3.1 Datenübertragung

Die Datenübertragung zwischen Client und Server wird über die Algorithmen DES oder IDEA (vgl. Kapitel 3.2) verschlüsselt. Einfache *Sniffing*-Angriffe durch den Einsatz von Hackertools werden damit wirkungsvoll verhindert. Nicht ausgeschlossen werden kann ein Angriff, der mehrere Datenübertragungen protokolliert und sie miteinander vergleicht, da auf Anwendungsebene verschlüsselt wird und dieses Verfahren für niedrigere

Schichten transparent ist, d.h. Zieladresse und TCP-Port sind im IP- bzw. TCP-Header erkennbar.

Dieses Szenario bietet theoretisch Raum für einen Angriff nach der Known-Plaintext-Methode (vgl. Kapitel 3.10). Hier nutzt der Angreifer wiederkehrende, bekannte Muster im Chiffrat um durch Absuchen des Schlüsselraumes mit verschiedenen selbstgenerierten *session keys* den passenden Schlüssel zu finden. Im vorgestellten Modell wird dieser Angriff durch den Einsatz einer Stromchiffre, DES im CBC- Modus (vgl. Kapitel 3.2), unterbunden.

Im *CBC-Modus* fließt das Ergebnis der DES-Verschlüsselung eines Datenblockes in den nächsten Datenblock mit ein, so daß gleiche Klartextblöcke in einen unterschiedlichen Chiffretext umgesetzt werden. Durch diese Technik wird ein Angriff verhindert, da bekannter Klartext, etwa Protokoll-Header oder standardisierte Antworten des Servers, unterschiedliche Muster liefert. Ebenso werden aktive Angriffe durch Permutation oder Wiedereinspielung einzelner Chiffretext-Blöcke abgewehrt, da Veränderungen am Datenstrom sich unmittelbar auswirken und bemerkt werden.

Die Verwendung kryptographisch sicherer *Zufallszahlen* beugt Replay-Angriffen vor. Würde der Server dem Client eine vorhersagbare Zahl oder Zeichenkette zur Signatur liefern, würden auch alle Ergebnisse vorhersehbar sein. In diesem Falle könnte ein Angreifer versuchen, eine Signatur des Client erneut einzuspielen, die vom Server akzeptiert würde. Ist der Bereich der generierten Zufallszahlen genügend groß und sind diese Zahlen nicht vorhersehbar, hat ein solcher Angriff keine Aussicht auf Erfolg. Die Erzeugung von sicheren Zufallszahlen durch einen deterministischen Automaten ist, wie in Kapitel 3.8 dargestellt, nicht ganz unproblematisch. Die

Programmiersprache Java bietet einen Mechanismus, der zu diesem Zweck verwendet werden kann (s.u.).

Die Übernahme einer bestehenden Verbindung nach der Authentifizierung, der in Kapitel 6.5 geschilderte *Hijacking*-Angriff, wird zunächst dadurch erschwert, daß der Server dem Client eine zufällige Portadresse für einen sicheren Kanal zuweist, und wird vollends unmöglich gemacht durch die oben angeführte Verwendung einer Stromchiffre. Ein Angreifer kann den Server nur täuschen, wenn er den verschlüsselten Datenstrom weiterführt. Ohne Kenntnis des *session key* und der momentanen Parameter des DES-Algorithmus ist dies jedoch ausgeschlossen.

Durch Angriffe, die auf *Spoofing* basieren (ARP oder DNS), kann die Authentifizierung nicht beeinflußt werden, solange der private Schlüssel des Benutzers sicher gespeichert wird, da die IP-Adresse eines Systems nicht zur Identifikation verwendet wird.

Problematischer ist in diesem Zusammenhang die Möglichkeit, durch *Spoofing* den Verbindungsaufbau eines Client zu einem falschen Server zu erzwingen. Dieser Server könnte den Prozeß der Authentifizierung vortäuschen und vom Client wiederholt Signaturen über ausgewählte Zufallszahlen fordern. Geht der Client darauf ein, so gelang ein Angreifer in Besitz zusammengehöriger Challange-Response-Paare, d.h. Zufallszahlen und zugehöriger Signaturen, die später eingesetzt werden, um Zugriff auf einen echten Server zu erhalten.

Diese Art des Angriffs wird durch zwei Mechanismen verhindert. Zunächst erfolgt der Verbindungsaufbau immer vom Client aus und wird nach einer definierten Zeitspanne ohne Datenübertragung zum Server abgebrochen, so daß der Angriff sehr zeitintensiv wird. Weiterhin sorgt, wie im nächsten Abschnitt dargestellt, ein Signaturzähler auf der Chipkarte für eine

Begrenzung der maximal durchführbaren Signaturen. Ist die Zahl möglicher Signaturen im Verhältnis zu den verwendeten Zufallszahlen sehr klein, fällt die Erfolgsrate des Angriffs gegen Null.

12.3.2 Datenbank

Der öffentliche Schlüssel eines Client wird in einer Datenbank abgelegt, auf die der Server Zugriff hat, um die Signatur des Client prüfen zu können. Gelingt es einem Angreifer, einen selbstgewählten *public key* in die Datenbank einzufügen, würde der Server eine Signatur akzeptieren, die mit diesem Schlüssel assoziiert ist.

Ein solcher Angriff auf die Datenbank-Komponente kann entweder beim Transport der Daten während des Datenaustausches mit CA bzw. Server oder durch direkte Manipulation der Datenbank durchgeführt werden. Eine Möglichkeit, die Daten beim Transport zu schützen, besteht auch hier in der Verschlüsselung des Kommunikationskanals. Dieser Prozeß würde jedoch zusätzlichen Aufwand bedeuten und darüber hinaus keinen Schutz vor Innentätern mit direktem Zugriff auf den Datenbank-Server bieten.

Aus diesem Grund werden Benutzerdaten und *public key* vor der Übertragung verschlüsselt und später abgespeichert. Beim Zugriff auf die Daten findet die Entschlüsselung erst nach der Übertragung statt, so daß die Daten weder im Klartext übertragen noch gespeichert werden. Ein Angreifer kann ein RSA-Schlüsselpaar generieren, den öffentlichen Schlüssel jedoch nicht ablegen, ohne im Besitz des verwendeten symmetrischen Schlüssel zu sein.

Eine verfeinerte Variante des Angriffs besteht in der Vertauschung zweier öffentlicher Schlüssel in der Datenbank, so daß auch die Zugriffsrechte zweier Clients durch den Angreifer vertauscht würden. Dieser kann so die Identität eines regulären Benutzers annehmen. Durch einen zusammen mit dem *public key* abgelegten MD5-Hash über die Benutzerdaten läßt sich auch dieser Angriff abwehren.

12.3.3 Chipkarte

Zur Speicherung des *private key* sind Mechanismen mit abgestuftem Sicherheitsniveau möglich. Im einfachsten Fall wird der Signaturschlüssel ohne Schutzmaßnahmen auf einem *unsicheren Medium* (Diskette, Festplatte, Magnetkarte o.ä.) abgelegt und ist so allen Angriffen ausgesetzt, einschließlich unberechtigter Kenntnisnahme und Modifikation. Dieser Ansatz wird in keinem auf dem Markt befindlichen System gewählt. Das Mißbrauchsrisiko bei Verwendung des Schlüssels durch unberechtigte Dritte wäre zu hoch.

In der Praxis, z.B. beim weit verbreiteten PGP (vgl. Kapitel 9.3.5), wird der Signaturschlüssel meist mit einem symmetrischen Algorithmus verschlüsselt abgespeichert und erst nach Eingabe des korrekten Paßwortes freigeschaltet. Die Schlüsseldatei selbst liegt aber auch hier auf einem unsicheren Medium und kann daher von einem Angreifer entwendet werden, der dann versucht, über *eine Brute-Force-Attack* (vgl. Kapitel 3.10) die symmetrische Verschlüsselung zu brechen.

Ein deutlich höheres Sicherheitsniveau wird durch Speicherung des Signaturschlüssel auf einem *sicheren Medium* erreicht, d.h. einem externen Medium, welches den Lesezugriff durch Sicherheitsmaßnahmen schützt.

Chipkarten, besonders Mikroprozessor-Karten sind für diese Aufgabe ideal geeignet, da hier der Zugriff auf Dateiinhalte von der Eingabe einer Ziffern- oder Zeichenfolge abhängig gemacht werden kann, der *Personal Identifikation Number* (PIN). Anders, als der Name vermuten läßt, muß die PIN nicht aus einer Folge von Ziffern bestehen, sondern kann auch eine Zeichenkette oder eine Kombination aus beidem sein, die mit einem auf der Karte geschützt abgelegten Wert verglichen wird.

An mindestens zwei Stellen ist dieser Mechanismus beim Einsatz auf Personal Computern angreifbar. Einerseits sind Kartenterminals mit integriertem Ziffernblock zur Eingabe der PIN relativ wenig verbreitet, so daß die PIN über die Tastatur des Rechners eingegeben werden muß. An dieser Stelle kann durch den Einsatz eines Hintergrundprogrammes (vgl. Kapitel 8.3) ein Angriff erfolgen, der die PIN vom Benutzer unbemerkt mitprotokolliert. Andererseits wird auch bei diesem Ansatz der *private key* in den Rechner des Benutzers transferiert und dort zur Signatur eingesetzt. Sowohl beim Transfer als auch bei der Ablage im Hauptspeicher des Rechners sind Angriffe möglich.

Eine zusätzliche symmetrische Verschlüsselung, die von vielen Chipkarten beim Transport von Daten zur Anwendung unterstützt wird, erhöht das Sicherheitsniveau nur wenig, da die Problematik der Verarbeitung im Hauptspeicher bestehen bleibt.

Eine hinreichend sichere Durchführung der Signatur ist möglich, sofern sie in der Chipkarte erfolgt. Neuere Chipkarten enthalten eigene Kryptocontroller, die asymmetrische Verschlüsselungen (meist RSA) auf der Karte durchführen können. Die Karte erhält die zu signierenden Daten, führt die Signatur geschützt aus, und liefert das Ergebnis an die Anwendung zurück. Bei diesem Ansatz verläßt der Signaturschlüssel zu keiner Zeit die Karte und kann auch nicht aus der Karte ausgelesen werden.

