Kapitel 3

Benutzersicherheit im Internet

3.1 Einführung

Diese Kurseinheit befasst sich mit dem Thema Sicherheit aus dem Blickwinkel des Webbenutzers. Die Schutzziele, die für den „normalen“ Websurfer wichtig sind, werden noch einmal zusammengestellt. Für die Anwendungen E-Mail (Abschnitt 3.2), Websurfen (Abschnitt 3.3) und Zugriff auf entfernte Rechner (Abschnitt 3.4) werden einzelne Lösungen vorgestellt. Danach gibt Abschnitt 3.5 noch einige Hinweise zur Sicherung eines privaten PCs unter dem Betriebssystem Windows.

3.2 Sichere E-Mail im Internet

Die Möglichkeit schnell, einfach und kostengünstig elektronische Nachrichten rund um die Welt schicken zu können, hat sicherlich einen großen Beitrag zur Popularität des World Wide Web geleistet. Allerdings sind E-Mails auch häufig die Ursache für Probleme.

Mit Hilfe von E-Mails verbreiten sich Computerviren extrem schnell. Falls die Schadensfunktion des Virus selbst wieder E-Mails verschickt, dann führt dies schnell dazu, dass die E-Mail-Server völlig überlastet sind. Am 4. Mai 2000 hat eine E-Mail mit dem Betreff „ILOVEYOU“ eine solche Kettenreaktion ausgelöst. Die E-Mail enthielt ein Skript in der Programmiersprache *Visual Basic* als Anhang. Wurde es gestartet, dann hat es sich selbst an alle Adressen im Adressbuch des E-Mail-Programms *Outlook* verschickt. Hatte ein Anwender sein E-Mail-Programm aus Bequemlichkeit so konfiguriert, dass Anhänge beim Lesen der E-Mail automatisch gestartet werden, dann hat sich das Virus auch automatisch im System eingenistet und an die Adressen weitergeleitet. Deshalb sollten Sie ihr E-Mail-Programm *immer* so konfigurieren, dass Anhänge *nicht* automatisch geöffnet werden!

Konfiguration der E-Mail-Software beachten

Wie in Abschnitt 1.3.4 gezeigt, werden elektronische Nachrichten meist unverschlüsselt vom Absender zum Empfänger übertragen. In vielen Fällen wünscht man sich als Benutzer allerdings, dass *nicht* jeder auf dem Weg liegende Rechner die Nachrichten mitlesen kann. Außerdem möchte man sicher sein, dass eine E-Mail tatsächlich vom angegebenen Absender stammt und unterwegs nicht verändert wurde.

In den folgenden Abschnitten werden einige Lösungen vorgestellt, mit denen man als Benutzer sicher per E-Mail kommunizieren kann.

3.2.1 Pretty Good Privacy (PGP)

Dieses Programm wurde von Phil Zimmermann entwickelt. Es enthält Algorithmen zur *Public-Key*-Verschlüsselung, zur Erstellung digitaler Signaturen, Hash-Algorithmen und verschiedene *Private-Key*-Verschlüsselungsalgorithmen. Alle in PGP benutzten Algorithmen sind bekannt und wurden von verschiedensten Forschern, bisher ohne Erfolg, auf Schwachstellen untersucht. Zusätzlich hat Phil Zimmermann den Quelltext und das Programm im Internet (für Privatpersonen) frei verfügbar gemacht. Somit kann sich jedermann von der Qualität der „inneren Werte“ des Programms ein eigenes Bild machen. Auch kann man damit das Programm auf versteckte Eigenschaften (Trojanische Pferde, siehe Abschnitt 1.4.3) hin untersuchen.

Neben der für Privatpersonen freien Version gibt es auch eine kommerzielle Version von PGP. Sie ist für Firmen gedacht, die Software ohnehin häufig nur zusammen mit Supportverträgen einsetzen. Die Firma *Symantec* aus Mountain View in Kalifornien vertreibt die kommerzielle Version von PGP unter dem Namen *Symantec Encryption Family*.

Neben diesen Implementierungen existiert auch eine komplett freie Implementierung. In RFC 2440 (OpenPGP-Message-Format) ist das Format von Nachrichten definiert, die „PGP-ähnliche“ Sicherheitsfunktionen erfüllen. Daneben ist *GNU-Privacy-Guard (GnuPG)* eine freie Implementierung der Funktionen aus dem RFC 2440. Auf Linux-Systemen findet man *GnuPG* häufig bereits vorinstalliert. Das Kommando heißt `gpg`. PGP bzw. OpenPGP bieten die folgenden Funktionen:

GnuPG

PGP Funktionen

- Verschlüsseln von Dateien, z. B. auf der Festplatte
- Sicheres Löschen (durch Überschreiben) von Dateien (nur PGP)
- Vertrauliche Kommunikation, indem der Sender die Nachricht mit dem öffentlichen Schlüssel des Empfängers verschlüsselt
- Authentische Kommunikation, indem der Sender die Nachricht mit seinem privaten Schlüssel digital signiert
- Erstellen von Schlüsselpaaren
- Management eigener und fremder Schlüssel

Von PGP existieren mehrere Versionen. Weit verbreitet sind die Version 2.6.3i, die Version 6.5.x und Versionen 8.x. Die Versionen unterscheiden sich dadurch, dass die neueren Versionen mehr Verschlüsselungsalgorithmen enthalten. Außerdem enthält die kommerzielle Version von PGP auch weitere patentierte Funktionen wie beispielsweise einen *Additional-Decryption-Key* oder *Cryptographic-Policy-Enforcement* sowie Funktionen, die Verschlüsselung mit zusätzlichem Benutzerkomfort verbinden, wie *Full-Disk-Encryption*. Details findet man in den *Technischen Berichten* (engl. **white paper**) der Firma *Symantec*.

PGP-Algorithmen

Auch von GnuPG gibt es zwei Versionen. Die „klassische“ Implementierung trägt die Versionsnummern 1.* und die „neue“ Implementierung `gpg2` die Versionsnummern 2.*. Neben veränderten Implementierungsdetails wurden in `gpg2` auch S/MIME-Fähigkeiten (siehe auch Abschnitt 3.2.2) hinzugefügt.

Die folgende Tabelle zeigt, welche Algorithmen in den jeweiligen Versionen implementiert sind:

	PGP 2.6.x	PGP 6.5.x	PGP 8.x	GnuPG
symmetrische Verschlüsselung	IDEA	IDEA CAST-128 3DES	IDEA CAST-128 3DES Twofish AES	 CAST5 3DES Twofish AES Blowfish
Hashfunktion	SHA-1	SHA-1	SHA-1 MD5 RIPEMD-160	SHA-1 MD5 RIPEMD-160 SHA-224, SHA-256 SHA-384, SHA-512
asymmetrische Verschlüsselung	RSA	RSA DSS Diffie-Hellman	RSA DSS Diffie-Hellman	RSA DSS ElGamal

PGP ist ein *hybrides* Verschlüsselungssystem. Die Nachrichten selbst werden mit einem symmetrischen Verschlüsselungsverfahren verschlüsselt. Der Grund ist, dass symmetrische Verfahren deutlich schneller ausgeführt werden können als asymmetrische Verfahren. Der Schlüssel des symmetrischen Verfahrens wird von PGP selbst berechnet bzw. generiert. Damit dieser Schlüssel nur ein einziges Mal erzeugt und benutzt wird, beruht seine Erstellung auf Zufallsereignissen (vergleiche Abschnitt 2.8.2). Der Benutzer muss dazu i. d. R. einige Tastatureingaben machen und die Tippgeschwindigkeit sowie die getippten Tasten werden als Zufallsquelle benutzt. Dieser Einmalschlüssel für die symmetrische Verschlüsselung wird auch **Session Key** genannt.

PGP-Verschlüsselung

Session Key

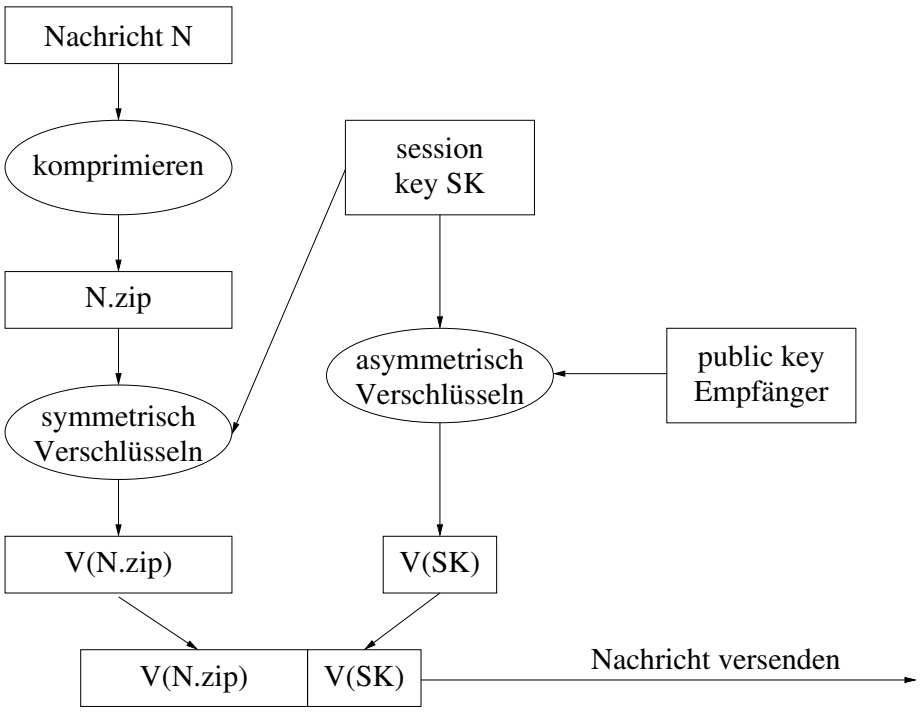


Abbildung 3.1: Ablauf bei der Verschlüsselung mit PGP

Die Verschlüsselung einer Nachricht besteht aus den in Abbildung 3.1 gezeigten Schritten.

1. Um Speicherplatz zu sparen, komprimiert PGP zunächst die Nachricht N. Dazu wird der **ZIP**-Algorithmus benutzt. Er basiert auf dem Verfahren von Lempel und Ziv. Eine C-Implementierung ist für nahezu alle Betriebssysteme verfügbar.

ZIP

2. Es wird ein Sitzungsschlüssel (engl. **session key**) *SK* generiert.

3. Die komprimierte Nachricht *N.zip* wird mit dem Sitzungsschlüssel verschlüsselt. Dazu wird einer der symmetrischen Algorithmen verwendet.

4. Der Sitzungsschlüssel wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Dazu wird einer der asymmetrischen Algorithmen benutzt.

5. Die verschlüsselte Nachricht und der verschlüsselte Sitzungsschlüssel werden aneinandergehängt (engl. **to concat**) und zusammen an den Empfänger geschickt.

Soll eine Nachricht an mehrere Empfänger geschickt werden, dann muss sie nicht für jeden Empfänger neu verschlüsselt werden. Stattdessen wird *nur* der Sitzungsschlüssel mit dem öffentlichen Schlüssel des weiteren Empfängers verschlüsselt und an die Nachricht angehängt. Eine Nachricht an mehrere Empfänger wird also nur unwesentlich größer, denn der verschlüsselte Sitzungsschlüssel braucht i. d. R. nur einen Bruchteil des Platzes der eigentlichen Nachricht.

Der Empfänger der Nachricht zerlegt sie in die beiden Teile und entschlüsselt den Sitzungsschlüssel mit seinem privaten Schlüssel. Mit dem so berechneten Sitzungsschlüssel entschlüsselt der Empfänger die komprimierte Nachricht. Zuletzt dekomprimiert der Empfänger die Nachricht und kann sie lesen.

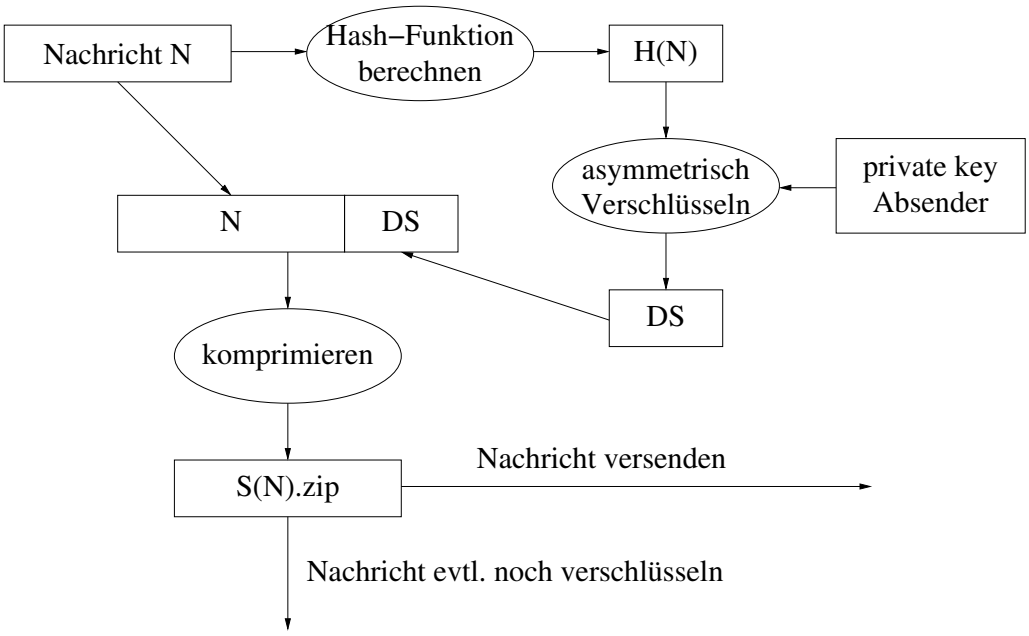


Abbildung 3.2: Ablauf beim digitalen Signieren mit PGP

PGP digitale Signatur

Weiterhin bietet PGP die Möglichkeit, Nachrichten digital zu signieren. Abbildung 3.2 zeigt den Ablauf.

1. Auf die Nachricht N wird eine der Hashfunktionen angewandt.

2. Der berechnete Hashwert $H(N)$ wird mit dem privaten Schlüssel des Absenders asymmetrisch verschlüsselt.

3. Die Nachricht und der verschlüsselte Hashwert (die digitale Signatur) werden konkateniert und anschließend komprimiert.