Der *Signaturschlüssel* wird entweder extern erzeugt und beim Initialisierungsprozeß der Karte abgespeichert oder durch die Karte generiert. Die externe Erstellung des Signaturschlüssels bietet in geringem Maß immer noch Angriffsmöglichkeiten. Dies gilt beispielsweise, wenn es gelingt, während dieses Prozesses Veränderungen beim Transfer auf die Karte vorzunehmen, oder wenn bereits ein Angriff stattgefunden hat und die Generierungs-Komponente modifiziert wurde. Bei Verlagerung dieses Vorganges auf einen externen, nicht vernetzten Rechner sinkt das Risiko eines Angriffes jedoch auf ein vertretbares Niveau.

Die sicherste Methode besteht in der Erzeugung des Signaturschlüssels in der Karte. Hier wird ein Schlüsselpaar erzeugt und der öffentliche Schlüssel zur späteren Prüfung der Signatur freigegeben. Der private Teil bleibt in einem geschützten Bereich auf der Karte und kann auch vom Anwender nicht eingesehen werden. Bisher sind Algorithmen mit Schlüssellängen bis zu 1024 Bit auf Chipkarten implementiert.

Ein Gesamtsystem besteht aus Komponenten, die Daten im Internet oder im lokalen Netz vertraulich und integer übertragen. Diese Komponenten werden durch Standard-Software implementiert, die selbst wiederum Ausgangspunkt für einen Angriff sein kann. In den nächsten Abschnitten werden die beim Bau des Systems verwendeten Software-Pakete und Hardware-Bestandteile kurz vorgestellt und auf ihre Sicherheit hin untersucht.

12.4 Eingesetzte Software

12.4.1 Java und JDBC

Die von SUN entwickelte Programmiersprache Java ist eine Kombination aus Compiler- und Interpretersprache. Bei dieser Sprache wird der Quellcode zunächst in ein maschinenunabhängiges Format, den *Bytecode* übersetzt. Der Bytecode kann dann ohne Modifikation auf allen Betriebssystemen ausgeführt werden, die einen entsprechenden Interpreter, die *Java Virtual Machine* (JVM) bereitstellen. Java ist daher für den Einsatz im Internet, einem Netz mit einer extrem inhomogenen Rechnerplattform, besonders geeignet. Darüber hinaus ist durch die Java Database Connectivity (JDBC) ein direkter Zugriff auf die Inhalte von Datenbanken aus Java-Programmen heraus möglich, wodurch die Einsatzmöglichkeiten im Internet stark erweitert werden.

Java unterstützt kryptographische Anwendungen durch eine eigens zu diesem Zweck entworfene Architektur, die *Java Cryptography Architecture* (JCA). Diese Architektur stellt einfache Sicherheitsfunktionen bereit, wie z.B. Einweg-Hashverfahren und digitale Signaturen, die über ein sogenanntes *Provider-Modul* eingebunden werden. Mit der aktuellen Version 2 der Programmiersprache liefert SUN einen Provider aus, der die Hashfunktionen MD5 und SHA-1 sowie DSA zur digitalen Signatur und Mechanismen zur Verarbeitung von Zertifikaten nach dem X.509 Standard unterstützt. Aufgrund der US-Exportbeschränkungen für starke Kryptographie werden Algorithmen zur starken symmetrischen Verschlüsselung, z.B. DES oder IDEA, an dieser Stelle nicht unterstützt.

Es existieren mehrere Möglichkeiten, dennoch Verschlüsselung in Java-Anwendungen zu implementieren. Innerhalb der USA kommt die *Java Cryptography Extension* (JCE) der Firma SUN zum Einsatz, mit der die JCA um einen Provider zur symmetrischen (z.B. DES, Triple-DES, RC4 u.a.) und asymmetrischen Verschlüsselung (z.B. RSA) erweitert wird. Aufgrund der verwendeten Mechanismen darf der JCE-Provider jedoch nicht aus den USA exportiert werden.

Weder die JCA noch die JCE und ihre Nachbauten unterstützen direkt die Verschlüsselung. Sie bilden zunächst nur eine Schnittstelle zwischen Anwendung und Provider Modul, die der Anwendung erlauben, bestimmte Algorithmen aus einem oder mehreren installierten Provider-Modulen aufzurufen. Durch diesen modularen Aufbau können auch Provider eingesetzt werden, die nicht den US-Exportbestimmungen unterliegen, da sie in anderen Ländern entwickelt wurden. Aus Sicherheitsgründen sollten nur Module eingesetzt werden, deren vollständiger Quellcode vorliegt, z.B. der Provider der Australian Business Access Ltd. (ABA). Er wurde ab 1995 mit dem Ziel entwickelt, Firmen und Institutionen im Bereich E-Commerce zu unterstützen.

Die im Modell für die Bonner Wortdatenbank geforderten Eigenschaften können durch den Einsatz der JCA/JCE-Architekturen und geeigneter Provider Module erfüllt werden. Unterstützt werden ein KeyPairGenerator für RSA und DSA sowie die zugehörigen Mechanismen zur Signatur, symmetrische Verschlüsselung zur Einrichtung von Kanälen zur vertraulichen Übertragung von Daten und unterstützende Verfahren, z.B. sichere Hashfunktionen und Zufallszahlengeneratoren. Diese Verfahren können auch eingesetzt werden, um Java-Programme selbst zu schützen.

Zusätzlicher Schutz des Java-Programmcodes kann überall da nötig werden, wo ein Angreifer durch Analyse oder Manipulation des Codes Vorteile

erlangen kann. Zwar müssen die eingesetzten Verfahren der digitalen Signatur und Verschlüsselung so konstruiert sein, daß ihre Sicherheit nicht auf der Geheimhaltung des Verfahrens, sondern nur auf der Geheimhaltung der verwendeten Schlüssel beruht. Dennoch gibt es Stellen im Gesamtmodell, an denen eine zusätzliche Absicherung des ausführbaren Codes sinnvoll ist.

Grundsätzlich ist es ohne Probleme möglich, Java-Bytecode zu analysieren und lesbaren Quelltext aus der Analyse abzuleiten. Deshalb kann man davon ausgehen, daß ein Angreifer mit geringem Aufwand detaillierte Kenntnisse über den gesamten Ablauf der Verschlüsselung, sowohl der symmetrischen als auch der asymmetrischen Verfahren, erhalten kann (vgl. [Knudsen 1998] S.248ff). Diese Kenntnisse können jedoch nur dann genutzt werden, wenn schwerwiegende Implementierungsfehler entdeckt werden, z.B. die Verwendung einer sich nicht verändernden Zeichenfolge an Stelle eines echten, zufällig gewählten *session key*, wodurch die Ergebnisse der Verschlüsselung vorhersehbar und reproduzierbar werden.

Die Analyse von Java-Bytecode kann zwar nicht verhindert, jedoch erschwert werden. Eine Möglichkeit besteht im Einsatz von Software, die alle Informationen aus dem Bytecode entfernt, die nicht unmittelbar zur Ausführung benötigt werden oder den Bytecode so verändert, daß der tatsächliche Programmablauf nicht mehr erkennbar ist. Diese als *bytecode obfuscator* bezeichnete Software sorgt dafür, daß eine Analyse stark verzögert wird und weniger Informationen liefert.

Eine bessere Absicherung ergibt sich durch die Verschlüsselung großer Teile des ausführbaren Codes durch einen symmetrischen Algorithmus. Ein Programm besteht dann aus einem einfachen Modul zum Nachladen von Bytecode und dessen Entschlüsselung zur Laufzeit. Der große Nachteil dieser Methode liegt darin, daß ein weiterer Schlüssel sicher gespeichert

werden muß, z.B. auf der Chipkarte, so daß ein Zugriff nur nach Eingabe der PIN möglich ist.

Muß die PIN an dieser Stelle oder bei Durchführung der Signatur über die Tastatur des Rechners eingegeben werden, weil der Kartenleser nicht über einen eigenen Ziffernblock verfügt, bleibt die Absicherung des Programmcodes nutzlos, da ein Angreifer direkt auf Ebene des Betriebssystems Modifikationen vornehmen und so in Kenntnis der PIN gelangen kann.

Auch zusätzliche Funktionen der JCA/JCE, wie z.B. das *Object Signing*, ein Mechanismus, der sicherstellt, daß ein Java-Object nicht unbemerkt verändert wird, können an dieser Stelle nicht sinnvoll eingesetzt werden, da der Angriff nicht auf die Java VM oder den ausführbaren Bytecode gerichtet ist.

Die vollständige Verlagerung der Programmausführung auf eine Chipkarte, z.B. die *JavaCard* der Firma Schlumberger, scheitert zur Zeit noch an der zu niedrigen Prozessorleistung von Chipkarten und den hohen Zeitverlusten bei der Übertragung von Informationen von und zur Karte. Eine Auslagerung besonders sicherheitsrelevanter Komponenten, z.B. der Verfahren zur symmetrischen Verschlüsselung, ist denkbar, würde jedoch die Antwortzeiten des Gesamtsystems deutlich herabsetzen. Die Chipkarte wird daher im vorliegenden Modell nur zur sicheren Speicherung des Signaturschlüssels eingesetzt. Die entsprechenden Mechanismen werden im Detail im nächsten Abschnitt beschrieben.

12.4.2 Opencard Framework und Chipkarte

Der Zugriff auf die Inhalte der Chipkarte erfolgt über das bereits angesprochene *OpenCard Framework* (OCF). Dieses Interface erlaubt den einheitlichen Zugriff auf Kartenterminals verschiedener Hersteller, die entweder direkt, d.h. durch einen eigenen Gerätetreiber, oder indirekt über die PC/SC-Schnittstelle unterstützt werden. Die zur Kommunikation mit der Chipkarte notwendigen Befehle, die *application protocol data units* (APDU), werden über diese Schnittstelle aus Java-Programmcode erzeugt, so daß die einzelnen Funktionen der Karte über eine einheitliche Plattform, die *Card Services* angesprochen werden können.

Das OpenCard Framework stellt verschiedene Card Services bereit, darunter solche, die auf das Filesystem der Karte zugreifen (*File System Card Service*) oder Signaturen durchführen (*Signature Card Service*). Nicht jede Chipkarte ist in der Lage, alle diese Dienste auszuführen. Einige Karten eignen sich nur als geschützte Speicherkarten, andere stellen auch asymmetrische Verschlüsselungsverfahren bereit.