Falls die Nachricht vertraulich bleiben soll, dann kann sie (zusammen mit der digitalen Signatur) auch noch verschlüsselt werden. Die Schritte hierzu sind in Abbildung 3.1 abgebildet. Da die Nachricht bereits komprimiert wurde, geht es dort aber im Kästchen *N.zip* weiter.

Der Empfänger führt die gezeigten Berechnungen wieder rückwärts aus, also dekomprimieren, den verschlüsselten Hashwert mit dem öffentlichen Schlüssel des Absenders entschlüsseln und das Ergebnis mit dem neu berechneten Hashwert der Nachricht selbst vergleichen.

PGP-Quellen

Als Endanwender kann man PGP aus verschiedenen Quellen beziehen. Eine kommerzielle Version erhält man bei seinem Softwarehändler. Freie Versionen werden unter Linux i. d. R. mitgeliefert oder können direkt aus dem Internet geladen werden, z. B. unter den folgenden URLs:

- Unter `http://www.gnupg.org/` findet man die allgemeine Projektseite.
- Unter `http://www.gpg4win.org/` gibt es ein angepasstes Paket für Microsoft Windows.
- Unter `http://gpgtools.org/` gibt es ein angepasstes Paket für Apple Mac OS X.

Nachdem man PGP auf seinen Rechner geladen hat, muss man noch sicher stellen, dass man keine manipulierte Version geladen hat. Ein Hacker könnte schließlich den Quellcode nehmen, dort Falltüren einbauen und die so erstellte Version im Internet anbieten. Daher sollte man evtl. Versionen von *verschiedenen Servern* laden und diese miteinander vergleichen. Sind sie identisch, dann ist es unwahrscheinlicher, dass ein Hacker sie manipuliert hat. Alternativ kann man auch MD5- oder SHA-Prüfsummen bilden und diese mit den Angaben auf den oben genannten Webservern vergleichen.

PGP-Schlüsselgenerierung

Hat man PGP zum ersten Mal installiert, dann muss man sein eigenes Schlüsselpaar erstellen. Je nach PGP-Benutzeroberfläche (Kommandozeile oder grafische Oberfläche) startet man die Aktionen anders. Im Folgenden werden daher nur die Aktionen an sich beschrieben.

Zuerst muss man sich entscheiden, ob man einen RSA- oder einen DSS/Diffie-Hellman-Schlüssel erstellen möchte. Die alten PGP-2.x-Versionen unterstützen nur RSA-Schlüssel. Die meisten neuen Benutzer von PGP haben einen DSS-Schlüssel gewählt. Nachdem man sich für einen Schlüsseltyp entschieden hat, muss man die Länge des Schlüssels angeben. Je größer die Schlüssellänge ist, desto schwieriger ist es, den Schlüssel zu knacken.[1] Allerdings dauert die Generierung des Schlüssels länger. Da dies ein einmaliger bzw. seltener Vorgang ist, sollte das kein Problem sein. Die Verschlüsselung von großen Nachrichten dauert mit großen Schlüsseln auch nur *unwesentlich* länger, da ja nur der Sitzungsschlüssel asymmetrisch verschlüsselt wird. Man sollte hier also ruhig die größte angebotene Schlüssellänge wählen.

Als nächstes gibt man seine Benutzer-Identifikation ein. Es empfiehlt sich, auch den ausgeschriebenen eigenen Namen und die eigene E-Mail-Adresse anzugeben. Damit kann der öffentliche Teil des Schlüssels später besser dem Eigentümer zugeordnet werden. Abbildung 3.3 zeigt die Attribute des öffentlichen Schlüssels des Kursautors.

[1] Zur Erinnerung: Einen RSA-Schlüssel knacken heißt, eine große Zahl in ihre Primfaktoren zu zerlegen.

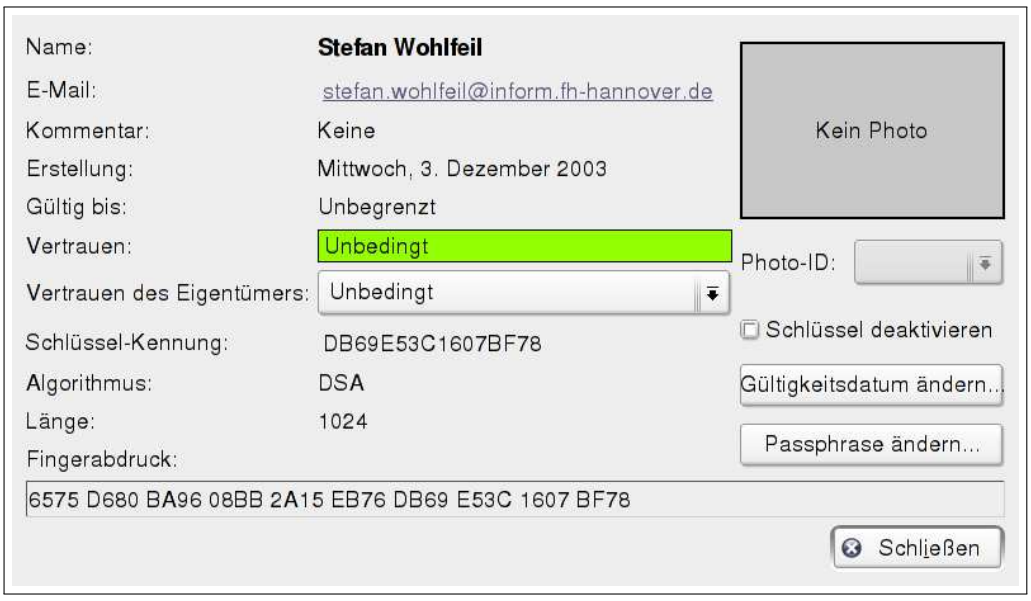


Abbildung 3.3: Eigenschaften eines PGP-Schlüssels

Der private Schlüssel muss nun auch irgendwo gespeichert werden. Da sichere Speicherumgebungen wie z. B. Chipkarten noch nicht so verbreitet sind, wird der private Schlüssel normalerweise in einer Datei gespeichert. Er wird allerdings zuvor noch einmal symmetrisch verschlüsselt. Dazu muss der Benutzer einen „Passwortsatz“ (engl. **pass phrase**)[2] eingeben. Dieser sollte relativ lang sein, aber trotzdem auch einfach zu merken. Man muss ihn später immer wieder eingeben, wenn man eingegangene Nachrichten entschlüsseln oder ausgehende Nachrichten signieren will. Aus dem Passwortsatz berechnet PGP einen Hashwert, mit dem dann der private Schlüssel symmetrisch verschlüsselt wird. Die Datei mit dem privaten Schlüssel heißt standardmäßig `secring.pgp`. Es empfiehlt sich, *unbedingt* Sicherheitskopien dieser Datei anzulegen, z. B. auf USB-Sticks oder CDs, die an einem sicheren Ort aufbewahrt werden. Trotzdem darf diese Datei nicht in die Hände anderer gelangen. Deshalb sind die Zugriffsrechte auf diese Datei so restriktiv wie möglich zu setzen.

pass phrase

Den öffentlichen Schlüssel muss man nun allen potentiellen Kommunikationspartnern bekannt machen. Dazu kann man seinen öffentlichen Schlüssel

- auf einen Server für öffentliche Schlüssel stellen.

  Unter der DNS-Domäne `pgp.net` stehen eine Reihe von Schlüsselservern zur Verfügung (siehe `http://www.pgp.net/`). Diese Server gleichen ihre Schlüsselbestände untereinander ab, d. h. man muss seinen Schlüssel nur auf einen dieser Server stellen.

  Außerdem betreibt die Firma PGP Corporation einen Schlüsselserver unter `http://keyserver.pgp.com/`

- per E-Mail verschicken.

- auf seiner Homepage im Internet veröffentlichen.

Auf diesen Wegen kann man dann auch den öffentlichen Schlüssel von jemandem erfahren, dem man eine verschlüsselte E-Mail schicken möchte. Damit dazu

[2] Die Passphrase wird oft auch „Mantra“ genannt.

nicht ständig solche Abfragen und Zugriffe erforderlich sind, speichert PGP die öffentlichen Schlüssel auch in einer lokalen Datei, die i. d. R. `pubring.pgp` heißt.

Einen einmal erstellten öffentlichen Schlüssel kann man von den Schlüsselservern nicht mehr einfach löschen, falls man seinen Passwortsatz vergessen hat oder falls man vermutet, dass der Schlüssel geknackt wurde. Stattdessen muss man den Schlüssel zurücknehmen (engl. **to revoke**). Hierfür braucht man eine „Widerrufurkunde“. Am besten erstellt man diese zusammen mit dem Schlüsselpaar und speichert die Widerrufurkunde an einem sicheren Ort.

Prüfen eines öffentlichen Schlüssels

Hat man den öffentlichen Schlüssel eines Kommunikationspartners geladen, so stellt sich die Frage, ob man tatsächlich den richtigen Schlüssel geladen hat. Schließlich kann sich jeder einen neuen Schlüssel generieren und dabei beliebige Namen benutzen. Man kann die Antwort auf diese Frage auf zwei verschiedenen Wegen erhalten:

1. Man ruft den Kommunikationspartner an und fragt nach dem Fingerabdruck seines öffentlichen Schlüssels. Diesen vergleicht man nun mit dem Fingerabdruck des geladenen Schlüssels. Stimmen sie überein, dann hat man den richtigen Schlüssel.[3]

2. Jemand, dem man *hinreichend viel* Vertrauen entgegenbringt und dessen öffentlichen Schlüssel man vorliegen hat, hat den zu prüfenden öffentlichen Schlüssel signiert. Diese Person nennt man bei PGP *Vorsteller* (engl. **introducer**).

Vorsteller

Der Introducer I macht einen Kommunikationspartner A mit einem anderen B bekannt. Danach kann B sicher sein, tatsächlich den öffentlichen Schlüssel von A vor sich zu haben. Die Voraussetzungen hierfür sind:

Voraussetzungen

1. B kennt den öffentlichen Schlüssel von I.

2. B vertraut I insoweit, dass I nur Personen bzw. deren Schlüssel signiert, die er tatsächlich genau kennt.

Technisch läuft dies wie folgt ab:

technischer Ablauf

- I hat den öffentlichen Schlüssel von A vorliegen und ist sicher, dass der Schlüssel korrekt ist.

- I signiert den öffentlichen Schlüssel von A, d. h. er berechnet den Hashwert des Schlüssels, verschlüsselt ihn mit seinem eigenen privaten Schlüssel und hängt seine Signatur an den öffentlichen Schlüssel von A.

- B erhält den von I signierten öffentlichen Schlüssel von A. Er prüft die Signatur mit dem öffentlichen Schlüssel von I. Anschließend kann B sicher sein, dass der öffentliche Schlüssel tatsächlich zu A gehört.

Dieses Schema liegt im Prinzip auch den in Abschnitt 2.7 vorgestellten Zertifizierungsstellen zu Grunde. Neben der Tatsache, dass man den öffentlichen Schlüssel der Zertifizierungsstelle kennen muss, ist auch die zweite Voraussetzung zu beachten. Woher kommt das Vertrauen in die Zertifizierungsstelle

[3] Sollten Sie dem Kursautor eine verschlüsselte E-Mail senden wollen, so vergleichen Sie den Fingerabdruck des geladenen öffentlichen Schlüssels einfach mit dem Fingerabdruck in Abbildung 3.3.

bzw. den Introducer? Was ist mit Stellen, denen man nicht 100-prozentig trauen will/kann? Wieviel Vertrauen hat man in öffentliche Schlüssel von nicht 100-prozentig vertrauenswürdigen Instanzen?

Vertrauensfragen

Die oben bereits genannten Voraussetzungen implizieren die beiden wichtigsten Vertrauensfragen:

1. Wieweit vertraue ich darauf, dass ein mir vorliegender öffentlicher Schlüssel korrekt ist? Das heißt, glaube ich tatsächlich, dass dieser Schlüssel zum vorgeblichen Besitzer gehört? In PGP nennt man dies Benutzervertrauen (engl. **owner trust**).

2. Wieweit vertraue ich jemandem, dessen öffentlichen Schlüssel ich sicher kenne, dass er andere öffentliche Schlüssel nur dann signiert, wenn sie korrekt sind? Das heißt, erwarte ich von einem Bekannten, dass er keine gefälschten öffentlichen Schlüssel signiert?[4] In PGP nennt man dies Signaturvertrauen (engl. **signature trust**).

Vertrauensstufen

Da PGP nicht an eine feste Zertifizierungs- bzw. Signierungshierarchie gekoppelt ist, muss jeder Benutzer auf Frage 2 eine eigene Antwort finden. PGP erlaubt hier folgende Abstufungen:

Ultimate Trust: Unbegrenztes Vertrauen wird nur der eigenen Person entgegen gebracht. Diese Stufe bekommen also nur öffentliche Schlüssel (Personen), deren privater Schlüssel auch bekannt ist (siehe Abbildung 3.3).

Always trusted to sign other Keys: Man vertraut diesem Benutzer soweit, dass nur korrekte Schlüssel von ihm signiert werden. Anders gesagt, man hält seine Überprüfungen für genauso gut wie die Prüfungen, die man selbst anstellen würde, bevor man einen fremden Schlüssel signiert. Man nennt diese Stufe auch *Full Trust*.

Usually trusted to sign other Keys: In diesem Fall geht man davon aus, dass von diesem Benutzer signierte öffentliche Schlüssel „normalerweise“ korrekt sind. Man nennt diese Stufe auch *Marginal Trust*.

Usually not trusted to sign other Keys: Diesem Benutzer misstraut man explizit. Er ist dafür bekannt, Schlüssel auch ohne eingehende Prüfung zu signieren. Man nennt diese Stufe auch *Never Trust*.

Unknown User: Man kennt diesen Benutzer nicht. Deshalb weiß man auch nicht, wie genau er Schlüssel prüft, bevor er sie unterschreibt.

Unknown Trust: Man weiß nicht, ob man dem Benutzer trauen kann. Was den Vertrauensgewinn in die Gültigkeit eines von diesem Benutzer unterschriebenen Schlüssels betrifft, ist diese Stufe äquivalent zur Stufe „Unknown User“. Diese beiden Stufen werden manchmal zusammengefasst.