Bevor eine Chipkarte eingesetzt werden kann, muß sie die Prozesse der *Initialisierung* und *Personalisierung* durchlaufen. Hier wird zunächst die Verzeichnisstruktur der Karte mit allen Dateien festgelegt, die später interne Kartendaten, Anwendungsprogramme und Benutzerdaten aufnehmen werden. Die Struktur und die Zugriffsrechte auf einzelne Dateien und Anwendungen der Karte werden vom Entwickler festgelegt und in der Regel im Textformat gespeichert. Mit einem Initialisierungs-Toolkit wird dann diese Information in ein Format umgewandelt, welches mit dem Speicheraufbau der Chipkarte (dem EEPROM) kompatibel ist. Nachdem die Informationen auf die Karte übertragen worden sind, ist diese einsatzbereit.

Als Chipkarte für die Bonner Wortdatenbank wurde die *IBM Multifunktionskarte* (MFC) in der Versionen 4.0 verwendet, zur Initialisierung und Personalisierung dieser Karte der *Smart Card Toolkit* der Firma IBM. Für die MFC-Karte dieser Version liegt eine Exportgenehmigung für den Einsatz außerhalb der USA vor, so daß sie international eingesetzt werden kann. Die Durchführung von Signaturen mittels RSA mit 512 Bit Schlüssellänge durch die Karte ist möglich, ebenso der Import von extern generierten Schlüsseln. Die Schlüsselgenerierung auf der Karte wird jedoch erst von Karten der Version 4.21 unterstützt, die den Exportbeschränkungen unterliegen.

Für den Zugriff auf die IBM-Chipkarte wurden Kartenleser der Firmen Towitoko (Chipdrive) und Utimaco Safeware (Cardman) getestet. Die Konfiguration des OpenCard Framework für diese Terminals erfolgt über zwei Initialisierungsdateien, die im Zusammenhang mit der Definition des Layout und der sich daraus ergebenden Zugriffsrechte in Abschnitt 12.6.3 vorgestellt werden.

12.5 Implementierung

Das für die Bonner Wortdatenbank erstellte Modell enthält neben Standardkomponenten, z.B. für den Datenbankzugriff oder die Datenübertragung nach dem Client/Server-Prinzip, an einigen Stellen Komponenten, die sicherheitsrelevante Funktionen durchführen. Um die Anwendbarkeit des Modells zu demonstrieren, wurden genau diese kritischen Teilkomponenten in Java implementiert.

Die prototypische Implementierung umfaßt einerseits symmetrische und asymmetrische Verschlüsselungsverfahren, die zum Erreichen der Sachziele *Vertraulichkeit, Integrität* und *Authentizität* eingesetzt werden, andererseits auch die zum Bau einer sicheren Infrastruktur notwendigen Mechanismen der Speicherung von Signaturschlüsseln auf einer Chipkarte. Im folgenden

werden die realisierten Komponenten vorgestellt, der vollständige Java-Quellcode ist mit allen benötigten Bibliotheken auf der beigelegten CD-ROM enthalten (s. Anhang C).

12.5.1 Symmetrische Verfahren

Verfahren der symmetrischen Verschlüsselung werden im Modell eingesetzt, um die Vertraulichkeit der Kommunikation zwischen Client und Server zu gewährleisten. Bevor diese Verfahren angewandt werden können, muß das in Abschnitt 12.5.1 angesprochene Provider-Modul aktiviert werden. Der Zugriff auf die kryptographischen Funktionen des Providers erfolgt über eine Instanz der Java-Klasse *Cipher*. Hier werden der zu verwendende Algorithmus und die zugehörigen Parameter spezifiziert, z.B. DES im Modus CBC (vgl. Kapitel 3.2). Dieser Teil des Verfahrens ist in Abbildung 12.10 dargestellt.

```
// Provider hinzufügen
Security.addProvider (new ABAProvider());

// Initialisierung
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
```

Abbildung 12.10: Auszug aus *descrypt.java*

Die neu erzeugte Instanz des Cipher-Klasse wird nun für Ver- oder Entschlüsselung initialisiert und erhält einen zuvor generierten gültigen Schlüssel. Die Generierung eines *session key*, der zur Verschlüsselung verwendet werden kann, erfolgt durch eine Instanz der Klasse *KeyGenerator*. Im einfachsten Fall werden *session keys*, die nur für eine Sitzung gültig sind, mit Hilfe des sicheren Zufallszahlengenerators der Java-Virtual Machine (VM) erzeugt. In Abbildung 12.11 wird dargestellt, wie der

Generator mit einer neuen Zufallszahl initialisiert wird und einen Sitzungsschlüssel zum Einsatz im DES-Algorithmus liefert.

```
Key desKey;

// Generator initialisieren
KeyGenerator desGenerator =  KeyGenerator.getInstance("DES");
desGenerator.init(new SecureRandom());

// Schlüssel generieren
desKey = desGenerator.generateKey();
```

Abbildung 12.11: Generierung eines *session key*

Die Instanz der Cipher-Klasse erhält nun den neu erzeugten Schlüssel, und der Prozeß der Verschlüsselung wird fortgesetzt. Die Verarbeitung von Klartext kann über die Methoden *Cipher.update* oder *Cipher.doFinal* durchgeführt werden. Die erste Methode ist zur Verschlüsselung größerer Datenmengen geeignet, die mit dem Aufruf der zweiten Methode abgeschlossen wird. Kleine Datenmengen, z.B. einzelne Strings können auch direkt durch *Cipher.doFinal* bearbeitet werden, wie Abbildung 12.12 zeigt.

```
// Verschlüsselung initialisieren
cipher.init(Cipher.ENCRYPT_MODE, desKey);

// Verschlüsselung durchführen
byte[] stringBytes = secretString.getBytes("UTF8");
byte[] raw = cipher.doFinal (stringBytes);
```

Abbildung 12.12: Durchführung der symmetrischen Verschlüsselung

Der Inhalt der Zeichenkette *secretString* wird mit dem DES-Algorithmus im ECB-Modus verschlüsselt, das Ergebnis im Byte-Array *raw* abgelegt. Die so erzeugte Folge von Bytes kann durch Angabe des Modus *DECRYPT_MODE* während einer neuen Initialisierung wieder in Klartext umgewandelt werden.

Die Ausgabe des DES-Algorithmus liegt zunächst in einer Form vor, die sich nur bedingt zur Speicherung eignet, da sie alle Formen von Sonderzeichen enthalten kann. Vor der Abspeicherung sollte das Byte-Array daher aufgearbeitet werden. Dies kann z.B. durch Verwendung des Base64-Algorithmus geschehen, der Binärdaten in eine ASCII-Repräsentation übersetzt, die sich auch für eine Speicherung in einer Datenbank eignet.

Zur Verarbeitung kontinuierlicher Datenströme wurden die Cipher-Input/Output-Streams in Java eingeführt, die auf den allgemeinen Input/Output-Filterströmen beruhen. Hier wird zunächst ebenfalls eine Instanz der Klasse *Cipher* gebildet und initialisiert, dann jedoch direkt mit einem Datenstrom verbunden. Durch diese Technik können z.B. auch Datenströme zwischen Client und Server auf einfache Weise verschlüsselt werden.

Etwas komplexer als das hier angeführte Verfahren mit einem Schlüssel ist die asymmetrische Verschlüsselung, die im nächsten Abschnitt dargestellt wird.

12.5.2 Asymmetrische Verfahren

Die im Modell dargestellten Mechanismen zur Authentifizierung von berechtigten Benutzern basieren auf Verfahren der digitalen Signatur, d.h. auf asymmetrischer Verschlüsselung. Die grundlegende Vorgehensweise für digitale Signaturen, d.h. Verschlüsselungsverfahren mit einem privaten und einem öffentlichen Schlüssel, ist mit den oben dargestellten symmetrischen Verfahren identisch, bei Verwendung mehrerer Schlüsseln gibt es jedoch signifikante Unterschiede.

Zunächst wird statt des *KeyGenerator* ein *KeyPairGenerator* eingesetzt, der ein Schlüsselpaar erzeugt. Als wichtigstes Argument erhält dieser Generator eine Angabe über die Länge der zu erzeugenden Schlüssel, in Abbildung 12.13 z.B. 512 Bit nach dem RSA-Verfahren.

```
// Provider ABA dynamisch hinzufügen
Security.addProvider (new ABAProvider());

// RSA Instanz erzeugen
KeyPairGenerator generator = KeyPairGenerator.getInstance("RSA");

// Bitlänge für RSA festlegen
generator.initialize(512);

// Schlüsselpaar generieren
KeyPair rsakeys = generator.generateKeyPair();
```

Abbildung 12.13: Erzeugung eines RSA-Schlüsselpaares

Das neue Schlüsselpaar kann nun über die Routinen *getPrivate* und *getPublic* separiert werden, wobei der Zugriff auf den *private key* eingeschränkt sein kann, wenn das Schlüsselpaar auf der Chipkarte erzeugt wurde. In diesem Fall ist nur der öffentliche Teil auslesbar. Der private Schlüssel verläßt die Karte nicht.

Die Datentypen *public key* und *private key* gehören in Java zum komplexen Datentyp *key*, der vergleichbar mit der Ausgabe eines Verschlüsselungsalgorithmus ebenfalls zunächst als Bytefolge interpretiert wird. Eine Umwandlung zur Speicherung ist auf verschiedene Arten möglich, z.B. durch Verwendung der Java-Klassen *KeySpec* und *KeyFactory*, deren Einsatz jedoch einige zusätzliche Konvertierungen erfordert. Eine einfache Möglichkeit besteht in der Speicherung über einen *ObjectOutputStream*, wie in Abbildung 12.14 dargestellt. Diese Methode hat darüber hinaus auch den Vorteil, daß die exportierten Schlüssel durch den oben angeführten

CipherOutputStream gelenkt und so symmetrisch verschlüsselt abgelegt werden können.

```
// privaten Schlüssel aus Schlüsselpaar entnehmen
privateKey = rsakeys.getPrivate();

// privaten Schlüssel speichern

ObjectOutputStream outpriv = new ObjectOutputStream(
new FileOutputStream("private.key"));
outpriv.writeObject (privateKey);
outpriv.close();
```

Abbildung 12.14: Speicherung der einzelnen Komponenten

Ist das Problem der Generierung und Speicherung von Schlüsseln gelöst, kann die Signatur durchgeführt werden. Zunächst wird der gespeicherte Signaturschlüssel von Datenträgern oder Chipkarte eingelesen. Dann wird eine Instanz der Klasse *Signature* unter Angabe des zu verwendenden Verfahrens gebildet. In der Regel sind an dieser Stelle Hashfunktion und asymmetrischer Algorithmus zu spezifizieren, im Beispiel MD5 und RSA (vgl. Kapitel 3.3 und 3.6). Nun folgt die Initialisierung mit dem privaten Schlüssel, der zur Signatur verwendet werden soll. Die zu signierenden Daten werden über die Methode *update* bereitgestellt und dann durch *sign* abschließend bearbeitet (vgl. Abbildung 12.15).

```
// privaten Schlüssel einlesen
ObjectInputStream in = new ObjectInputStream (
new FileInputStream("private.key"));
privateKey = (PrivateKey)in.readObject();
in.close();

// Signatur initialisieren
Signature rsaSig = Signature.getInstance("MD5withRSA");
rsaSig.initSign(privateKey);

// Signatur durchführen
byte [] sigInput = sigData.getBytes();
rsaSig.update(sigInput);
byte[] signature = rsaSig.sign();
```

Abbildung 12.15: Digitale Signatur in Java

Das Ergebnis ist ein Byte-Array, welches die Signatur enthält und hinsichtlich der Speicherung mit den oben angeführten Mechanismen behandelt werden kann. Zur *Verifizierung* der Signatur wird der öffentliche Teil des Schlüssels benötigt, mit dem das Verfahren initialisiert wird. Analog zum Prozeß der Signatur wird die Methode *update* zur Aufbereitung der Signaturdaten verwendet, die dann mittels *verify* geprüft werden (vgl. Abbildung 12.16).

```
// Signatur initialisieren
Signature rsaSig = Signature.getInstance("MD5withRSA");
rsaSig.initVerify(publicKey);

// Signatur verifizieren
byte [] sigInput = sigData.getBytes();
rsaSig.update(sigInput);

if(rsaSig.verify(signature)) {
        // Signatur ok }
else {
        // Signatur nicht ok }
}
```

Abbildung 12.16: Signatur verifizieren

Abhängig vom Ergebnis der Methode *verify* wird ein Rückgabewert geliefert, der anzeigt, ob die zu prüfenden Daten mit dem korrekten Signaturschlüssel

behandelt wurden. An dieser Stelle ist der komplette Vorgang der Signatur und Verifizierung beendet (vgl. [Knudsen 1998] Kap 6).

Um starke Verschlüsselung mit mehr als 512 Bit Schlüssellänge nutzen zu können, wurde die Signatur auf dem Rechner des Benutzers und nicht auf der Chipkarte ausgeführt, die in diesem Szenario nur als sicheres Speichermedium verwendet wird. Dies zeigt der nächste Abschnitt.

12.5.3 Sichere Speicherung

Die IBM-Chipkarte MFC 4.0 dient zur sicheren Speicherung der verwendeten Signaturschlüssel. Bevor auf die Karte zugegriffen werden kann, muß die Konfiguration des Kartenterminals dem OpenCard Framework (OCF) bekannt gemacht und die Karte initialisiert und personalisiert werden.

Eine Anpassung des OCF wird durch die Dateien *readers.cfg* und *opencard.properties* vorgenommen. Die erste Konfigurationsdatei enthält Angaben über die verwendete Schnittstelle zur Kommunikation mit dem Kartenleser und die verwendete DLL (vgl. Abbildung 12.17). Die Einträge der zweiten Datei legen u.a. fest, ob der angeschlossene Kartenleser direkt oder über PC/SC unterstützt wird (vgl. Abbildung 12.18).

```
Reader[1]
        PORT_NUMBER = 1
        READER_NAME = TOWITOKO terminals
        DLL_NAME    = gen_twk.dll

Reader[2]
        PORT_NUMBER = A
        READER_NAME = Utimaco Terminal 1
        DLL_NAME    = gen_uti.dll
```

Abbildung 12.17: Datei *readers.cfg* zur Konfiguration des Kartenlesers

```
// Dienste der MFC-Karte ansprechen
```

```
OpenCard.services = MFCCardServiceFactory

// Eigenen Kartentreiber aktivieren
OpenCard.terminals = PcscMigCardTerminalFactory
#OpenCard.terminals = Pcsc10CardTerminalFactory
```

Abbildung 12.18: Auszug aus der Datei *opencard.properties*

Ist das OpenCard Framework für den Kartenzugriff vorbereitet, können personalisierte Chipkarten gelesen und beschrieben werden. Zur Personalisierung werden zunächst das *Layout* der Chipkarte und die verwendeten *Zugriffsrechte* spezifiziert. Das virtuelle Dateisystem einer Chipkarte besteht aus dem *Master File* (MF), der Wurzel des Systems, den darunter angebrachten *Dedicated Files* (DF), vergleichbar mit Verzeichniseinträgen eines herkömmlichen Filesystems und den *Elementary Files* (EF), den einzelnen, in Unterverzeichnissen abgelegten Dateien. Für alle Dateien wird die relative Lage im EEPROM spezifiziert, das Master File erhält immer die Identifikation 0x3f00, die Kennzeichnung anderer Dateien ist variabel.

Für alle Dateien werden neben der Kennzeichnung auch Länge und Zugriffrechte festgelegt sowie optional Inhalte, die bereits beim Prozeß der Personalisierung aufgebracht werden. *Zugriffsrechte* auf Dateien einer Chipkarte reichen von *NEVER* bis *ALWAYS* und können an die korrekte Eingabe einer PIN gebunden werden. Die einzelnen Zugriffsrechte werden, vergleichbar mit den Mechanismen der Dateiverwaltung von Betriebssystemen, für verschiedenen Operationen (z.B. *READ* und *WRITE)* angegeben. Abbildung 12.19 zeigt ein Beipiel für eine Datei mit der Identifikation 0x1000, deren Länge 100 Bytes beträgt. Diese Datei darf immer gelesen werden (*Read_Seek* ist auf *ALWAYS* gesetzt). Sie darf jedoch nur nach Eingabe einer PIN oder *Card Holder Verifikation* (CHV) mit neuen Werten beschrieben werden (Update über CHV2) und ist mit der Zeichenkette "*username:username@host.domain*" initialisiert.

```
File
(
  userdata
  Id (0x1000)
  (100)
  Access
  (
    Update(CHV2)
    Read_Seek(Always)
    Write(Never)
    Invalidate(Never)
    Rehabilitate(Never)
  )
  Initial_data
  (
    "username:username@host.domain", 13, 10
  )
  Agent_data
  (
    userdata length(100)
  )
)
```

Abbildung 12.19: Auszug der Initialisierungsdatei

Im Zusammenhang mit der Nutzung der MFC-Karte für die Bonner Wortdatenbank werden in dieser Datei die Daten eines regulären Benutzers gespeichert. Die Kombination aus Vor- und Nachname mit der E-Mail-Adresse soll sicherstellen, daß Benutzerdaten eindeutig sind. Darüber hinaus kann dieses Format auch in Zertifikaten verwendet werden.

Benutzerdaten werden einmalig durch die CA-Komponente bei Ausgabe der Karte aufgebracht und können vom Benutzer selbst nur gelesen, jedoch nicht verändert werden. Die zum Beschreiben erforderliche PIN (CHV-2) ist nur der CA bekannt.

Eine weitere wichtige Datei auf der Chipkarte (ID 0x3000, Länge 300 Byte) enthält den Signaturschlüssel des Anwenders, der durch die CA erzeugt und abgespeichert wurde und ebenfalls vor erneuten Schreibzugriffen (durch CHV-2) geschützt ist. Leserechte an dieser Datei sind durch die PIN des Benutzers (CHV-1) gesichert, um ein Auslesen des Schlüssels durch Unberechtigte zu verhindern. Mit Eingabe seiner PIN authentifiziert sich der

Benutzer gegenüber der Chipkarte, die daraufhin den Signaturschlüssel freigibt, der zur Authentifikation des Client gegenüber dem Wortdatenbank-Server eingesetzt wird.

Das vollständige Layout der Karte ist im Anhang aufgeführt. Zusätzliche Felder, z.B. zur Speicherung von Konfigurationsinformationen des Client, können noch auf die Chipkarte aufgebracht werden, da nur ca. 20 Prozent des verfügbaren Speicherplatzes im EEPROM bisher genutzt wurden. Nach Initialisierung und Personalisierung ist die MFC-Karte für den geschützt ablaufenden lesenden und schreibenden Zugriff bereit.

Der Kartenzugriff durch das OpenCard Framework erfolgt in mehreren Schritten. Zunächst wird das Framework initialisiert und ein Kartendienst, z.B. *der File Access Service* aktiviert. Nun werden die zur Ausführung des Dienstes benötigten Daten übergeben (z.B. Pfadangaben und Dateideskriptoren) und Funktionen innerhalb dieses Dienstes aufgerufen. Abbildung 12.20 verdeutlicht das Verfahren. Hier erfolgt der schreibende Zugriff auf die bei der Personalisierung erstellte Datei mit Benutzerinformationen (ID 0x1000).

```
static SmartCard mfccard = null;
static FileAccessCardService facs = null;
String userdata = null;

// OCF starten
mfccard.start();

try {

        // vollständigen Pfad angeben
        CardFilePath filepath = new CardFilePath(":1000");
        CardFile root = new CardFile(facs);
        CardFile file = new CardFile(root,filepath);

        // Dateizugriff einrichten
        DataOutputStream dos = new DataOutputStream(new CardFileOutputStream(file));
        if (userdata == null) userdata = "Nobody";
        byte[] carddata = new byte[file.getLength()];
        byte[] userbytes = userdata.getBytes();
        System.arraycopy(userbytes, 0, carddata, 0, userbytes.length);

        // Benutzerdaten schreiben
        dos.write(carddata, 0, carddata.length);
        dos.close();
        }
catch (Exception e) {
        System.out.println (e);
        }
return;
```

Abbildung 12.20: Auszug aus *card.java*

Da in den Zugriffsrechten für diese Datei die Information enthalten ist, daß für den schreibenden Zugriff eine PIN (CHV-2) erforderlich ist, wird der Benutzer automatisch zur Eingabe der PIN durch das OpenCard Framework aufgefordert, wie in Abbildung 12.21 dargestellt.

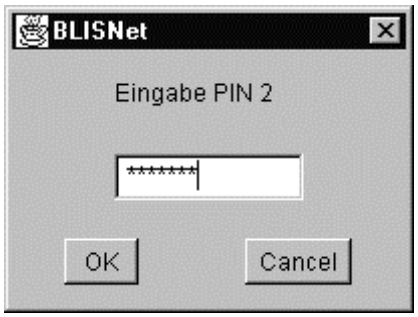


Abbildung 12.21: Eingabe der PIN-2 über OCF

Die vollständige Implementierung des Zugriffs auf die Chipkarte befindet sich auf der beigelegten CD-ROM, zusammen mit den in den vorigen Abschnitten beschriebenen Komponenten zur Verschlüsselung und digitalen Signatur.