Aus den Werten des Signature-Trust von Benutzern kann man nun eine Antwort auf die erste Vertrauensfrage (Owner-Trust) erhalten. In PGP kann ein öffentlicher Schlüssel von mehreren Introducern signiert sein. Lädt man also einen öffentlichen Schlüssel von einem Schlüsselserver oder bekommt einen

[4] Bei Zertifizierungsstellen sollte die Antwort immer „Ja“ lauten.

öffentlichen Schlüssel per E-Mail, dann ist er i. d. R. von mehreren Personen signiert. Falls man einige dieser Personen bereits kennt, dann kann man das Vertrauen „berechnen“.

Beispiel

Ein Beispiel: Ich habe den öffentlichen Schlüssel von Benutzer X bekommen, der mir bisher unbekannt ist. Der Schlüssel von X ist von A, B, C und D signiert. Falls ich nun B und D kenne und beiden den Wert „Always trusted to sign other Keys“ als Signature-Trust zugeordnet habe, dann kann ich mir recht sicher sein, dass ich tatsächlich den Schlüssel von X vor mir habe.

Technisch sieht das so aus, dass man zwei Werte K und L definiert. Man zählt bei den Signaturen eines unbekannten Schlüssels die Anzahl von Signaturen, die von Benutzern stammen, denen man Signaturvertrauen „Always trusted to sign other Keys“ entgegenbringt. Mindestens K solcher Signaturen sollten vorliegen, damit man dem unbekannten Schlüssel Benutzervertrauen schenkt. Im obigen Beispiel war $K = 2$. L ist dann die Zahl von Signaturen der Klasse „Usually trusted to sign other Keys“, die man für erforderlich hält. Normalerweise ist $L > K$. Bei einem neuen Schlüssel bildet man die folgende Summe:

$$\frac{\#\,\text{Signaturen } \textit{always}}{K} + \frac{\#\,\text{Signaturen } \textit{usually}}{L}$$

Ist diese gewichtete Summe größer gleich 1, dann vertraut man auf die Korrektheit des neuen öffentlichen Schlüssels. Für den neuen Schlüssel stellt sich allerdings erneut die Vertrauensfrage 2. Der Benutzer muss hier wieder selbst einen Wert zuordnen. Da man den „Neuen“ bisher nicht kennt (man ist sich nur sicher, dass er es tatsächlich ist), wird man seiner Signatur unter weiteren öffentlichen Schlüsseln eher wenig bis gar kein Vertrauen entgegenbringen.

web of trust

Mit der Zeit baut man also einen Bekanntenkreis oder auch ein Vertrauensnetzwerk (engl. **web of trust**) auf. Im Gegensatz zur starren Hierarchie von Zertifizierungsstellen kann bei PGP jeder jedem anderen ein selbst definiertes Maß an Vertrauen entgegen bringen. Eine Zertifizierungsstelle lebt davon, dass sie im Feld *Signature-Trust* von allen Teilnehmern einen hohen Vertrauenswert erhält.

Übungsaufgabe 3.1 *Von den folgenden fünf PGP-Benutzern liegen Ihnen die öffentlichen Schlüssel vor und Sie haben ihnen folgende Vertrauensstufen zugeordnet:*

	A	*B*	*C*	*D*	*E*
Vertrauen	*always trusted*	*usually trusted*	*usually not trusted*	*always trusted*	*unknown*

Weiterhin haben Sie sich für die Werte $K = 1$ und $L = 2$ entschieden. Sie erhalten die öffentlichen Schlüssel von zwei weiteren PGP-Benutzern X und Y, die wie folgt signiert sind:

	X	*Y*
Signiert von	*A, C*	*B, C, E*

Trauen Sie den öffentlichen Schlüsseln von X und Y?

Nachdem Sie festgestellt haben, dass ein vorliegender öffentlicher Schlüssel tatsächlich zum eingetragenen Besitzer gehört, sollten Sie das durch eine Unterschrift unter dem Schlüssel bestätigen. Sie machen also im Prinzip das Gleiche wie eine Zertifizierungsstelle. Durch ihre Unterschrift erreichen Sie die folgenden beiden Punkte:

1. Ihr System kennt nun den öffentlichen Schlüssel eines Kommunikationspartners und kann digital signierte Nachrichten von diesem Partner automatisch prüfen und entsprechend anzeigen. Außerdem können verschlüsselte Nachrichten an diesen Partner geschickt werden.

2. Falls Sie den unterschriebenen Schlüssel wieder dem Partner zur Verfügung stellen, dann kann der Partner den Schlüssel mit Ihrer Unterschrift weiter veröffentlichen. Andere haben dann evtl. mehr Vertrauen in die Gültigkeit des öffentlichen Schlüssels, da ja Ihre Unterschrift die Korrektheit bestätigt hat.

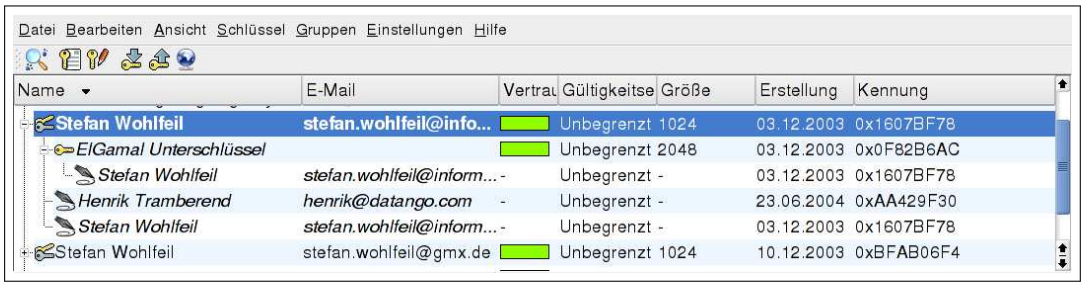


Abbildung 3.4: Verwaltung von PGP-Schlüsseln

Abbildung 3.4 zeigt, dass der öffentliche Schlüssel des Kursautors vom Kollegen Henrik Tramberend unterschrieben wurde. Außerdem ist der Schlüssel auch vom Besitzer selbst unterschrieben.

Zusammenfassung PGP/GnuPG: Mit PGP bzw. GnuPG bekommt man als Privatanwender ein gutes, gut untersuchtes und vertrauenswürdiges System an die Hand. Es ist auf den gängigsten Betriebssystemplattformen verfügbar. Man kann (und muss) in PGP allerdings selbst ein Vertrauensnetz aufbauen, um die Korrektheit unbekannter öffentlicher Schlüssel prüfen zu können. Außerdem muss man als „Neuer“ auch in das *Web of Trust* eingeführt werden. Hat man allerdings mit seinen Kommunikationspartnern erfolgreich die Schlüssel ausgetauscht, so kann man mit diesen Partnern sehr einfach vertraulich und authentisch kommunizieren.

Beim Einsatz von PGP/GnuPG braucht man also *keine* zentrale vertrauenswürdige Zertifizierungsstelle. Im Privatbereich ist PGP bzw. GnuPG deshalb sehr beliebt. Im geschäftlichen Umfeld wird PGP dagegen seltener eingesetzt.

3.2.2 Secure MIME (S/MIME)

In den Request for Comments (RFC) Nummer 2311 und 2312 werden die *Secure/Multipurpose Internet Mail Extensions (S/MIME)* definiert. Sie ergänzen die RFCs 2045 bis 2049 (in denen MIME definiert wurde) um Sicherheitsfunktionen für die Schutzziele Vertraulichkeit und Authentizität. Die MIME-Spezifikation ist wiederum eine Erweiterung einer älteren Definition. Die ursprünglichen E-Mail-Spezifikationen stehen in den RFCs Nummer 821 (Simple-Mail-Transfer-Protocol) und Nummer 822 (Standard for the Format of ARPA-Internet-Text-Messages). Eine E-Mail besteht demnach aus zwei Teilen, einem Umschlag (engl. **envelope**) und ihrem Inhalt (engl. **content**), siehe hierzu auch Abschnitt 1.3.4 in Kurseinheit 1. Der Inhalt besteht auch aus

zwei Teilen, dem Nachrichtenkopf (engl. **message header**) und dem Nachrichtenrumpf (engl. **message body**). Es ergibt sich folgende „Hierarchie“ von Spezifikationen.

Hierarchie von RFCs

RFC 2311–2312	S/MIME; Verschlüsselung, Signaturen etc.
RFC 2045–2049	MIME; Mehrteilige Multimedia-Nachrichten
RFC 821–822	SMTP und Inhaltsbeschreibung

Die Definitionen der RFCs Nummer 821 und 822 enthalten einige funktionale Defizite:

- Der Inhalt einer Nachricht sollte nur aus ASCII-Zeichen bestehen, d. h. das höchstwertige Bit jedes Bytes soll Null sein. Daher können keine ausführbaren Programme in einer E-Mail verschickt werden.

  Auch Nachrichten mit landesspezifischen Sonderzeichen, wie ä, ö oder ß, sind nicht vorgesehen. Solche Nachrichten werden von Mailsystemen zwar nicht zurückgewiesen, es ist aber auch nicht eindeutig definiert, wie sie behandelt werden. Ein Mailsystem kann sie korrekt weiterleiten, ein anderes setzt vielleicht alle höchstwertigen Bits auf Null und verändert dadurch u. U. die Nachricht.

- Ein SMTP-Server kann Nachrichten über einer bestimmten Größe zurückweisen.

- Beim Transport zwischen Rechnern mit verschiedenen Zeichensatzcodierungen (z. B. ASCII und EBCDIC) können Umcodierungsprobleme auftreten.

- Es ist nicht definiert, wie man eine Nachricht mit einem Anhang (engl. **attachment**) versieht.

Diese Defizite werden in MIME angegangen. Dazu werden neue Nachrichtenkopfeinträge (engl. **header fields**) definiert. Sie enthalten folgende Informationen über den Nachrichtenrumpf:

MIME-Header

- die Codierung des Rumpfes (engl. **content transfer encoding**),

- den Typ des Rumpfes (engl. **content type**) (z. B. Text, Image, Audio etc.) und

- eine Inhaltsbeschreibung (engl. **content description**).

S/MIME erweitert MIME, indem zusätzliche Nachrichtenrumpftypen (engl. **content types**) definiert werden.

MIME-Content-Transfer-Encodings: Während im SMTP-Protokoll die Codierung der Zeichen fest definiert ist (nämlich ASCII-Codierung), so schreibt MIME *keine* feste Codierung vor. Stattdessen können Sender und Empfänger eine zur Nachricht passende Codierung wählen. MIME definiert im Prinzip nur, wie die Auswahl der Codierung der anderen Seite unmissverständlich mitgeteilt wird. Die wichtigsten Codierungen sind:

MIME-Transfer-Encodings

`7bit`	Die Daten sind im ASCII-Format
`8bit`	Die Daten sind zwar überwiegend im ASCII-Format, jedoch können auch Sonderzeichen vorkommen.
`quoted-printable`	Die Daten bestehen überwiegend aus ASCII-Text. Sonderzeichen werden umcodiert und durch eine Kette von ASCII-Zeichen ersetzt. Eine so codierte Nachricht ist im großen und ganzen für Menschen noch vernünftig lesbar.
`base64`	Diese Codierung fasst jeweils drei Bytes zusammen. Die so entstandenen 24 Bit werden nun in vier Sechsergruppen geteilt. Jede mögliche Sechsergruppe wird *fest* auf ein bestimmtes ASCII-Zeichen abgebildet. Aus drei uncodierten Bytes werden so vier codierte Bytes, die garantiert korrekt übertragen werden.

MIME-Content-Types: Hat der Empfänger einer MIME-Nachricht diese erfolgreich dekodiert, so muss immer noch geklärt werden, wie die Nachricht zu interpretieren ist. Konkret bedeutet das, dass man unterscheiden muss, ob die Nachricht ein Bild, ein normaler Text, ein Video oder ... ist. Da für Bilder beispielsweise verschiedene Formate existieren, ist in MIME eine zweistufige Hierarchie von *Content-Types* vorgesehen. Der *Type* beschreibt, um welchen grundsätzlichen Typ es sich handelt (z. B. Bild) und ein *Subtype* konkretisiert dann das Format (z. B. GIF-Bild). Die wichtigsten MIME-Content-Types sind:

MIME-Content-Types

Type	Subtype	Beschreibung
`text`	`plain`	unformatierter ASCII- bzw. ISO 8859-Text
	`enriched`	Text mit einigen Formatierungsinformationen
`image`	`jpeg`	Eine Grafik im JPEG-Format
	`gif`	Eine Grafik im GIF-Format
`application`	`postscript`	Ein Programm in PostScript
	`octet-stream`	Ein beliebiges binärcodiertes Programm
`multipart`	`mixed`	Die Nachricht besteht aus mehreren Teilen, z. B. aus Text und einem Anhang

Bei einer mehrteiligen Nachricht wird im Content-Type auch spezifiziert, woran man in der Nachricht erkennt, wann ein Teil aufhört und der nächste Teil beginnt. In jedem Teil können wieder MIME-Kopfzeilen vorkommen, in denen z. B. die Codierung und der Typ des Teils beschrieben ist. Ein Beispiel für eine MIME-E-Mail aus dem RFC 2046 ist:

MIME-Beispiel

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Date: Sun, 21 Mar 1993 23:56:48 -0800 (PST)
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble.  It is to be ignored, though it
is a handy place for composition agents to include an
explanatory note to non-MIME conformant readers.

--simple boundary
```

```
This is implicitly typed plain US-ASCII text.
It does NOT end with a linebreak.
--simple boundary
Content-type: text/plain; charset=us-ascii

This is explicitly typed plain US-ASCII text.
It DOES end with a linebreak.