12.6 Integration in die vorhandene Architektur

In den vorherigen Abschnitten wurden verschiedene Maßnahmen zur Sicherung der Client/Server-Kommunikation zwischen Benutzerrechner und Wortdatenbank-Server diskutiert und als Komponenten eines Gesamtsystems dargestellt. Im Rahmen dieser Arbeit sind alle dargestellten Sicherheitsmechanismen des vorgeschlagenen Modells für die Bonner Wortdatenbank realisiert worden.

Teilkomponenten bilden hier einerseits die symmetrische Verschlüsselung zum Schutz der Kommunikation zwischen Client und Server sowie zur sicheren Ablage des *public key* und andererseits die asymmetrische Verschlüsselung zur Authentifizierung des Client. Da der Signaturschlüssel des Client auf einer Chipkarte gespeichert wird, wurde das Layout dieser Karte auf den Einsatz mit der Wortdatenbank ausgelegt.

Die Integration dieser Sicherungsmechanismen in die bestehende BLISNet-Architektur kann auf verschiedene Art und Weise erfolgen. Welcher Ansatz realisiert werden soll, muß entschieden werden im Hinblick auf die rechtliche Situation bei Bereitstellung der BWDB-Inhalte im Internet. Hier müßen auch die Aspekte des kommerziellen Einsatzes der Wortdatenbank berücksichtigt werden (z.B. Kostenstrukturen) und die Auswahl zugriffsberechtigter Personen und Institutionen.

Grundsätzlich stehen mehrere Möglichkeiten der Integration zur Verfügung. Die vorgestellte Architektur ist zu beiden Versionen von BLISNet kompatibel, da die Transportmechanismen vollständig vom Benutzerinterface getrennt

wurden. So ist es möglich, einen in Java geschriebenen Client als Proxy auf dem lokalen System einzusetzen. Alle Anfragen, die von BLISNet-Clients der Versionen 1 und 2 erzeugt werden, gehen zunächst an diesen Proxy, der sich aus Sicht des Clients wie ein BLISNet-Server verhält, die Anfragen jedoch nicht selbst bearbeitet, sondern zur Bearbeitung an den tatsächlichen Server weiterleitet. Vor dem Verbindungsaufbau erfolgt die Authentifizierung der Benutzer mittels Signatur unter Verwendung der Chipkarte, dann der sichere Transport der Daten über das Internet zum Server.

Negativ wirkt sich bei diesem Konzept die aus der Datenkonvertierung entstehende Zeitverzögerung bei Verbindungsaufbau und Datentransport aus. Dem gegenüber steht jedoch der Vorteil der Wiederverwendung erprobter Komponenten. Zeitverluste könnten durch eine Lösung vermieden werden, bei der die vom BLISNet-Server gelieferten Inhalte der Wortdatenbank direkt durch den Java-Client dargestellt werden. Die Möglichkeiten der graphischen Darstellung mittels Java werden ständig erweitert und können zur Darstellung und Bearbeitung dieser Inhalte eingesetzt werden.

Durch die flexibele Integration der neuen Sicherheitsmechanismen in die bestehende Architektur ist eine Anpassung an neue Anforderungen jederzeit möglich.

13 Ausblick

Kommunikation im Internet ist ein vielschichtiges Phänomen mit vielen Aspekten, die einer eingehenden Untersuchung würdig sind. Die Zahl der Publikationen, die sich mit Internet-Kommunikation beschäftigen, nimmt stetig zu. In diesen Publikationen werden die im Internet transportierten Inhalte aus soziologischer, psychologischer und linguistischer Perspektive und aus vielen anderen Blickwinkeln betrachtet.

In einem anonymen Medium wie dem Internet stellen sich Probleme, die bei einer direkten Kommunikation gar nicht erst enstehen. Die große räumliche Distanz zwischen den Kommunikationspartnern und die im Gegensatz zu einer Face-Face Situation fehlenden Erkennungsmerkmale führen zu einem Verlust von Vertrauen in die Identität des Partners und die Sicherheit der übertragenen Inhalte.

Die vorliegende Arbeit hat das Internet daher zunächst als Medium zum Transport von Informationen betrachtet. Nicht der Inhalt dieser Informationen hat den Mittelpunkt der Untersuchung gebildet, sondern die Art und Weise, wie diese Informationen sicher durch das Internet von einem Kommunikationspartner zum anderen gelangen können.

Von sicherer Kommunikation wurde im Zusammenhang mit den Sachzielen Vertraulichkeit, Integrität und Authentizität gesprochen. Erst wenn diese Ziele erreicht werden können, ist eine vertrauenswürdige Kommunikation überhaupt erst möglich. Um eine technische Lösung für zur Erreichung der Sachziele entwickeln zu können, wurden zunächst Sicherheitsmängel bei der Übertragung von Informationen im Internet sowie auf den beteiligten Rechnersystemen offengelegt und bewertet. Anhand des OSI-Modells wurden die Schwachstellen einzelner Protokolle dargestellt und mögliche Angriffe erläutert. Eingebettet in eine Übersicht der Kommunikationsabläufe

im Internet wurde auf diese Weise die Problematik der sicheren Übertragung von Informationen über ein unsicheres Medium verdeutlicht.

Im nächsten Schritt wurden Möglichkeiten zur Erreichung der zuvor definierten Primärziele der sicheren Kommunikation dargestellt. Hier wurde gezeigt, welche Mechanismen angewendet werden können, um sichere Kommunikationskanäle zu implementieren und einzelne Systeme abzusichern. An vielen Stellen wurde deutlich, daß Rechner- und Netzwerksicherheit nicht isoliert voneinander betrachtet werden dürfen, sondern in ein übergreifendes Sicherheitskonzept eingebunden werden müssen.

Sicherheit in diesem Sinne ist nur durch starke Kryptographie zu erreichen. Sie bildet die Nahtstelle zwischen Rechner- und Netzwerksicherheit. Kryptographie ist mehr als nur die Wissenschaft vom Ver- und Entschlüsseln von Nachrichten. Moderne Kryptographie stellt Werkzeuge wie Einweg-Hashfunktionen und digitale Signaturen bereit, die angewendet werden können, um sichere Gesamtsysteme zu bauen. In diesen Systemen werden Benutzer, Rechner und Netzwerke als Elemente eingebunden.

Vor diesem Hintergrund wurde ein Modell der sicheren Kommunikation für die Bonner Wortdatenbank entwickelt, das diesen Überlegungen Rechnung trägt und die Sicherheit aller beteiligten Komponenten berücksichtigt. Mögliche Angriffe auf die entworfene Architektur wurden dargestellt und kritisch bewertet, unter Einbezug der in früheren Kapiteln dargestellten realen Szenarien.

Um die Anwendbarkeit des Modells für die Praxis darzulegen, wurden die sicherheitsrelevanten Funktionen des Zugriffs auf die Wortdatenbank in Java implementiert, darunter Verfahren der symmetrischen Verschlüsselung, der digitalen Signatur und des Zugriffs auf eine Chipkarte, deren Layout

ebenfalls im Detail spezifiziert wurde. Die so entstandene und im Bereich der Sicherheit bereits umgesetzte Architektur kann durch ihren modularen Aufbau an die zukünftigen Erfordernisse der sicheren Kommunikation mit der Bonner Wortdatenbank angepaßt werden.

Erstmals stehen die technischen Mittel bereit, um eine sichere Kommunikation mit der Bonner Wortdatenbank zu gewährleisten. Eine Weiterentwicklung dieser Datenbank über das Internet ist damit möglich.

14 Literatur

[Assfalg/Goebels/Welter 1998]
Assfalg,R./Goebels,U./Welter,H.: Internet Datenbanken - Konzepte, Methoden, Werkzeuge. Bonn

[Baker/Hurst 1998]
Baker,S./Hurst,P.: The Limits of Trust. Cryptography, Governments, and Electronic Commerce. Den Haag.

[Bauer 1994]
Bauer,F.: Kryptologie. Methoden und Maximen. Berlin.

[Beutelspacher 1996]
Beutelspacher,A.: Kryptologie.Göttingen.

[Bitzer 2000]
Bitzer,F.: Digitale Signatur. Grundlagen, Funktion und Einsatz. Berlin

[Brobeil 1992]
Brobeil,H.: Softwareangriffe auf PCs und Netzwerke. München

[Brotz/Föckeler 1997]
Brotz,K./Föckeler,P.: Security unter Windows NT 4.0. Wien

[Brustkern 1992]
Brustkern,J.: Maschinenlesbare Lexika für die maschinelle Sprachverarbeitung - Repräsentation und Wiederverwendung. Holos - Reihe Linguistik Band 3. Bonn

[Burkart 1998]
Burkart,R.: Kommunikationswissenschaft. Grundlagen und Problemfelder.Umrisse einer interdisziplinären Sozialwissenschaft. Wien.