--simple boundary--
```

Im Nachrichtenkopf wird die Nachricht als mehrteilige Nachricht definiert. Als Begrenzungszeichenkette (engl. **boundary**) wird der String `simple boundary` festgelegt. Eine Zeile, die mit zwei Bindestrichen beginnt und denen die definierten Boundary folgt, markieren den Anfang des nächsten Teils. Das Ende des letzten Teils wird durch zwei zusätzliche Bindestriche am Ende der o. g. Trennzeile dargestellt. Im zweiten Teil der Nachricht ist dann eine eigene Kopfzeile definiert, die besagt, wie die Codierung im zweiten Teil erfolgt.

Übungsaufgabe 3.2 *Die MIME-Content-Types erlauben es einem E-Mail-Client, den Inhalt der E-Mail direkt aufzubereiten. Beispielsweise könnte der E-Mail-Client eingebundene Grafiken sofort (und* ohne *Aktion des Benutzers) anzeigen. Welche Sicherheitsrisiken sind damit verbunden, wenn ein E-Mail-Client sofort auf die verschiedenen Content-Types reagiert?*

S/MIME-Content-Types: S/MIME führt weitere Content-Types ein. Die wichtigsten sind:

`multipart`	`signed`	Die Nachricht besteht aus zwei Teilen, dem Nachrichteninhalt und einer digitalen Signatur
`application`	`pkcs7-mime`	Dieser Teil enthält eine verschlüsselte Nachricht oder eine digitale Signatur
	`pkcs7-signature`	Dieser Teil enthält die digitale Signatur
	`pkcs10-mime`	Dieser Teil enthält den Antrag auf Registrierung eines Zertifikats

Der Typ `application/pkcs7-mime` hat einen Parameter mit Namen `smime-type`. Dieser beschreibt eine dritte Hierarchie-Ebene. Darin wird unterschieden, ob dieser Teil eine nur verschlüsselte Nachricht, eine verschlüsselte und signierte Nachricht oder eine nur signierte Nachricht ist. Ein Beispiel aus dem RFC für eine nur signierte Nachricht ist:

S/MIME nur signierte Nachricht

```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
```

```
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

Zusammenfassung S/MIME: Mit Hilfe von S/MIME kann man eindeutig beschreiben, aus welchen Teilen eine Nachricht besteht (z. B. verschlüsseltem Inhalt, digitaler Signatur, Absenderzertifikat etc.) und wie diese Teile zu interpretieren sind. Weiterhin ist in S/MIME auch festgelegt, welche Algorithmen benutzt werden.

Mehr und mehr E-Mail-Clientprogramme unterstützen S/MIME. Zusammen mit Zertifikaten im X.509v3-Standard erlauben sie sichere E-Mail-Kommunikation. Alles was man als Benutzer noch braucht ist ein X.509-Zertifikat. Der DFN-Verein betreibt eine PKI, über die man als Hochschulangehöriger (über das lokale Hochschulrechenzentrum) ein Zertifikat bekommen kann. Den zugehörigen privaten Schlüssel muss man dann auf seinem Rechner installieren.

3.2.3 Zusammenfassung: Sichere E-Mail

Mit PGP und S/MIME stehen zwei Möglichkeiten zur Verfügung, vertraulich und authentisch E-Mails auszutauschen. Sie unterscheiden sich im Wesentlichen darin, wie Zertifikate gehandhabt werden. PGP setzt dabei auf das eher informelle *Web of Trust*, bei dem sich die Teilnehmer gegenseitig zertifizieren. Demgegenüber setzt S/MIME auf das eher formale System von Zertifizierungsstellen.

Im privaten Umfeld trifft man daher mehr PGP-Benutzer, im geschäftlichen Umfeld mehr S/MIME-Benutzer. Vom Standpunkt der Sicherheit sind beide Möglichkeiten vergleichbar gut. Ihr persönliches Umfeld bestimmt daher am ehesten, welche der beiden Techniken Sie einsetzen möchten. Man kann beides auf seinem Rechner installieren und dann wahlweise mal mit PGP verschlüsseln und mal mit S/MIME.

Moderne E-Mail-Programme wie *Thunderbird* (für alle Betriebssysteme verfügbar), *Outlook* (für MS Windows verfügbar), *Mail* (für Mac OS X) oder *KMail*/*Evolution* (für Linux verfügbar) haben S/MIME direkt einprogrammiert. PGP ist auch direkt einprogrammiert, oder man kann es als plug-in nachinstallieren. In nahezu jedem System sind also beide Verschlüsselungsoptionen vorhanden. Als Benutzer haben Sie damit die freie Wahl.

3.3 Sicheres „Surfen“ im Internet

Wie schon in Abschnitt 1.2 dargestellt, wird das World Wide Web (WWW) mehr und mehr für kommerzielle Transaktionen (z. B. Einkäufe oder Electronic Banking) genutzt. Dabei sind insbesondere die Schutzziele

Schutzziele

Vertraulichkeit: Nur Berechtigte können vom Inhalt der Kommunikation Kenntnis erlangen.

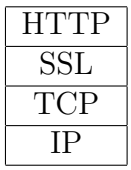
Authentizität: Die Beteiligten sind sich über die Identität des Kommunikationspartners im Klaren.

Privatheit: Über einen Benutzer werden ohne sein Wissen und Einverständnis keine Datensammlungen angelegt.

gefährdet. In Abschnitt 3.3.1 wird dargestellt, wie man Vertraulichkeit und Authentizität mit Hilfe von Verschlüsselungstechniken sicherstellen kann. Anschließend wird in Abschnitt 3.3.2 darauf eingegangen, wie diese Technik in der Praxis eingesetzt wird. Konkret werden die wichtigsten Sicherheitseinstellungen von Webbrowsern besprochen. Der Aspekt der Privatheit (Anonymität) wird in Kurs *(01868) Sicherheit im Internet 1 – Ergänzungen* besprochen.

3.3.1 Secure Socket Layer

Das Secure-Socket-Layer-Protokoll (SSL-Protokoll) wurde ursprünglich von der Firma *Netscape* entwickelt. Es erlaubt vertrauliche und authentische Kommunikation. Von der Architektur her liegt es zwischen der Transport- und der Anwendungsschicht des Protokollstacks (siehe Abbildung 3.5).



Anwendungsprotokoll
Sicherungsprotokoll
Transportprotokoll
Internetprotokoll

Abbildung 3.5: Einordnung von SSL in den Protokollstack

Vorteile SSL-Architektur

Diese Architektur hat zwei Vorteile:

1. Die Anordnung von SSL unterhalb der Anwendungsprotokolle erlaubt die mehr oder weniger unveränderte Weiternutzung bestehender Anwendungsprotokolle. Es müssen also keine grundlegend neue Protokolle entwickelt und auf Client- sowie Serverseite implementiert werden.

2. Obwohl ursprünglich für das „Surfen“ im Web (also das Anwendungsprotokoll HTTP) gedacht, kann SSL auch zur Absicherung anderer Anwendungsprotokolle eingesetzt werden. Beispielsweise können *telnet*, SMTP, *ftp* oder POP3 zusammen mit SSL benutzt werden.

Transport Layer Security

Man spricht deshalb auch allgemein von „Sicherung auf Transportebene“ (engl. **Transport Layer Security (TLS)**). Die *Internet Engineering Task Force (IETF)* hat eine Arbeitsgruppe zur Standardisierung von TLS eingesetzt. Der dort erarbeitete Standard TLS 1.0 ist im *Request for Comments (RFC)* Nummer 2246 beschrieben und wurde 2006 unter dem Namen TLS 1.1 in RFC 3546 aktualisiert und erweitert. Im Jahr 2008 wurde dann in RFC 5246 die Version TLS 1.2 definiert. TLS basiert auf SSL, ist allerdings *nicht* 100-prozentig kompatibel zu SSL. Details unterscheiden sich, das Prinzip ist allerdings gleich. Zu den unterschiedlichen Details gehören beispielsweise die auszuhandelnden Verschlüsselungsalgorithmen, Hash-Algorithmen, usw. In der Praxis sollten statt der SSL Versionen SSL 2 oder SSL 3 besser die TLS-Versionen TLS 1.2 oder TLS 1.1 benutzt werden. In SSL 2 wurden Sicherheitsprobleme gefunden und es gilt daher als unsicher. In TLS 1.2 wurden neue Algorithmen wie SHA-256, AES und der Counter Mode eingeführt.

Die SSL[5]-Schicht ist intern wiederum in Schichten aufgeteilt, siehe Abbildung 3.6.

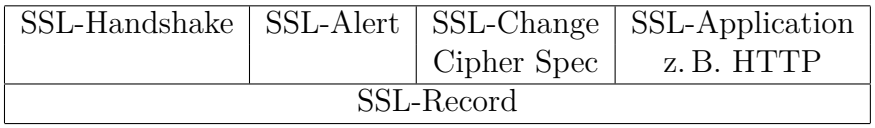


Abbildung 3.6: Schichten von SSL

Das SSL-Record-Protokoll hat dabei die Aufgabe, die Anwendungsdaten gesichert zu übertragen. Dazu können die Daten zuerst komprimiert werden. Dann wird ein Message Authentication Code (MAC) (siehe dazu auch Abschnitt 2.6.1) an den Datenblock angehängt. Der so entstandene Block wird dann symmetrisch verschlüsselt und mit einem *SSL-Header* versehen an die Transportschicht zur Übertragung übergeben.

Die Berechnung des Message Authentication Codes und die Verschlüsselung erfolgen mit Hilfe eines geheimen Schlüssels. Die Hashfunktion zur Berechnung des MAC heißt HMAC und ist im *Request for Comments RFC 2104* definiert. Es ist eigentlich eher ein Hashberechnungsschema als eine konkrete Hashfunktion. Das Schema beschreibt nur, wie mit Hilfe einer konkreten Hashfunktion, wie z. B. MD5 oder RIPEMD-160, ein MAC berechnet wird. Bei der symmetrischen Verschlüsselung können verschiedene Algorithmen wie DES, RC2, RC4, IDEA, Camellia[6] oder AES eingesetzt werden. Da die Schlüssellänge bei DES zu kurz ist und weil heute Berichte vorliegen, nach denen die NSA mit RC4 verschlüsselte Daten „knacken“ kann, sollten diese beiden Algorithmen besser nicht mehr benutzt werden.

Welcher Algorithmus letztlich benutzt wird und welche geheimen Schlüssel eingesetzt werden, handeln der Webbrowser und der Webserver zu Beginn der Kommunikation aus. Das *SSL-Handshake*-Protokoll regelt diese Verhandlung.

SSL-Handshake

Abbildung 3.7 zeigt den prinzipiellen Ablauf.

Client Hello: Jede SSL-Verbindung wird vom Client initiiert. Er schickt eine Nachricht an den Server, die unter anderem die folgenden Daten enthält:

- Die Versionsnummer des vom Client verwendeten SSL-Protokolls.
- Eine Liste von Verschlüsselungsalgorithmen, die der Client ausführen kann. Diese Liste ist nach Prioritäten sortiert, so dass der Server erkennen kann, welchen Algorithmus der Client gerne verwenden möchte.
- Neben den Verschlüsselungsalgorithmen schlägt der Client auch eine Reihe von Schlüsselaustauschverfahren und Message Authentication Codes vor.

Server Hello: Nachdem der Client dem Server mitgeteilt hat, dass er eine Verbindung aufbauen möchte, antwortet der Server mit der *Server-Hello*-Nachricht. Sie enthält dieselben Daten, wie die *Client-Hello*-Nachricht. Allerdings wählt der Server aus den Listen des Clients ein Verschlüsselungs- und ein Schlüsselaustauschverfahren aus.

[5]Im folgenden werden die Begriffe SSL und TLS synonym benutzt. Es ist immer das grundlegende Prinzip, über das geschrieben wird.

[6]Camellia ist ein symmetrischer Verschlüsselungsalgorithmus, der 128 Bit Blöcke in einem Feistel-Netz verarbeitet. Er gilt als sicher.

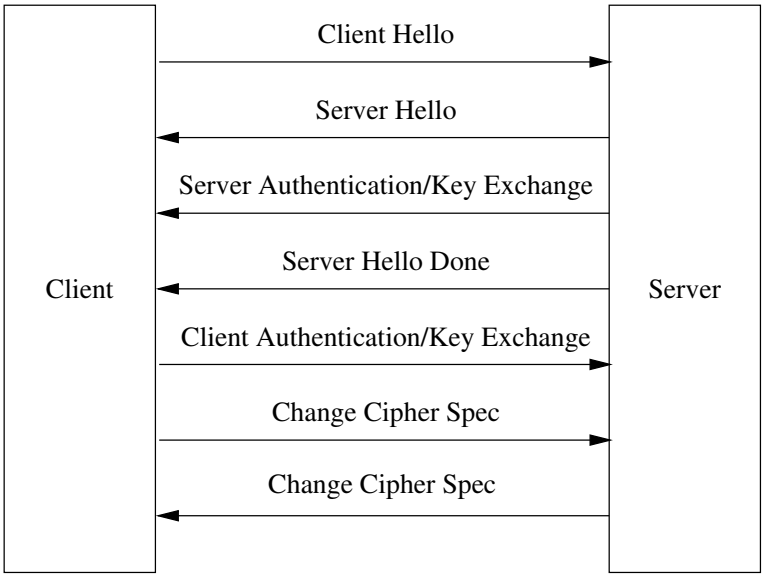


Abbildung 3.7: SSL-Handshake-Ablauf

Server-Authentication/Key-Exchange: Falls der Client in seiner *Client-Hello*-Nachricht verlangt hat, dass sich der Server authentisiert, so schickt der Server nun sein Zertifikat an den Client. Das Zertifikat enthält den öffentlichen Schlüssel des Servers. Hat der Client kein Zertifikat verlangt, so kann der Server stattdessen auch die Daten für den Austausch des Diffie-Hellman-Schlüssels (siehe hierzu auch Abschnitt 2.4.2) schicken.

Server Hello Done: Mit dieser Nachricht teilt der Server dem Client mit, dass er alle Antworten und Authentisierungsnachrichten geschickt hat.

Client-Authentication/Key-Exchange: Hat der Client ein Serverzertifikat erhalten, so überprüft er es als Erstes. In der Praxis sieht das so aus, dass der Browserhersteller eine Reihe von Zertifikaten von anerkannten Zertifizierungsstellen zusammen mit dem Browser ausliefert. Kritische Benutzer überprüfen diese Zertifikate einmal (z. B. bei der Installation des Browsers) mit Hilfe der Fingerprints, die sie auf anderem Wege erfahren haben.

Falls der Server den Client authentisieren will, so schickt der Client nun sein Zertifikat an den Server. Hat der Client kein Zertifikat, so teilt er dies dem Server mit. Danach schickt der Client eine Nachricht zum Schlüsselaustausch. Diese Nachricht hängt von der Art der Serverauthentisierung ab.