[Burnett 2001]
Burnett,S.: Kryptographie. RSA Security's Official Guide. Hollywell

[Chapman/Zwicky 1995]
Chapman,B./Zwicky,E.: Building Internet Firewalls. Sebastopol

[Cheswick/Bellovin 1996]
Cheswick,W./Bellovin,S.: Firewalls und Sicherheit im Internet. Reading, Massachusetts

[Comer/Stevens 1993]
Comer,D./Stevens,D.: Internetworking with TCP/IP Volume III. Client-Server Programming and Applications. BSD Socket Version. New Jersey

[Damgard 2000]
Damgard,I.: Lectures on Data Security. Modern Cryptology in Theory and Practice. IN: Lecture Notes in Computer Science, Vol. 1561

[Damker/Federrath/Schneider 1996]
Damker,H./Federrath,H./Schneider, M.: Maskerade-Angriffe im Internet. IN: Datenschutz und Datensicherung DuD 20/5

[Deikman 1989]
Deikman,A.: UNIX Programming on the 80286/80386. Davenport

[Dicken 1997]
Dicken,H.: JDBC - Internet-Datenbankanbindung mit Java. Bonn/ Albany

[Dittmann 2001]
Dittmann,J.: Digitale Wasserzeichen. Grundlagen, Verfahren, Anwendungsgebiete. Berlin

[Doberenz 1996]
Doberenz,W.: Java. München

[Eckert 2001]
Eckert,C.: IT- Sicherheit. Konzepte, Verfahren, Protokolle. München

[Ellison/Schneier 2000]
Ellison,C./Schneier,B.: Ten Risks of PKI. What You're not Being Told aubout Public Key Infrastructure. IN: Computer Security Journal, Vol. 16/2000

[Elsen 1996]
Elsen,H.: Multifunktionale lexikalische Informationssysteme am Beispiel der Bonner Wortdatenbank. Unveröffentlichte Magisterarbeit am IKP. Bonn

[Elsen/Hartmann 1993]
Elsen,H./Hartmann,J.: Satzbandanalyse und Aufbereitung des Definitions-wortschatzes eines deutschen Wörterbuchs. In: Pütz, H./Haller, J. (Hg.): Sprachtechnologie: Methoden, Werkzeuge, Perspektiven. Reihe Sprache und Computer Band 13. Hildesheim

[Faust 1991]
Faust,H.: Datenschutz und Arbeitsplatzrechner. München

[Flanagan 1997]
Flanagan,D.: Java in a Nutshell - A Desktop Quick Reference. Sebastopol

[Flanagan 1996]
Flanagan,D.: JavaScript - The definitive Guide. Sebastopol

[Fries 1993]
Fries,O./FritschA./Kessler,V./Klein,B. (Hg.): Sicherheitsmechanismen Bausteine zur Entwicklung sicherer Systeme. München/Wien

[Fröse 1994]
Fröse,J.: Effiziente Systementwicklung mit Oracle 7 - Ein Handbuch für die Praxis des Anwendungsentwicklers. Bonn/Paris/Reading, Massachusetts

[Fumy 1997a]
Fumy, W.: Key Management Techniques. IN: Preenel, B./ Rijmen, V. (Ed): State of the Art in Applied Cryptography. Berlin.

[Fumy 1997b]
Fumy, W.:Internet Security Protocols. IN: Preenel, B./ Rijmen, V. (Ed): State of the Art in Applied Cryptography. Berlin.

[Fumy/Horster/Kraaibeek 1995]
Fumy,W./Horster,P./Kraaibeek,P.: Standards und Patente zur IT-Sicherheit. München

[Fumy/Rieß 1994]
Fumy,W./Rieß,H.: Kryptographie - Entwurf, Einsatz und Analyse symmetrischer Kryptoverfahren. München/Wien

[Fuhrberg 1998]
Fuhrberg,K.: Internet-Sicherheit. Browser, Firewalls und Verschlüsselung. München

[Garfinkel/Spafford 1994]
Garfinkel,S./Spafford,G.: Practical Unix Security. Sebastopol

[Geis 2000]
Geis,I.: Die digitale Signatur - eine Sicherheitstechnik für die Informationsgesellschaft. Ein Leitfaden für Anwender und Entscheider. Hamburg

[Gerhardt 1993]
Gerhardt,W.: Zugriffskontrolle bei Datenbanken. München

[Gurry/Corrigan 1996]
Gurry,M./Corrigan,P.: Oracle Performance Tuning. Second Edition. London

[Halabi 1998]
Halabi,B.: Internet Routing-Architekturen - Grundlagen, Design und Implementierung. München

[Hartmann 1995]
Hartmann,J.: BLISNet - Lexikalisches Wissen im Netzwerk. In: Hitzenberger, L. (Hg.): Angewandte Computerlinguistik. Reihe Sprache und Computer Band 15. Hildesheim

[Hartmann 1997]
Hartmann,J.: BLISNet - Die Bonner Wortdatenbank im Internet. Unveröffentlichte Magisterarbeit am IKP. Bonn

[Heilmann 2000]
Heilmann,H.: Security Management. Heidelberg

[Hein 1997]
Hein,M.: TCP/IP - Internet Protokolle im professionellen Einsatz. 3. Aufl. Bonn

[Herda/Mund/Steinacker 1993]
Herda,S./Mund,S./Steinacker,A.: Szenarien zur Sicherheit informationstechnischer Systeme. München/Wien

[Heß/Brustkern/Lenders 1983]
Heß,K./Brustkern,J./Lenders,W.: Maschinenlesbare deutsche Wörterbücher. Dokumentation, Vergleich, Integration. Tübingen

[Hochmann 2001]
Hochmann,S.: Elektronische Signatur. Norderstedt

[Hornung 1990]
Hornung,E.: Entwurf und Implementierung einer betriebssystemunabhängigen Benutzeroberfläche für die Verwaltung lexikalischer Informationen. In: Brustkern, J. (Hg.): IKP Arbeitsbericht Nr.8. Bonn

[Horster 2001]
Horster,P.: Systemsicherheit. Grundlagen, Konzepte, Realisierungen, Anwendungen. Wiesbaden

[Hunt 1995]
Hunt,C.: TCP/IP Network Administration. London

[Hunt 1996]
Hunt,C.: Networking Personal Computers with TCP/IP. London

[Imai 2000]
Imai,H.: Public Key Cryptography. IN: Lecture Notes in Computer Science, Vol. 1562

[Joyner 2000]
Joyner, D.: Coding Theory and Cryptography. From Enigma and Geheimschreiber to Quantum Theory. Berlin

[Juchem 1988]
Juchem,J.: Kommunikation und Vertrauen. Ein Beitrag zum Problem der Reflexivität in der Ethnomethodologie. Aachen.

[Jumes/Cooper 1999]
Jumes,J./Cooper,N.: Microsoft Windows NT 4.0 - Sicherheit Überwachung, Steuerung. Washington

[Jungnickel 1995]
Jungnickel,D.: Codierungstheorie. Berlin

[Kandler/Winter 1992]
Kandler,G./Winter,S.: Wortanalytisches Wörterbuch - Deutscher Wortschatz nach Sinn-Elementen. München

[Kelly/Pohl 1990]
Kelly,A./Pohl,I.: A Book on C - Programming in C. Second Edition. Redwood City

[Kernighan/Ritchie 1990]
Kernighan,B./Ritchie,D.: The C Programming Language, Second Edition, Ansi C. New Jersey

[Kersten 1991]
Kersten,H.: Einführung in die Computersicherheit. Wien

[Kruglinski 1994]
Kruglinski,D.: Inside Visual C++. Second Edition. Washington

[Kuhlen 1999]
Kuhlen,R.: Die Konsequenzen der Informationsassistenten. Was bedeutet informationelle Autonomie oder wie kann Vertrauen in elektronische Dienste in offenen Informationsmärkten gesichert werden? Frankfurt 1999

[Kuppinger 1996]
Kuppinger,M.: Der Microsoft Internet Information Server. München

[Kyas 1998]
Kyas,O.: Sicherheit im Internet. Bonn

[Lai/Massey 1991]
Lai,X./Massey,J.: Proposal for a new Block Encryption Standard. In: Eurocrypt ’90 Proceedings. Berlin

[Langenbach 2001]
Langenbach,C.: Elektronische Signaturen. Kulturelle Rahmenbedingungen einer technischen Entwicklung. IN: Wissenschaftsethik und Technikfolgenbeurteilung. Schriftenreihe der Europäischen Akademie zur Erforschung von Folgen wissenschaftlich-technischer Entwicklungen. Band 12. Bad Neuenahr

[Lenders/Willee 1986]
Lenders,W./Willee,G.: Linguistische Datenverarbeitung - Ein Lehrbuch. Opladen

[Lenders 1993]
Lenders,W.: Tagging - Formen und Tools. In: Pütz, H./Haller, J. (Hg.): Sprachtechnologie: Methoden, Werkzeuge, Perspektiven. Reihe Sprache und Computer Band 13. Hildesheim

[Ligon 1996]
Ligon,T.: Client/Server Communications Services. A Guide for Applications Developer. New York

[Lücke/Neßler/Schettl 1997]
Lücke,B./Neßler,B./Schettl,D.: Borland IntraBuilder 1.5 - Intranet-Anwendungen effizient entwickeln. Bonn

[Maletzke 1998]
Maletzke,G.: Kommunikationswissenschaft im Überblick. Opladen

[Mansfeld 1997]
Mansfeld,G.: Windows NT 4.0 Referenz. Washington

[Meckel/Kriener 1996]
Meckel,M./Kriener,M.(Hg.): Internationale Kommunikation. Eine Einführung. Opladen

[Messelken 1997]
Messelken,D.: Semantische Strukturen in Wörterbüchern. Dargestellt am Beispiel der Bonner Wortdatenbank. Unveröffentlichte Magisterarbeit am IKP. Bonn

[Mintert 1997]
Mintert,S.: JavaScript 1.2. Einführung, Referenz, Praxislösungen. Bonn/Reading, Massachusetts

[Musciano/Kennedy 1996]
Musciano,C./Kennedy,B.: HTML - The definitive Guide. Cambridge

[Nusser 1998]
Nusser,S.: Sicherheitskonzepte im WWW. Berlin

[Okuda/Okuda/Mirek 1997]
Okuda,M./Okuda,D./Mirek,D.: Star Trek Encyclopedia - A Reference Guide to the Future. New York

[Ousterhout 1994]
Ousterhout,J.: Tcl and the Tk Toolkit. Reading, Massachusetts

[Papaj/Burleson 1997]
Papaj,R./Burleson,D.: Oracle Databases on the Web. Scottsdale, Arizona

[Perlman 1992]
Perlman,R.: Bridges and Routers in OSI and TCP/IP. Reihe Professional Computing. Reading, Massachusetts

[Pfitzmann/Rannenberg 1993]
Pfitzmann, A./Rannenberg,K.: Staatliche Initiativen und Dokumente zur IT-Sicherheit. Eine kritische Würdigung IN: Computer und Recht Vol. 9/3

[Pieprzyk 1999]
Pieprzyk,J.: Information Security and Privacy. 4th Australasian Conference, ACISP'99 Proceedings. IN: Lecture Notes in Computer Science, Vol. 1587

[Pohl/Weck 1993]
Pohl,H./Weck,G. (Hg.): Einführung in die Informationssicherheit. Wien

[Pohl/Weck 1995]
Pohl,H./Weck,G.: Handbuch 2 - Managementaufgaben im Bereich der Informationssicherheit.München

[Prommer/Vowe 1998]
Prommer,E./Vowe,G.: Computervermittelte Kommunikation. Öffentlichkeit im Wandel.Konstanz