- Hatte der Server ein Zertifikat geschickt, so generiert der Client einen geheimen „Vorab-Schlüssel“ (engl. **premaster secret**). Dieser wird mit dem öffentlichen Schlüssel des Servers verschlüsselt und an diesen geschickt.
- Hatte der Server Diffie-Hellman-Schlüsselaustausch gewählt, so antwortet der Client mit den entsprechenden Parametern.

Change Cipher Spec: Der Client berechnet nun aus dem Vorab-Schlüssel und weiteren Parametern (z. B. Zufallswerte, die in den *Hello*-Nachrichten ausgetauscht wurden und das spätere Wiedereinspielen der Nachrichten

verhindern sollen) den endgültigen geheimen Schlüssel. Nun schickt er dem Server die Nachricht, ab jetzt bitte verschlüsselt zu senden. Zum Test schickt der Client eine verschlüsselte kurze Bestätigungsnachricht (*Finished Message*) mit.

Der Server berechnet (hoffentlich) denselben endgültigen Schlüssel und teilt dem Client mit, dass er ab jetzt verschlüsselt sendet. Zur Kontrolle wird wieder eine verschlüsselte Bestätigungsnachricht mitgeschickt.

Nun haben Client und Server eine sichere Verbindung aufgebaut. Die Beschreibung hier gibt nur die Prinzipien des Ablaufs wieder. Details zu den Verschlüsselungsverfahren und der Art der einzelnen Nachrichten findet man bei Stallings [Sta06] und in den RFCs 2246 und 3546.

Wird beim Schlüsselaustausch die Variante mit dem premaster secret gewählt, so hat das eine wichtige Konsequenz für Angriffe auf verschlüsselte Verbindungen. Kann ein Angreifer die Nachrichten des SSL handshake mitlesen, so kann er auch die Nachricht des Client mit dem premaster secret speichern. Kommt der Angreifer nun nachträglich in den Besitz des privaten Schlüssels des Servers, so kann er nachträglich den session key ausrechnen. Dazu wiederholt der Angreifer die Berechnungen, die auch der Server durchführt. Hat der Angreifer auch den anschließenden (mit SSL verschlüsselten) Nachrichtenaustausch protokolliert, so kann er diesen nun entschlüsseln. Normalerweise erwartet man, dass Server-Betreiber ihre privaten Schlüssel nicht einfach herausgeben. Die Firma *Lavabit* aus den USA hat 2013 ihren verschlüsselten E-Mail-Dienst eingestellt. Grund dafür war, dass US-Gerichte die Firma verpflichtet hatten, die privaten Schlüssel herauszugeben. Damit wollte die NSA den Datenverkehr zwischen den *Lavabit*-Kunden und den *Lavabit*-Servern entschlüsseln und mitlesen.

Besser wäre es also, wenn die session keys mit Hilfe des Diffie-Hellman-Schlüsselaustausch bestimmt werden. Konkret sollten *ephemeral Diffie-Hellman (TLS_DHE)* oder *ephemeral Elliptic-Curve Diffie-Hellman (TLS_ECDHE)* benutzt werden. Dabei erzeugt der Server jedes Mal einen anderen Exponenten und vergisst diesen nach Ablauf der Sitzung. Der session key kann also nachträglich nicht mehr rekonstruiert werden. Man spricht in diesem Fall auch von **perfect forward secrecy**.

perfect forward secrecy

Häufig besteht eine HTML-Seite nicht nur aus einer Datei, sondern sie enthält Grafiken u. ä. Damit der aufwendige SSL-Handshake nicht für jede Datei neu durchgeführt werden muss, enthält SSL (ebenso wie HTTP/1.1) ein Sitzungskonzept. Man beginnt eine Sitzung (engl. **session**) mit dem SSL-Handshake. Die während der Sitzung hergestellten Verbindungen können dann ohne weiteren SSL-Handshake erfolgen. Eine Sitzung wird entweder durch explizites Beenden oder durch Ablauf einer Zeitscheibe beendet. Nachfolgende Verbindungswünsche müssen wieder mit dem SSL-Handshake eingeleitet werden.

SSL-Sessions

3.3.2 Sicherheitseinstellungen von Webbrowsern

SSL wird beim E-Commerce im World Wide Web oft eingesetzt. Am Beispiel Internetbanking werden nun einige Aspekte aus der Praxis diskutiert.

Verschlüsselung: Soll eine Seite aus dem Web nur mit SSL verschlüsselt übertragen werden, so erkennt man dies an dem *Uniform-Resource-Locator (URL)* der Seite. Statt mit `http://...` beginnt die URL mit `https://...`

Wählt man so eine Seite an, so veränderte sich die Benutzeroberfläche des Browsers früher nur wenig. Abbildung 3.8 zeigt die Benutzeroberfläche, wenn eine Seite ungesichert übertragen wurde. Man erkennt das an der normalen Schrift in der die URL oben angezeigt wird.

https

Abbildung 3.8: Ungesicherte Übertragung einer WWW-Seite

Zum Vergleich hierzu zeigt Abbildung 3.9 das Aussehen der Benutzeroberfläche, wenn die Seite mit SSL gesichert übertragen wird. Im URL-Feld des Browsers steht vor der URL in grüner Schrift der Name des Inhabers des Zertifikats, mit dem der Server sich während des SSL-Handshake authentisiert hat. So kann ein Benutzer auf den ersten Blick erkennen, ob die Verbindung verschlüsselt ist und ob der Anbieter der Webseite ein Zertifikat einer vertrauenswürdigen Zertifizierungsstelle besitzt.

In anderen Browsern, beispielsweise dem *Internet Explorer* der Firma *Microsoft* oder *Safari* der Firma *Apple*, finden sich ähnliche Hinweise. Beachten Sie dazu bitte das jeweilige Benutzerhandbuch.

SSL und Zertifikate

Internetbankingserver haben i. d. R. alle ein Zertifikat. Schließlich will ein Kunde sicher sein, tatsächlich mit seiner Bank zu kommunizieren. Die Zertifikate kann man in seinem Browser einsehen. Dazu klickt man im *Firefox* bzw. in *Mozilla* auf den grün hinterlegten Bereich mit den Namen des Zertifikatinhabers. Es folgt ein Fenster, in man „weitere Informationen...“ anklicken kann um ein weiteres Fenster zu bekommen. Im dann erscheinenden Fenster Seiteninformation klickt man auf „Zertifikat anzeigen“. Anschließend öffnet sich ein Fenster wie in Abbildung 3.10.

Server-Zertifikat

Man erkennt am Zertifikat, dass es tatsächlich für die *Deutsche Bank* ausgestellt wurde. Ausgegeben wurde es von der Firma *VeriSign* am 19.09.2012.

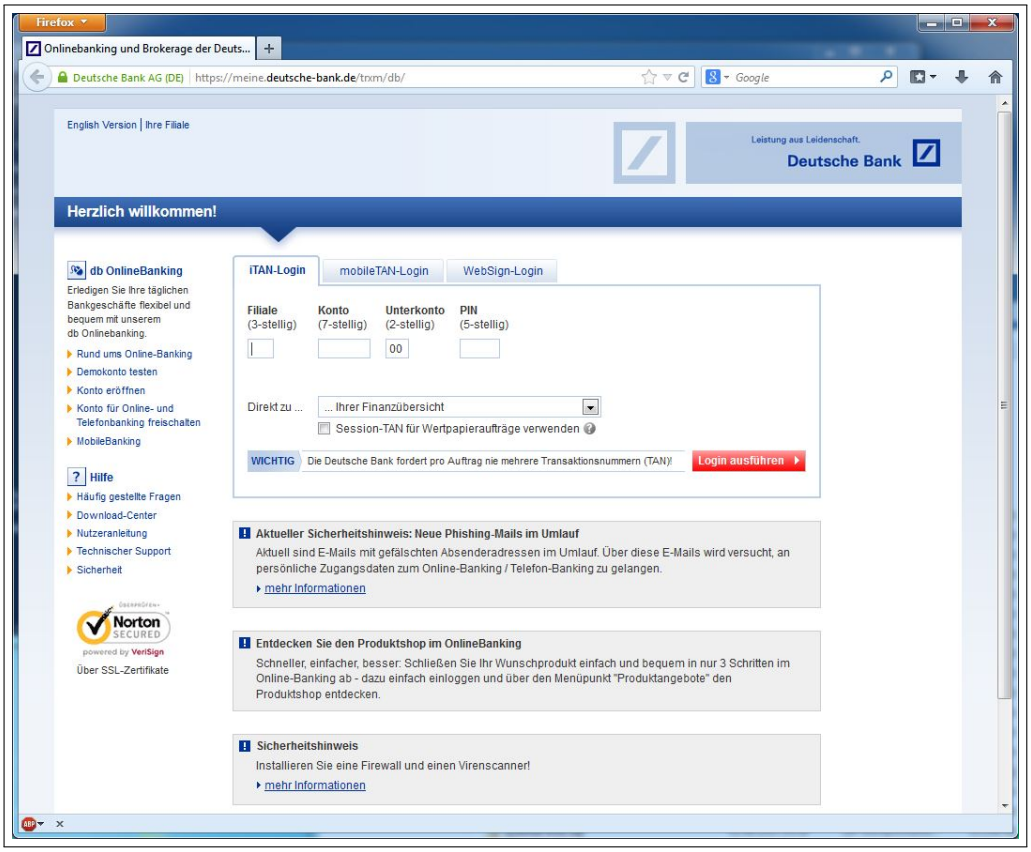


Abbildung 3.9: Gesicherte Übertragung einer WWW-Seite

Weiterhin sind der Gültigkeitszeitraum und zwei digitale Fingerabdrücke (SHA-1 und MD5) angegeben. Diese Fingerabdrücke kann man auch auf anderem Wege, z. B. durch Nachfrage erfahren und somit sicher sein, dass das Zertifikat tatsächlich zur *Deutschen Bank* gehört. Da der öffentliche Schlüssel von *VeriSign* dem Browser bereits bekannt ist, wird die Verbindung ohne Rückfrage aufgebaut.

Übungsaufgabe 3.3 *Welche Probleme ergeben sich aus der Tatsache, dass öffentliche Schlüssel von Zertifizierungsstellen dem Browser bekannt sein müssen?*

Benutzer-Authentisierung

Nachdem sich der Benutzer sicher sein kann, tatsächlich mit der Bank verbunden zu sein, muss die Bank erfahren, welcher Kunde die Verbindung gerade aufbaut. Da Zertifikate für Privatpersonen heute noch selten sind, werden andere Authentisierungsverfahren benutzt. Typischerweise werden eine Benutzer-Identifikation (Kontonummer) und ein Passwort (PIN) abgefragt (siehe Abbildung 3.9). Diese werden erst dann übertragen, wenn die verschlüsselte Verbindung bereits aufgebaut ist. Das SSL-Handshake-Protokoll ist also schon abgeschlossen.

Nach der Eingabe der Benutzer-ID und der PIN sind sich also beide Kommunikationspartner über die Identität des anderen im Klaren. Obwohl es für Angreifer bereits extrem schwer ist, diese Kommunikation abzuhören, gibt es im Internetbanking drei weitere Schutzmaßnahmen.

TAN

1. Jede Transaktion (d. h. Überweisung, Aktienorder, PIN-Änderung etc.) des Benutzers muss zusätzlich durch ein Einmalpasswort (Transaktionsnummer) autorisiert werden. Eine Liste dieser Nummern wird von der

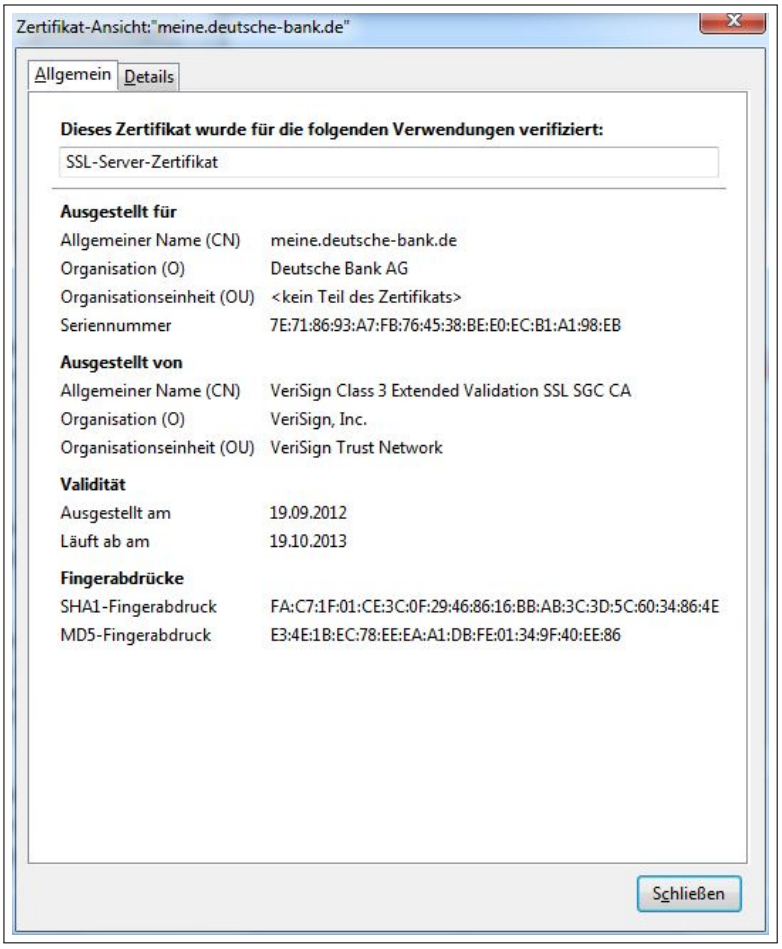


Abbildung 3.10: Anzeige eines Zertifikates im *Firefox*-Browser

Bank generiert und der Kunde erhält einen Ausdruck per Post zugeschickt. Transaktionsnummern sind mindestens sechsstellig und daher von einem Angreifer nicht so ohne weiteres zu erraten. Während es früher ausreichte eine beliebige, unverbrauchte TAN zur Autorisierung einzugeben wird heute ein weiterer Schutzmechanismus eingesetzt. Die TANs auf der Liste sind nummeriert und bei einer Transaktion verlangt der Internetbankingserver immer eine bestimmte TAN, beispielsweise TAN Nr. 17.

Alternativ kann die TAN von der Bank auch bei Bedarf generiert werden und dem Kunden dann als SMS auf das Mobiltelefon geschickt werden. Ein Angreifer muss nun zwei Kommunikationswege abhören, bzw. manipulieren können. Weitere Alternativen sind Verfahren, bei denen die Bank dem Kunden ein Gerät zur Erzeugung der TAN zur Verfügung stellt. In dieses Gerät muss man seine ec-Karte (mit Chip) schieben und ggf. noch transaktionsspezifische Daten eingeben. Das kann noch einmal die Kontonummer des Empfängers sein oder eine Kontrollnummer sein, die die Banking-Anwendung auf dem Bildschirm anzeigt, oder man „fotografiert“ mit dem Generator ein flickerndes Bild, das die Banking-Anwendung anzeigt.

2. Versucht ein Angreifer, die Werte der PIN oder TANs der Reihe nach auszuprobieren, so sperrt das System den Zugriff nach einer vorgegebenen Zahl (häufig 3) von falschen Eingaben. Der Angreifer kann also keinen

direkten finanziellen Schaden anrichten. Allerdings kann er damit zumindest den Dienst für den rechtmäßigen Benutzer nicht mehr verfügbar machen.

3. Sollte es trotz dieser Schutzmaßnahmen einmal zu Missbrauch kommen, so ist der Schaden i. d. R. begrenzt. Dies erreicht man durch die Vereinbarung eines Limits. An einem Tag können also keine Verfügungen getroffen werden, die in der Summe das Limit übersteigen.

Cookies: Das im Internet überwiegend verwendete Protokoll (http) ist zustandslos. Das bedeutet, dass bei einer Folge von Anfragen kein Zusammenhang zwischen diesen Anfragen herstellbar ist. Der Webserver weiß bei der zweiten Anfrage also nicht, auf welche vorhergehende Anfrage sie sich beziehen könnte. Öffnet ein Webbrowser also eine Seite mit Warenangeboten und anschließend eine „Kassenseite“ auf demselben Server, dann muss der Server wissen, welche Waren vorher ausgewählt wurden. Alleine mit Hilfe von http geht das nicht!

Cookies

Abhilfe schaffen die sog. **Cookies**. Ein Cookie ist ein Paar aus Variablenname und Wert. Sie werden vom Webserver gesetzt und auf dem Clientrechner gespeichert. Beim nächsten Zugriff auf den Server übermittelt der Client auch das Cookie an den Server. Dieser kann dadurch den Client „wiedererkennen“ und entsprechend handeln.

Möchte ein Webserver, dass der Browser bestimmte Informationen speichern soll, so schickt der Webserver im Kopf (engl. **header**) seiner Antwort (engl. **response**) eine `Set-Cookie`-Zeile mit. Diese hat folgende Form:

Cookie-Syntax

```
Set-Cookie: NAME=WERT; [expires=DATUM;] [path=PFAD;]
[domain=DNSNAME;] [secure]
```

Dabei werden Wörter in `GROßBUCHSTABEN` durch konkrete Zeichenketten ersetzt. Angaben in eckigen Klammern `[]` sind optional. `NAME` ist der Bezeichner des Cookies und `WERT` der zugeordnete Wert.

Wird `expires` gesetzt, so ist das dort angegebene Datum das Ende der Lebensdauer des Cookies. Nach Ablauf wird es verworfen. Wird auch ein Pfad angegeben, so wird das Cookie nur dann an den Webserver zurückgeschickt, wenn der Benutzer eine URL anfordert, in der der Anfang des Pfads mit `PFAD` übereinstimmt. Mit der Angabe einer DNS-Domäne kann man steuern, an welche Webserver das Cookie später zurückgeschickt wird. Hier muss das Ende des Webservernamens mit `DNSNAME` übereinstimmen. Endet die `Set-Cookie`-Zeile mit der Zeichenkette `secure`, dann wird das Cookie nur dann an einen Webserver zurückgeschickt, wenn die Verbindung mit SSL (siehe Abschnitt 3.3.1) verschlüsselt wird.

Nachdem der Browser das Cookie empfangen hat und der Benutzer eine passende URL anfordert, schickt der Browser im Kopf seiner http-Anfrage eine `Cookie`-Zeile mit an den Webserver. Sie hat folgendes Format:

```
Cookie: NAME1=WERT1; NAME2=WERT2; ...
```

Beispiel

Ein (fiktives) Beispiel: Der Webserver `www.fernuni-hagen.de` schickt in einer Antwort folgende Zeile im Kopf mit:

```
Set-Cookie: student=Wohlfeil; path=/; domain=fernuni-hagen.de;
```

Folgt nun eine http-Anfrage an einen beliebigen Webserver (beispielsweise `www.fernuni-hagen.de` oder `wwwpi6.fernuni-hagen.de`), dessen Name auf `fernuni-hagen.de` endet, und auf einen Pfad, der mit `/` beginnt[7], so enthält der Kopf des HTTP-Requests auch die folgende Zeile:

```
Cookie: student=Wohlfeil;
```

Damit Browser-Programmierer nicht beliebig viel Speicher für Cookies vorhalten müssen sollten Web-Server von folgenden Einschränkungen ausgehen: Einschränkungen

- Ein Browser speichert nicht mehr als 3000 Cookies.
- Ein Cookie darf nicht größer als 4 KB sein.
- Pro DNS-Domäne werden nicht mehr als 50 Cookies gespeichert.

Die genannten Werte sollte jeder Browser mindestens einhalten, er darf aber auch gerne mehr als 50 Cookies pro DNS-Domäne speichern. Weitere Details zur Spezifikation von Cookies finden Sie im RFC 2109 im WWW unter

`http://www.w3.org/Protocols/rfc2109/rfc2109`

oder in RFC 2965 oder RFC 6265. Bei dem RFC 2965 handelt es sich um eine Weiterentwicklung der ursprünglichen Spezifikation der Firma *Netscape*.

Mit Hilfe von Cookies kann ein Betreiber einer großen Webseite die Besucher wiedererkennen und damit Statistiken über das Verhalten der Besucher erstellen. Dadurch lassen sich Interessen und Vorlieben (z. B. im Internetbuchladen werden überwiegend Krimis nachgefragt, beim Nachrichtenanbieter werden überwiegend Sport- und Börsennachrichten angeschaut usw.) der Besucher erfahren. Dies erscheint zunächst nicht weiter kritisch. Diese Informationen über die Besucher der Webseite lassen sich aber evtl. auch mit anderen Datenquellen (z. B. Telefon- und Adressbücher) abgleichen. Solche Datensammlungen haben für die Werbewirtschaft immensen Wert. Damit kann „zielgruppenorientiertes Marketing“ betrieben werden, d. h. dass man seine Werbemaßnahmen nur noch an potentielle Interessenten richtet und dadurch Kosten sparen kann. Gefahren durch Cookies

Übungsaufgabe 3.4 *Welche weiteren Sicherheitsrisiken existieren, wenn Cookies dazu benutzt werden, einen Benutzer zu erkennen?*

Möchten Sie also, dass solche Datensammlungen nicht so einfach möglich sind, dann sollten Sie den Einsatz von Cookies in Ihrem Browser kontrollieren. Cookies werden meistens in normalen Dateien gespeichert. *Netscape* unter UNIX speichert Cookies im Heimatverzeichnis des Benutzers in einem versteckten Verzeichnis `.netscape/cookies` ab. Unter Windows NT stehen Cookies häufig in der Datei `cookies.txt`. Sie kann je nach Konfiguration des Systems (Mehrbenutzer vs. Einzelplatz) in unterschiedlichen Verzeichnissen stehen. Cookies kontrollieren

Möchte man, dass vom eigenen Browser keine Cookies verschickt werden, so hat man zwei Möglichkeiten:

1. Man konfiguriert den Browser so, dass keine Cookies verschickt werden.
2. Man sorgt dafür, dass das Betriebssystem den Zugriff auf die Cookie-Datei verhindert.

[7] Das ist für jeden Pfad der Fall.

Zu 1. Im *Firefox*-Browser kann man unter dem Menü „Bearbeiten → Einstellungen“ einen Konfigurationsdialog (siehe Abbildung 3.11, links) starten. Man kann nun alle Cookies blockieren oder nur Cookies erlauben, die nur an den absendenden Webserver zurück geschickt werden dürfen oder speziellere Einstellungen zur Privatsphäre (engl. **privacy settings**) tätigen. Außerdem kann man eine Warnung aktivieren, so dass jedesmal, wenn ein Webserver einen Cookie setzen will, eine Dialogbox mit einer Rückfrage erscheint. Sie zeigt, welche Informationen der Webserver in dem Cookie speichern möchte. Nun können Sie entscheiden, ob Sie das Cookie akzeptieren oder ablehnen.

Man kann zusätzlich Ausnahmen (siehe Abbildung 3.11, rechts) festlegen, also URLs, von denen man keine Cookies akzeptieren will. Über den Button „Cookies anzeigen“ bekommt man eine Liste mit allen Informationen zu den Cookies der einzelnen Webserver die im Firefox gespeichert sind (siehe Abbildung 3.11, unten).

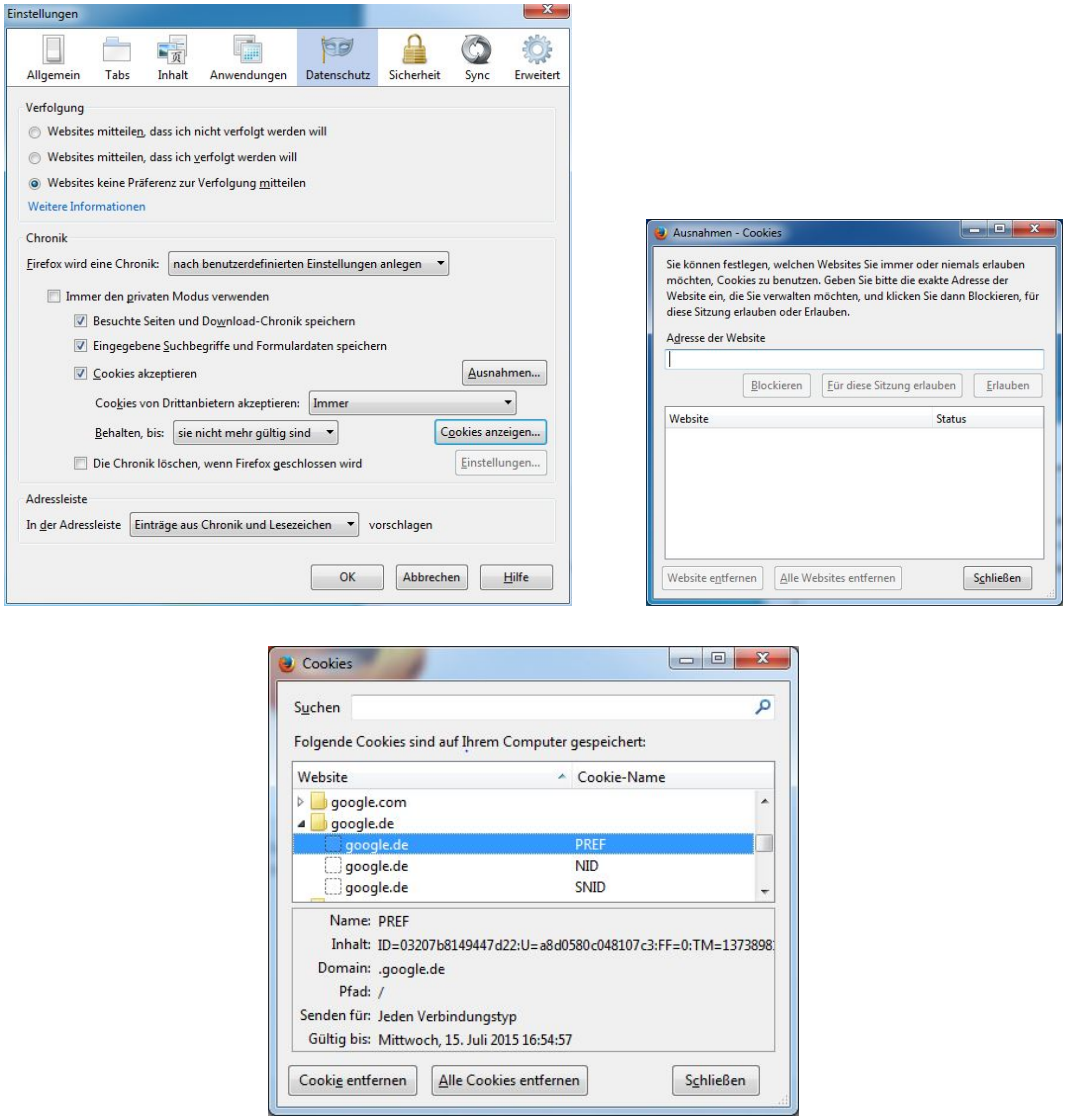


Abbildung 3.11: Konfiguration der Akzeptanz von Cookies in Firefox

Die Webseite `google.de` hat einen Cookie mit dem Namen `PREF` auf den Wert `ID=03207b8...` gesetzt. Über diese ID wird google verschiedene Suchanfragen dieses Benutzers einander zuordnen können.

Zu 2. Unter UNIX kann man eine leere Cookie-Datei erstellen und sich dann

selbst die Schreibrechte auf diese Datei entziehen. Dann kann der Webbrowser ein Cookie zwar akzeptieren, er kann es aber nicht dauerhaft speichern. Alternativ löscht man die Cookie-Datei und legt dann einen Verweis (engl. **link**) auf `/dev/null` unter dem Namen der Cookie-Datei an. Dann wandern alle Cookies direkt in den Papierkorb.

Weitere Entwicklungen

Die Einstellmöglichkeiten bzgl. Cookies in Webbrowsern haben sich in den letzten Jahren zwar deutlich verbessert, sind allerdings immer noch nicht ausreichend. Die Benutzer sollten nicht nur entscheiden können, von welchen Webservern sie Cookies akzeptieren wollen und von welchen nicht. Zusätzlich möchte man als Benutzer auch anhand der beabsichtigten Verwendungszwecke unterscheiden, ob man Cookies erlauben will bzw. Daten über sich preisgeben möchte. Einem Webserver will man seine Adresse beispielsweise anvertrauen, wenn sie ausschließlich dazu benutzt wird, bestellte Waren dorthin zu liefern. Vielleicht möchte man aber nicht, dass die Adresse für Werbemaßnahmen weiter verkauft wird.