[Rankl/Effing 1996]
Rankl,W./Effing,W.: Handbuch der Chipkarten. Aufbau, Funktionsweise, Einsatz von Smart-Cards. München/Wien

[Raepple 1998]
Raepple,M.: Sicherheitskonzepte für das Internet - Grundlagen, Technologien, und Lösungskonzepte für die kommerzielle Nutzung. Heidelberg

[Rodley 1996]
Rodley,J.: Writing Java Applets. Scottsdale, Arizona

[Rössler 1998]
Rössler,P.(Hg.): Online-Kommunikation. Beiträge zur Nutzung und Wirkung. Opladen

[Schimpf 1995]
Schimpf,A.: Client/Server-Konzepte. Haar b. München

[Schmeh 2001]
Schmeh,K.: Kryptografie und Public- Key Infrastrukturen im Internet. Heidelberg

[Schneier 1996]
Schneier,B.: Angewandte Kryptographie - Protokolle, Algorithmen und Sourcecode in C. Bonn

[Schneier 2001]
Schneier,B.: Secrets & Lies. Digital Security in a Networked World. New York

[Schumann 1997]
Schumann,M.: Der erfolgreiche Einstieg in Java. Frankfurt, Main/ Berlin

[Sedgewick 1992]
Sedgewick,R.:Algorithmen. Bonn/München/Reading, Massachusetts

[Selke 2000]
Selke,G.: Kryptographie. Verfahren, Ziele, Einsatzmöglichkeiten. Bonn

[Sinha 1996]
Sinha,A.: Network Programmming in Windows NT. Reading, Massachusetts

[Sloane/Wyner 1993]
Sloane,N./Wyner,A. (Hg): Claude Elmwood Shannon - Collected Papers. New York

[Smith 1998]
Smith,R.: Internet Krptographie. Bonn

[Stainov 1997]
Stainov,R.: IPnG - Das Internet-Protokoll der nächsten Generation. Bonn

[Stallings 1995]
Stallings,W.: Datensicherheit mit PGP. München

[Stallings 2000]
Stallings,W.: Sicherheit im Internet . Anwendungen und Standards. Reading

[Staubach 1989]
Staubach,G.: UNIX-Werkzeuge zur Textmusterverarbeitung. Awk, Lex und Yacc. Reihe Informationstechnik und Datenverarbeitung. Heidelberg

[Stevens 1990]
Stevens,W.: UNIX Network Programming. London

[Tanenbaum 1996]
Tanenbaum,A.: Computernetzwerke. 3.Aufl. New York

[Tinnefeld/Philipps/Weis 1993]
Tinnefeld,M.-T./Philipps,L./Weis,K. (Hg.): Die dunkle Seite des Chips-Herrschaft und Beherschbarkeit neuer Technologien. München

[Tinnefeld/Philipps/Weis 1994]
Tinnefeld,M.-T./Philipps,L./Weis,K. (Hg.): Institutionen und Einzelne im Zeitalter der Informationstechnik. Wien

[Washburn/Evans 1993]
Washburn,K./Evans,J.: TCPIP - Running a Successful Network. Reading, Massachusetts

[Welch 1995]
Welch,B.: Practical Programming in Tcl and Tk. New Jersey

[Wobst 1997]
Wobst,R.: Abenteuer Kryptologie - Methoden, Risiken und Nutzen der Datenverschlüsselung. Reading, Massachusetts/Bonn

[Zenk 1994]
Zenk,A.: Lokale Netze - Kommunikationsplattform der 90er Jahre. LAN-Betriebssysteme, Internetworking, Netzwerkmanagement. Bonn/Paris/Reading, Massachusetts

15 Anhang A - OpenCard Framework

15.1 Struktur der IBM-Chipkarten MFC 4.0/4.1x/4.2x

```
/*****************************************************************************/
/*  Layout der IBM-Chipkarte                                                 */
/*  © J.Hartmann 10/01                                                       */
/*****************************************************************************/

#include M4TFIXDC.INC
#include MFCFIXAR.INC

Declarations
(
  ICCDataT54
  (
    "T54",
    0,
    "DEIBM",
    0x12345678,
    0x99,
    0x00,
    0x00
  )
)

// ***************** ENVIRONMENT ****************
environment ( initial_only )
fixed_data_area ( fixed_data )
// ***************** MF-DIR *****************

Directory
(
  MF
  Id ( 0x3F00 )
  Access
  (
    Update(Never)
    Read_seek(Never)
    Create_delete(Never)
    Write(Never)
    Invalidate(Never)
    Rehabilitate(Never)
  )
  Asc_access ( No )
  Agent_data ( MF Select )
```

```
// ***************** CHV1 *****************

  File
  (
    EF_CHV1
    Id (0x0000)
    Access
    (
      Update(CHV2)
      Read_seek(Never)
      Write(Never)
      Invalidate(Never)
      Rehabilitate(Never)
    )
    Access_status
    (
     //   CHV1_change_not_allowed
    )
    Initial_data
    (
      0x01,
      0x00,
      0x00
    )
    Initial_data
    (
      0x31,                    // "1234"
      0x32,
      0x33,
      0x34,
      0x00,
      0x00,
      0x00,
      0x00
    )
    Initial_data
    (
      0x03,                    // 3 attempts
      0x03
    )
```

```
    Initial_data
    (
      0x00,
      0x00,
      0x00,
      0x00,
      0x00,
      0x00,
      0x00,
      0x00
    )
    Initial_data
    (
      0x00,
      0x00
    )
    Agent_data ( CHV01 length (23) )
  )

// ***************** CHV2 *****************

  File
  (
    EF_CHV2
    Id (0x0100)
    Access
    (
      Update(Never)
      Read_seek(Never)
      Write(Never)
      Invalidate(Never)
      Rehabilitate(Never)
    )
    Access_status
    (
     CHV2_change_not_allowed
    )
    Initial_data
    (
      0x01,
      0x00,
      0x00
    )
    // Personal_data(8, Pers_id, MAC_ENC) or
    Initial_data
    (
      ********                  // CHV2
    )
```

```
    Initial_data
    (
      0x0A,                  // 10 attempts
      0x0A
    )
    Initial_data
    (
      0x00,
      0x00,
      0x00,
      0x00,
      0x00,
      0x00,
      0x00,
      0x00
    )
    Initial_data
    (
      0x00,
      0x00
    )
    Agent_data ( CHV02 length (23) )
  )

  File
  (
    userdata
    Id (0x1000)
    (100)
    Access
    (
      Update(CHV2)
      Read_seek(Always)
      Write(Never)
      Invalidate(Never)
      Rehabilitate(Never)
    )
    Initial_data
    (
      "username:username@host.domain", 13, 10
    )
    Agent_data
    (
      userdata length(100)
    )
  )
```

```
// ***************** USER MD5 *****************

  File
  (
    cardholder_md5
    Id (0x2000)
    (16)
    Access
    (
      Update(CHV2)
      Read_seek(Always)
      Write(Never)
      Invalidate(Never)
      Rehabilitate(Never)
    )

    Agent_data
    (
      cardholder_md5 length(16)
    )
  )

// ***************** PRIVATE KEY *****************

  File
  (
    private_key
    Id (0x3000)
    (300)
    Access
    (
      Update(CHV2)
      Read_seek(CHV1)
      Write(Never)
      Invalidate(Never)
      Rehabilitate(Never)
    )
    Agent_data
    (
      private_key length(300)
    )
  )
```

```
// ***************** MD5 HASH ****************

  File
  (
    md5_hash
    Id (0x4000)
    (16)
    Access
    (
      Update(CHV2)
      Read_seek(Always)
      Write(Never)
      Invalidate(Never)
      Rehabilitate(Never)
    )
    Agent_data
    (
      md5_hash length(16)
    )
  )

 ) // End of MF-DIR


Statistics(3*16)
```

15.2 Opencard.properties

```
# Copyright © 1997, 1998 IBM Corporation.
# Redistribution and use in source (source code) and
binary (object code)
# forms, with or without modification, are permitted
provided that the
# following conditions are met:
# 1. Redistributed source code must retain the above
copyright notice, this
# list of conditions and the disclaimer below.
# 2. Redistributed object code must reproduce the above
copyright notice,
# this list of conditions and the disclaimer below in the
documentation
# and/or other materials provided with the distribution.
# 3. The name of IBM may not be used to endorse or
promote products
# derived from this software or in any other form without
specific prior written
# permission from IBM.
# 4. Redistribution of any modified code must be labeled
"Code derived from
# the original OpenCard Framework".
#
# THIS SOFTWARE IS PROVIDED BY IBM "AS IS" FREE OF
CHARGE.
# IBM SHALL NOT BE LIABLE FOR INFRINGEMENTS OF THIRD
PARTIES # RIGHTS BASED ON THIS SOFTWARE.  ANY EXPRESS OR
IMPLIED
# WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE ARE DISCLAIMED.  IBM DOES NOT
# WARRANT THAT THE FUNCTIONS CONTAINED IN THIS
# SOFTWARE WILL MEET THE USER'S REQUIREMENTS OR THAT THE
# OPERATION OF IT WILL BE UNINTERRUPTED OR ERROR-FREE.
IN
# NO EVENT, UNLESS REQUIRED BY APPLICABLE
# LAW, SHALL IBM BE LIABLE FOR ANY DIRECT, INDIRECT,
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO,
# PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
# USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
# WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
```

```
# USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY
# OF SUCH DAMAGE.  ALSO, IBM IS UNDER NO OBLIGATION TO
# MAINTAIN, CORRECT, UPDATE, CHANGE, MODIFY, OR OTHERWISE
# SUPPORT THIS SOFTWARE.
OpenCard.services =
com.ibm.opencard.factory.MFCCardServiceFactory
OpenCard.terminals =

com.ibm.opencard.terminal.pcsc10.Pcsc10CardTerminalFactor
y
# Configure ApplicationManagerFactories
OpenCard.application.applicationManagerFactories=
     com.ibm.zurich.smartcard.application.IBMApplicationM
anagerFactory
```

```
# Configure ApplicationManagers
# IBMApplicationManagerFactory supports a number of IBM
provided
# ApplicationManagers
# Naming schemes and application managers are associated
via tuples
#    < naming_scheme, class_name >
# where   naming_scheme  is an integer in the range of 0
to 65535
#    class_name     is the name of the application
manager class
#
OpenCard.application.IBMApplicationManagerFactory.registr
y=
     <16,com.ibm.zurich.smartcard.application.GenericSCTA
pplication
     Manager><17,com.ibm.zurich.smartcard.application.EMV
Application
     Manager>

# Configure naming schemes
# For the GenericSCTApplicationManager, naming schemes
consist of
# tuples       < naming_scheme, url_1, url_2, ..., url_n
>
# where naming_scheme    is an integer in the range of 0
to 65535
# url_n is a URL to retrieve the file containing the name
resolution info
#
OpenCard.application.GenericSCTApplicationManager.registr
y=
     <16,file://localhost/d:/OpenCard/src/opencard/test/l
ayout,
     http://lopagno/layout>
OpenCard.application.EMVApplicationManager.registry=
     <17,file://localhost/d:/OpenCard/src/opencard/test/l
ayout,
     http://lopagno/layout>

com.ibm.opencard.service.ServiceFactory.risky=true

########################
# TRACE configuration #
########################
OpenCard.trace = opencard:1 com.ibm:1 tests.system:1
```

15.3 Dialog zur Eingabe der PIN

```
// Eingabedialog für eine Chipkarten-PIN, abgeleitet aus OCF-Standard
// © J.Hartmann 10/01

import java.awt.Frame;
import java.awt.Dimension;

import opencard.core.service.CHVDialog;
import com.ibm.opencard.handler.IDDialog;

public class PINDialog implements CHVDialog
{

  public String getCHV(int chvNumber)
  {
    Frame pinframe  = new Frame("");
    pinframe.setVisible(false);

    IDDialog pindialog = new IDDialog(pinframe,
                              "BLISNet",
                              "Eingabe PIN ",
                              chvNumber);

    Dimension dim = pinframe.getToolkit().getScreenSize();
    pindialog.setSize(dim.width/5, dim.height/5);
    pindialog.setLocation(dim.width/2-dim.width/6, dim.height/2-dim.height/6);

    pindialog.show();
    pinframe.dispose();

    String pin = pindialog.pin();

    if ((pin != null) && (pin.length() == 0))  pin = null;

    return pin;
  }
}
```

15.4 Kartenzugriff

```
// Funktionen zum Zugriff auf IBM MFC 4.0 und 4.2x
// © J.Hartmann 09/01

import java.io.*;
import opencard.core.service.*;
import opencard.opt.iso.fs.*;
import opencard.core.terminal.*;
import opencard.core.util.OpenCardPropertyLoadingException;
import com.ibm.opencard.*;

public class card {

// OCF initialisieren

public card() {
        start();
        }

static SmartCard mfccard = null;
static FileAccessCardService facs = null;

// Benutzerdaten einlesen

public static String readuser() {
String userdata = null;

try {
        CardFilePath filepath = new CardFilePath(":1000");
        CardFile root = new CardFile(facs);
        CardFile file = new CardFile(root,filepath);
        DataInputStream dis = new DataInputStream(new
        CardFileInputStream(file));
         byte[] userbyte = new byte[file.getLength()];
        dis.read(userbyte);
        dis.close();
        userdata = new String(userbyte);
        userdata.trim();
        }
catch (Exception e) {
        System.out.println (e);
        }
return userdata;
}
```

```
// Benutzerdaten schreiben

public static void writeuser(String userdata) {

try {
        CardFilePath filepath = new CardFilePath(":1000");
        CardFile root = new CardFile(facs);
        CardFile file = new CardFile(root,filepath);
        DataOutputStream dos = new DataOutputStream(new
        CardFileOutputStream(file));
        if (userdata == null) userdata = "Nobody";
        byte[] carddata = new byte[file.getLength()];
        byte[] userbytes = userdata.getBytes();
        System.arraycopy(userbytes, 0, carddata, 0, userbytes.length);
        dos.write(carddata, 0, carddata.length);
         dos.close();
        }
catch (Exception e) {
        System.out.println (e);
        }
return;
}

// Auf eine Chipkarte warten

public static int getCard() {

try {
        CardRequest cr = new CardRequest ();
        cr.setWaitBehavior (CardRequest.NEWCARD);
        SmartCard mfccard = SmartCard.waitForCard(cr);
        facs = (FileAccessCardService)
        mfccard.getCardService(FileAccessCardService.class,true);
        PINDialog pindialog = new PINDialog();
        facs.setCHVDialog(pindialog);
        }

catch (Exception e) {
        System.out.println (e.getMessage ());
        return 1;
        }
return 0;
}
```

```
// OCF starten

public static void start() {
try {
        if(!SmartCard.isStarted()) SmartCard.start ();
        }

catch (Exception e) {
        System.out.println (e);
        }
}

// OCF beenden

public static void stop() {
try {
        SmartCard.shutdown ();
        }
catch (Exception e) {
        System.out.println (e);
        }
}
}
```

16 Anhang B - Verschlüsselungsmodule

16.1 DES

```
// Verschlüsselung mittels 56 Bit-DES-Algorithmus
// Ergebnis BASE64 codiert
// © J.Hartmann 10/01

import java.io.*;
import java.security.*;
import javax.crypto.*;
import au.net.aba.crypto.provider.ABAProvider;

public class descrypt {
        public static void main (String[] args) throws Exception {

        if (args.length < 1) {
        System.out.println("USAGE : descrypt string");
        return; }

        Security.addProvider (new ABAProvider());

        String sample = args[0];

        // Schlüssel generieren

        Key desKey;
        KeyGenerator desGenerator =  KeyGenerator.getInstance("DES");
        desGenerator.init(new SecureRandom());
        desKey = desGenerator.generateKey();
        System.out.println("- key generation ok");

        // Initialisierung
        Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

        // Verschlüsselung
        cipher.init(Cipher.ENCRYPT_MODE, desKey);
        byte[] stringBytes = sample.getBytes("UTF8");
        byte[] raw = cipher.doFinal (stringBytes);
        BASE64Encoder encoder = new BASE64Encoder();
        String base64 = encoder.encode(raw);
        System.out.println("- encrypted string " + args[0] + " is " + base64);
```

```
        // Entschlüsselung
        cipher.init(Cipher.DECRYPT_MODE, desKey);
        byte[] outputBytes = cipher.doFinal(raw);
        String result = new String(outputBytes, "UTF8");
        System.out.println("- decrypted string is " + result);
        }

}
```

16.2 RSA

```
// Erzeugung eines RSA Schlüsselpaares mit 512 Bit Länge
und
// Speicherung der Schlüssel

// © J.Hartmann 10/01

import java.io.*;
import java.security.*;
import javax.crypto.*;

// Security-Provider Klasse einfügen
import au.net.aba.crypto.provider.ABAProvider;

public class generator {
     public static void main (String[] args) throws
Exception {

     // Provider ABA dynamisch hinzufügen
     Security.addProvider (new ABAProvider());

     Key privateKey, publicKey ;

     // RSA Instanz erzeugen
     KeyPairGenerator generator =
KeyPairGenerator.getInstance("RSA");

     // Bitlaenge für RSA festlegen
     generator.initialize(512);

     // Schlüsselpaar generieren
     KeyPair rsakeys = generator.generateKeyPair();

     System.out.println("- key generation ok");

     // privaten Schlüssel aus Schlüsselpaar entnehmen
     privateKey = rsakeys.getPrivate();

     // privaten Schlüssel speichern
     ObjectOutputStream outpriv = new ObjectOutputStream(
     new FileOutputStream("private.key"));
     outpriv.writeObject (privateKey);
     outpriv.close();
     System.out.println("- private key in file
private.key");
```

```
     // öffentlichen Schlüssel aus Schlüsselpaar
entnehmen
     publicKey = rsakeys.getPublic();
```

```
      // öffentlichen Schlüssel speichern
      ObjectOutputStream outpub = new ObjectOutputStream(
      new FileOutputStream("public.key"));
      outpub.writeObject (publicKey);
      outpub.close();
      System.out.println("- public key in file
public.key");
      }
}
```

```
// Eingabestring signieren und Signatur speichern
// © J.Hartmann 11/01

import java.io.*;
import java.security.*;
import javax.crypto.*;

// Provider-Klasse einbinden
import au.net.aba.crypto.provider.ABAProvider;

public class sign {
      public static void main (String[] args) throws Exception {

      // Argumente prüfen
      if (args.length != 1) {
      System.out.println("USAGE : sign string");
      return; }

      // ABA-Provider dynamisch hinzufügen
      Security.addProvider (new ABAProvider());

      PrivateKey privateKey;

      // privaten Schlüssel einlesen
      ObjectInputStream in = new ObjectInputStream (new
      FileInputStream("private.key"));
      privateKey = (PrivateKey)in.readObject();
      in.close();

      // Signatur initialisieren
      Signature rsaSig = Signature.getInstance("MD5withRSA");
      rsaSig.initSign(privateKey);

      // Eingabestring signieren
      byte [] sigInput = args[0].getBytes();
      rsaSig.update(sigInput);
      byte[] signature = rsaSig.sign();
      System.out.println("- signature ok");
```

```
        // Signatur speichern
        ObjectOutputStream outsign = new ObjectOutputStream( new
        FileOutputStream("signature.dat"));
        outsign.writeObject (signature);
        outsign.close();
        System.out.println("- signature data saved in file signature.dat");
        }
}


// Signatur verifizieren
// © J.Hartmann 10/01

import java.io.*;
import java.security.*;
import javax.crypto.*;

// Provider-Klasse einbinden
import au.net.aba.crypto.provider.ABAProvider;

public class verify {
        public static void main (String[] args) throws Exception {

        // Argumente prüfen
        if (args.length != 1) {
        System.out.println("USAGE : sign string");
        return; }

        // ABA-Provider dynamisch hinzufügen
        Security.addProvider (new ABAProvider());

        PublicKey publicKey;
        byte[] signature;

        // öffentlichen Schlüssel einlesen
        ObjectInputStream inkey = new ObjectInputStream (new
        FileInputStream("public.key"));
        publicKey = (PublicKey)inkey.readObject();
        inkey.close();

        // gespeicherte Signatur einlesen
        ObjectInputStream insig = new ObjectInputStream (new
        FileInputStream("signature.dat"));
        signature = (byte[])insig.readObject();
        insig.close();
```

```
        // Signatur initialisieren
        Signature rsaSig = Signature.getInstance("MD5withRSA");
        rsaSig.initVerify(publicKey);

        // Signatur verifizieren
        byte [] sigInput = args[0].getBytes();
        rsaSig.update(sigInput);
        if(rsaSig.verify(signature)) {
                System.out.println("- signature ok"); }
        else {
                System.out.println("- signature NOT verified"); }
        }
}
```