P3P

Das World-Wide-Web-Konsortium arbeitet an der Spezifikation einer Plattform zur Spezifikation von „Privacy“-Vorlieben (engl. **platform for privacy preferences**). Sie ist seit Juli 2004 in der Version 1.1 verfügbar. Diese Spezifikation erlaubt es Webservern, die von ihnen verwendeten Praktiken in Bezug auf Privatsphäre klar und eindeutig zu beschreiben. Diese Beschreibung ist einerseits maschinenlesbar, also automatisch zu verarbeiten, und andererseits auch für einen Menschen lesbar. Benutzer oder die in ihrem Namen agierenden Webbrowser können anhand dieser Beschreibung und den gespeicherten Vorgaben des Benutzers dann entscheiden, welche Daten sie übermitteln möchten und welche nicht. Darüber hinaus können Browser auch dynamisch das Speichern von Cookies erlauben oder verhindern.

Diese Spezifikation umfasst allerdings *nur* die technischen Details, d. h. die Dinge die erforderlich sind, damit die Beschreibungen verständlich sind. Konkret sind das (1) die XML-Schemata, die die Syntax der Beschreibung definieren, (2) eine Semantikbeschreibung und (3) die Festlegung eines HTTP-basierten Übertragungsprotokolls für die Beschreibungen.

Sie umfassen allerdings *nicht* die Frage, ob sich der Betreiber eines Webservers an seine beschriebenen Praktiken hält oder nicht. Unter der URL

`http://www.w3.org/TR/P3P11`

steht die aktuelle Version des Standards.

Ein weiteres Problem des Standards besteht darin, dass es viele verschiedene Benutzungsmöglichkeiten für möglicherweise unterschiedlich sensible private Daten gibt. Genauso komplex können die Wünsche und Vorlieben von Benutzern im Hinblick auf Privatheit der eigenen Daten sein. Es ist also schwer und aufwendig, solche Beschreibungen zu lesen oder zu verstehen oder gar selbst zu erstellen.

Zur Zeit (August 2013) gibt es kaum Programme, die mit diesem Standard umgehen können. Benutzer müssen selbst entscheiden, wo sie welche Daten eingeben und welche Cookies sie (bzw. der Browser des Benutzers) speichern oder nicht.

3.3.3 Zusammenfassung: Sicher Surfen

Als Teilnehmer im Internet ist man verschiedenen Bedrohungen ausgesetzt. Auch wenn man nur Informationen sucht, muss man darauf achten, dass man die korrekten Informationen findet. Man muss sich über die Identität des „Informationslieferanten“ im Klaren sein. Außerdem muss man sicher sein, dass die Informationen korrekt, d. h. unverändert, bei einem ankommen. Man kann das heute schon durch den Einsatz von SSL sicher stellen. Sie sollten also, wann immer es möglich ist, Verbindungen mit SSL sichern.

SSL einsetzen!

Das hilft auch, wenn man Informationen senden will, z. B. Bestellungen oder ähnliche Transaktionen. Hier sollten Sie vertrauliche Daten wie Ihre Kreditkartennummer auf keinen Fall ungesichert übertragen. Anbieter, die keine SSL-Verbindungen erlauben, zeigen, dass sie es mit der Sicherheit nicht richtig ernst nehmen. Wenn Alternativen vorhanden sind, dann sollten Sie sie nutzen.

Allerdings ist der Einsatz von SSL nur eine notwendige Bedingung für einen guten Anbieter. Mit SSL sind die Daten nur auf dem Übertragungsweg geschützt. Speichert der Anbieter die sicher erhaltenen Daten dann auf einem ungesicherten und jedermann zugänglichen Rechner, so ist einem Missbrauch der Daten Tür und Tor geöffnet.

Weiterhin sollte man als Surfer im Web darauf achten, dass man nicht zu viel von seiner Privatsphäre preisgibt. Überlegen Sie sich immer genau, ob sie beliebigen Webservern das Speichern von Cookies auf Ihrem Rechner erlauben wollen. Sicherlich ist es in bestimmten Fällen sinnvoll, wenn Cookies erlaubt sind.

Auf Cookies achten!

Als sicherheitsbewusster Surfer sollten Sie auf jeden Fall nachschauen, in welcher Datei in Ihrem Rechner Cookies gespeichert werden. Es schadet auch nichts, wenn Sie hin und wieder einmal einen Blick in diese Datei werfen und mit einem beliebigen Editor einige Cookies löschen. Es ist auch nicht verkehrt, wenn Sie in Ihrem Browser die Warnmeldungen beim Speichern eines Cookies einschalten. So bekommen Sie auch einmal mit, welche Server wie viele Cookies setzen.

3.4 Zugriff auf entfernte Rechner

In der Praxis kommt es immer wieder vor, dass man nicht an seinem lokalen Computer arbeitet, sondern auf einem entfernt stehenden Computer Programme laufen lässt. Das kann zum einen daran liegen, dass man die betreffenden Programme nur auf dem entfernten Rechner zur Verfügung hat. Oder der entfernte Computer ist wesentlich leistungsfähiger als die eigene Maschine und soll deshalb benutzt werden.

Warum?

Ein anderer wichtiger Grund für den Zugriff auf entfernte Computer ist die zentrale Administration. Ein Administrator ist oft für viele Computer zuständig. Es ist sehr wünschenswert, dass der Administrator nicht wegen jeder Kleinigkeit erst einmal zum betroffenen Computer gehen muss (der Computer könnte auch am anderen Ende der Stadt oder des Landes stehen). Besser ist es, wenn sich der Administrator von seinem Arbeitsplatz aus auf dem entfernten Rechner anmelden kann. Dann kann die Wartungsarbeit deutlich schneller und effizienter erledigt werden.

Der Internetdienst Telnet (siehe Abschnitt 1.3.4) ist dafür im Prinzip

hervorragend geeignet. Leider ist es so, dass bei der Anmeldung auf einem entfernten Rechner mit Hilfe von Telnet der Benutzername, das Passwort und auch alle anderen Daten unverschlüsselt übertragen werden. Jeder, der diese Übertragung abhören kann, kann sich dann später auch auf dem entfernten Rechner anmelden. Eine Anmeldung als Administrator ist natürlich besonders gefährlich, da Administratoren i. d. R. mehr Rechte als „normale“ Anwender haben.

Telnet und seine Risiken

3.4.1 Sichere Verbindung mit SSH

Die *Secure Shell (SSH)* wurde ursprünglich von Tatu Ylonen entwickelt. Mit SSH kann man sich sicher auf einem entfernten Rechner anmelden und dort dann sicher arbeiten. Konkret bedeutet dies, dass die Authentisierung durch *Public-Key*-Verschlüsselungsverfahren erfolgt und dass die anschließende Kommunikation mit einem Sitzungsschlüssel symmetrisch verschlüsselt wird.

Heute existieren zwei Versionen von SSH, nämlich SSH 1 und SSH 2. SSH 1 ist die ursprüngliche Version und SSH 2 ist eine Neuimplementierung mit einigen Verbesserungen und Erweiterungen. Die Version 2 von SSH wurde von der Firma *SSH Communications Security Ltd.* entwickelt. Da in Version 1 inzwischen Sicherheitsprobleme gefunden wurden, sollte man nur noch SSH 2 benutzen.

Daneben gibt es eine freie Implementierung der SSH-Funktionen. Im Rahmen des OpenBSD-Projektes wurde OpenSSH entwickelt. Es bietet im Prinzip dieselben Funktionen wie SSH, verzichtet aber auf patentierte Algorithmen wie beispielsweise IDEA. OpenSSH unterstützt beide Varianten, SSH 1 und SSH 2. Im Internet finden Sie weitere Informationen unter den URLs

OpenSSH

`http://www.openssh.org/` `http://www.ssh.com/`

Das SSH-Protokoll besteht aus vier Teilen [BSB05]:

SSH Komponenten

1. Das SSH Transport Layer Protokoll (SSH-TRANS). Diese Komponente ist für das Aushandeln von Verschlüsselungs- oder Hash-Algorithmen zuständig. Weiterhin findet hier die Authentisierung des Servers statt und für den Datentransport werden in diesem Teil die Operationen wie Komprimierung, Verschlüsselung und Integritätssicherung durchgeführt.

2. Das SSH Authentication Protocol (SSH-AUTH). Dieser Teil befasst sich mit der Authentisierung der Clients. Hierzu bietet das Protokoll verschiedene Verfahren an, z. B. Authentisierung mit Benutzerkennung und Passwort oder Authentisierung mit asymmetrischen Schlüsselpaaren (RSA-Authentisierung). Außerdem ist hier auch spezifiziert, wie man weitere Authentisierungsverfahren an SSH anbinden kann.

3. Das SSH Connection Protokoll (SSH-CONN). Mit Hilfe dieser Komponente können neben einer sicheren Shell-Verbindung auch zusätzliche „Tunnel“ realisiert werden. Ein Tunnel ist wie ein virtueller Kanal (engl. **channel**), durch den – parallel zu den Shell-Kommandos – auch andere Daten sicher übertragen werden können.

4. Das SSH File Transfer Protocol (SSH-SFTP). Hier sind die Mechanismen zum sicheren Kopieren von Dateien, bzw. für den sicheren Zugriff auf ein entferntes Dateisystem realisiert.

Überblick SSH

Um SSH zu benutzen, sind ein Serverprogramm (es läuft auf dem entfernten Rechner) und ein Clientprogramm sowie einige Hilfsprogramme erforderlich. Das Clientprogramm beginnt den Verbindungsaufbau, indem es eine Nachricht an das Serverprogramm schickt. Der Server akzeptiert die Verbindung und schickt einen Identifikations-String an den Client. Dadurch wird überprüft, ob der Verbindungsaufbau auch zum richtigen Serverprogramm erfolgt ist. Weiterhin enthält die Identifikation auch Informationen über die Software- und Protokollversionen.

SSH Ablauf

Parameter aushandeln

Als erstes einigen sich Client und Server auf Verschlüsselungs-Algorithmen, Hash-Algorithmen und passende Schlüsselaustauschverfahren. In der Regel wird Diffie-Hellman benutzt, um ein gemeinsames Geheimnis g auszuhandeln. Aus diesem gemeinsamen Geheimnis werden dann die eigentlichen Sitzungsschlüssel abgeleitet.

Server Authentisierung

Bevor Client und Server aber sicher kommunizieren können, müssen sie sich gegenseitig authentisieren. Zuerst authentisiert sich der Server gegenüber dem Client. Dazu berechnen beide Seiten eine SSH-Session-ID aus dem Geheimnis g, beispielsweise indem sie einen Hashwert $H(g)$ berechnen. Der Server signiert $H(g)$ mit seinem privaten Schlüssel und überträgt das Ergebnis an den Client. Der Client prüft die digitale Signatur mit Hilfe des öffentlichen Schlüssels des Servers und hat somit den Server authentisiert. Wichtig ist hierbei, dass weder der Client noch der Server eine bestimmte SSH-Session-ID erzwingen können. Das SSH-Protokoll legt allerdings nicht fest, wie der Client an den öffentlichen Schlüssel des Servers kommt. Für die sichere Funktion muss der Client aber den öffentlichen Schlüssel des Servers kennen.

Übungsaufgabe 3.5 *Welche Gefahr besteht, wenn ein Client beim Verbindungsaufbau dafür sorgen könnte, dass diese Verbindung eine bestimmte SSH-Session-ID (nennen wir sie x) bekommt?*

Client Authentisierung

Als nächstes muss sich der Client dem Server gegenüber authentisieren. Das kann auf unterschiedlichen Wegen (Passwort oder RSA-Authentisierung) erfolgen. In einer „normalen“ Terminal-Sitzung ist die Passwortauthentisierung der Regelfall. Beachten Sie dabei, dass das Passwort *nicht* im Klartext über das Netz geschickt, sondern mit einem Sitzungsschlüssel verschlüsselt wird. Dieser Sitzungsschlüssel leitet sich auch aus dem o. g. Geheimnis g ab. Den Sitzungsschlüssel kennen also nur der Client und der Server. Außerdem hat der Client den Server bereits authentisiert, so dass ein Benutzer sicher sein kann, dass das eigene Passwort nicht an einen SSH-Server eines Angreifers geschickt wird. Das ist auch der Grund, warum zwingend die Reihenfolge erst (1) Server-Authentisierung, dann (2) Client-Authentisierung eingehalten werden muss.

Das SSH-Protokoll sieht nicht nur einen Sitzungsschlüssel, sondern mehrere Sitzungsschlüssel vor. Beispielsweise werden Daten vom Client zum Server mit einem anderen Schlüssel verschlüsselt als die Daten vom Server zum Client. Außerdem werden die Sitzungsschlüssel regelmäßig neu ausgehandelt, so dass ein Angreifer nach einem eventuell erfolgreichen *brute force* Angriff auf einen Sitzungsschlüssel trotzdem nicht zu viele vertrauliche Daten entschlüsseln kann.

Um die RSA-Authentisierung (des Clients) einzusetzen, braucht der Client ein eigenes Schlüsselpaar aus privatem und öffentlichem Schlüssel. Mit dem Kommando `ssh-keygen` kann der Client dieses Schlüsselpaar erstellen. Nun kopiert der Client seinen öffentlichen Schlüssel auf den Server. Die privaten

Schlüssel bleiben auf den jeweiligen Rechnern und werden vor unbefugtem Zugriff geschützt. Neben den passenden Zugriffsrechten auf die Datei gehört dazu auch die symmetrische Verschlüsselung der privaten Schlüssel durch einen Passwortsatz. Die folgende Tabelle zeigt, welche Schlüssel wo bekannt sind.

Server-Seite	Client-Seite
öffentlicher Schlüssel (Server)	öffentlicher Schlüssel (Server)
privater Schlüssel (Server)	
öffentlicher Schlüssel (Benutzer wohlfeil)	öffentlicher Schlüssel (Benutzer wohlfeil)
	privater Schlüssel (Benutzer wohlfeil)

Wichtig ist, dass die öffentlichen Schlüssel unverfälscht auf die „andere“ Seite gelangen. Wenn der Benutzer das erste Mal eine SSH-Verbindung zum Server aufbaut, dann kann der öffentliche Schlüssel des Servers automatisch (und erst einmal nicht verschlüsselt) zum Client übertragen werden. In diesem Fall zeigt SSH dem Benutzer auf der Client-Seite den Fingerprint des öffentlichen Schlüssels des Servers an. Der Client sollte den Fingerprint prüfen und dann die Korrektheit bestätigen. Nun wird der öffentliche Schlüssel des Servers auf dem Client gespeichert. Seinen eigenen öffentlichen Schlüssel kann der Benutzer später beispielsweise mit Hilfe von SSH auf den Server kopieren. Bevor das passiert ist, steht natürlich nur die Authentisierung durch ein Passwort zur Verfügung. Der öffentliche Schlüssel des Benutzers wird auf dem Server im „Heimatverzeichnis“ (engl. **home directory**) des Benutzers gespeichert.

Die Idee der RSA-Authentisierung ist wie folgt: Der Server kennt den öffentlichen Schlüssel des Clients und generiert eine Herausforderung (engl. **challenge**). Sie wird mit dem öffentlichen Schlüssel des Clients verschlüsselt und vom Server an den Client geschickt. Der Client beweist seine Identität, indem er die Herausforderung lösen kann, d. h. er kann sie korrekt entschlüsseln.

Nach der Authentisierung kann die Shell für den Client auf dem Server gestartet werden. In der nun folgenden Sitzung werden Daten in beiden Richtungen übertragen. Möchte der Client die Shell beenden, dann schickt er eine „exit“-Nachricht. Der Server merkt, dass die Shell beendet ist und schließt anschließend die Verbindung zum Client.

Die folgende Tabelle zeigt, welche Algorithmen und Verfahren in SSH implementiert sind: SSH Algorithmen

	SSH 1	SSH 2	OpenSSH
Server-Authentisierung	RSA	RSA, DSA	RSA, DSA
Client-Authentisierung	Passwort	Passwort	Passwort
	RSA	RSA, DSA	RSA, DSA
symmetrische Verschlüsselung	3DES, DES	3DES, DES	3DES, Cast-128
	IDEA, RC4	IDEA, RC4	AES, Arcfour
	Blowfish	Blowfish	Blowfish

Da in SSH die Daten vor der Verschlüsselung noch komprimiert werden können, ist die Übertragung trotz Verschlüsselung oft noch schneller als ohne SSH.

Mit SSH kann man nicht nur sichere „Telnet“-Sitzungen durchführen, sondern auch einige andere Anwendungen absichern: SSH-Funktionen

SCP: Mit dem Hilfsprogramm *scp* kann man sicher Dateien von einem Rechner auf einen anderen Rechner im lokalen Netz kopieren.

SFTP: Seit der Version 2 von SSH wird auch eine sichere Version *sftp* des File-Transfer-Protocol (*ftp*) unterstützt. Damit kann man sicher von anderen Rechnern im Internet Dateien laden.

Das zentrale Problem bei der Administration von SSH liegt nun darin, die Serverschlüssel sicher zu verteilen. Am sichersten ist es im Moment noch, diese Schlüssel auf Disketten, CDs oder USB-Sticks persönlich zu transportieren. Da es sich um einen einmaligen Vorgang handelt, ist der Aufwand durch den Gewinn an Sicherheit gerechtfertigt.

Ein „Nachteil“ von SSH ist, dass man als Benutzer zunächst nur eine Kommandozeile auf dem entfernten Rechner zur Verfügung hat. Viele aktuelle Programme benutzen jedoch Fenster zur Darstellung von Informationen und eine Maus mit der das Programm bedient werden kann. Beides sind Dinge, die in einer Kommandozeile nur sehr begrenzt verfügbar sind. Mit dem Grafikprotokoll X11 des Massachusetts Institute of Technology (MIT) existiert schon seit langem eine Möglichkeit, grafische Programme auf entfernten Rechnern laufen zu lassen, während die Ein-/Ausgabe auf dem lokalen Rechner erfolgt. Auf dem lokalen Rechner läuft dann ein sog. X11-Server. Er zeigt die Fenster auf dem lokalen Display an, ordnet die Tastatureingaben den Fenstern bzw. den dahinter laufenden Programmen zu usw. Auf dem entfernten Rechner läuft dann ein sog. X11-Client, also ein Programm das die X11-Library-Funktionen benutzt.

Kommandozeile vs. Fenster

X11

Dem entfernten Programm ist es dabei egal, ob der X11-Server auf demselben Rechner läuft wie der X11-Cient oder nicht. Die Kommunikation zwischen X11-Client und X11-Server kann auch über ein Netz laufen. Auf dem X11-Client ist dazu eine Umgebungsvariable `DISPLAY` gesetzt. In dieser Variablen steht die Adresse des Rechners mit dem X11-Server sowie die Nummer des Displays (an einen Rechner könnten mehrere Bildschirme angeschlossen sein). Will nun ein X11-Client-Programm eine Ausgabe tätigen, so ruft es die X11-Ausgabefunktion auf. Was ausgegeben werden soll, steht in der Parameterliste. Wohin ausgegeben werden soll, erkennt die X11-Ausgabefunktion am Wert der Umgebungsvariablen. Steht in der Umgebungsvariablen die Adresse eines entfernten X11-Servers, so werden die Informationen verpackt, übers Netz an den X11-Server geschickt, dort ausgepackt und dann verarbeitet. Damit nun nicht jeder beliebige (entfernte) X11-Client auf einen (lokalen) X11-Server zugreifen kann, gibt es hierfür eine Zugriffskontrolle. Dazu existieren zwei Möglichkeiten:

X11 ist netzwerkfähig

1. Auf dem X11-Server erlaubt der Benutzer mit dem Kommando `xhost +X11Client-DNS-Name` dem Rechner mit dem DNS-Namen `X11Client-DNS-Name` den beliebigen Zugriff auf den X11-Server.

   An Stelle des DNS-Namens darf auch eine IP-Adresse angegeben werden. Das Kommando `xhost +` erlaubt dann jedem Rechner den Zugriff auf den X11-Server.

2. Ein X11-Server erzeugt beim Start ein sog. *Magic Cookie*. Das ist im Prinzip eine Zufallszahl die der X11-Server in der Datei `.Xauthority` im *Home-Directory* des Benutzers ablegt, der den X11-Server gestartet hat. Das ist i. d. R. der Benutzer, der am lokalen Rechner arbeitet.

   X11-Client-Programme auf entfernten Rechnern müssen nun den Inhalt dieser Datei kennen, um auf den X11-Server zugreifen zu können.

Diese Architektur ist aus IT-Sicherheitsgesichtspunkten sehr problematisch. Zunächst kann ein entfernter X11-Client einen X11-Server derart manipulieren,

X11-Sicherheitsprobleme

dass alle Tastatureingaben auf dem X11-Server zuerst auf dem entfernten X11-Client landen und danach (oder parallel) dem Programm zugeordnet werden, für das der Tastendruck tatsächlich gedacht war. Passwörter lassen sich so einfach ausspionieren. Das Kommando `xhost +` sollte man also nur zu Testzwecken und niemals im normalen Betrieb benutzen.[8]

Zum Zweiten werden alle Daten zwischen X11-Client und X11-Server im Prinzip unverschlüsselt übertragen. Ein Angreifer kann alles mitlesen oder manipulieren. Und Drittens ist der Authentisierungsmechanismus mit dem *Magic Cookie* kein besonders sicherer Mechanismus. Ein Angreifer könnte versuchen, das Cookie vorherzusagen oder zu raten.

Um also grafische X11-Programme sicher über ein Netz auf einem entfernten Rechner auszuführen, braucht man Verschlüsselungsfunktionen, wie sie beispielsweise SSH bietet. Man kann nun auf dem lokalen Rechner mit dem Kommando

X11 durch SSH tunneln

```
ssh -X entfernterRechner
```

eine Kommandozeile auf dem entfernten Rechner öffnen. Die Option „Groß-X“ sorgt zusätzlich dafür, dass ein SSH-Tunnel eingerichtet wird, dass auf dem entfernten Rechner die Umgebungsvariable `DISPLAY` passend gesetzt wird und der X11-Server auf dem lokalen Rechner über den Tunnel mit X11-Clients kommuniziert. In der Kommandozeile auf dem entfernten Rechner kann man nun ein X11-Client-Programm starten und bekommt die Ausgabe dieses Programms auf dem lokalen Rechner angezeigt. Mit dem Programm *xeyes*, gestartet auf dem entfernten Rechner, bekommt man dann ein Fenster auf dem lokalen Rechner geöffnet, in dem ein Augenpaar immer auf die Position der Maus auf dem lokalen Display schaut.

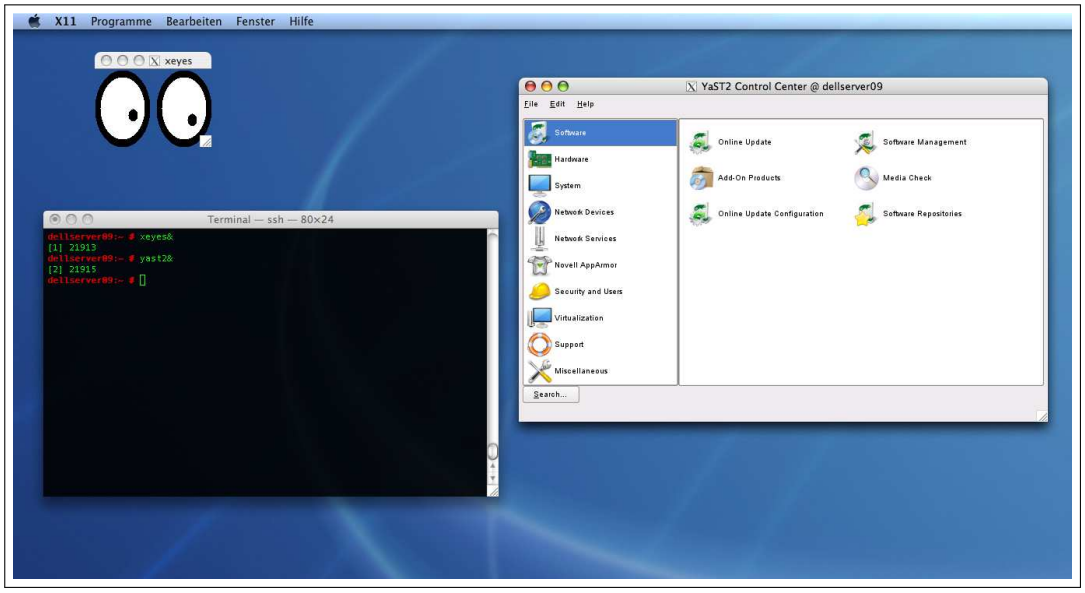


Abbildung 3.12: Beispiel eines entfernten Linux Programmes auf einem Mac OS X Desktop

Abbildung 3.12 zeigt einen Bildschirmausschnitt eines *Apple* Mac auf dem ein Terminal mit der SSH-Session zu sehen ist. Auf dem entfernten Linux-

[8] Außer Sie benutzen ein lokales und wirklich privates Netz, in dem sich nur Ihre Rechner befinden: das Netz funktioniert nur per Kabel, alle Kabel und Geräte befinden sich in Ihrer Wohnung und es existiert keine Verbindung ins Internet.

Rechner wurden das Programm *xeyes* und das *SuSE*-Administrationsprogramm *yast2* gestartet. Diese beiden Fenster werden angezeigt und die Informationen zwischen dem Mac und dem Linux-Rechner werden durch SSH verschlüsselt und sicher ausgetauscht.

Verfügbarkeit

Einen freien X11-Server bekommt man mit jeder Linux-Distribution automatisch mitgeliefert. Auch in *Apples* Mac OS X kann man einen X11-Server installieren. Er wird auf der Installations-DVD des Betriebssystems mitgeliefert oder kann von Apples Internetseiten geladen werden. Einzig die Betriebssysteme von *Microsoft* werden ohne X11-Server geliefert. Es gibt allerdings verschiedene frei verfügbare X11-Server-Programme auch für *MS Windows*.

3.4.2 Virtual Network Computing (VNC)

Mit SSH hat man ein sehr mächtiges und sicheres Werkzeug, um Kommandozeilenprogramme oder auch grafische Programme auf einem anderen Rechner auszuführen. Möchte man allerdings nicht nur einzelne Programme, sondern den kompletten Desktop des entfernten Rechners in einem Fenster auf seinem lokalen Rechner sehen, dann hilft SSH nicht mehr weiter.

Virtual Network Computing wurde vom *Olivetti Research Laboratory* zur Fernwartung von Rechnern über ein Netz entwickelt. Die Idee ist, dass man eine komplette Grafikkarte eines entfernten Rechners auf einem lokalen Rechner simuliert. Das RFB-Protokoll (engl. **Remote Framebuffer Protocol**) spezifiziert, wie die Kommunikation abläuft. Auf dem entfernten Rechner läuft ein VNC-Server, während ein VNC-Client auf dem lokalen Rechner läuft. Im Vergleich zum X11-Protokoll werden hier die Bezeichnungen *Client* und *Server* also gerade andersherum benutzt. Der Vorteil von VNC ist, dass es unabhängig vom Betriebs- bzw. Grafiksystem ist. Es läuft auf *MS Windows* genauso wie auf Linux genauso wie auf Mac OS X.

RFB-Protokoll

Das RFB-Protokoll ist so entworfen, dass die Implementierung eines Clients möglichst einfach wird. Der Client muss im Wesentlichen „nur“ Rechtecke mit einem Muster füllen können, Ereignisse aufnehmen und weiterleiten können (Mausbewegungen oder Tastatureingaben) und über ein TCP/IP-Netz kommunizieren können.

Verbindet man sich mit einem lokalen VNC-Client zu einem entfernten VNC-Server, so bekommt man ein Fenster auf dem lokalen Rechner angezeigt, dessen Inhalt dem Bildschirminhalt eines Monitors entspricht, der direkt am entfernten Rechner angeschlossen ist. Abbildung 3.13 zeigt einen VNC-Client auf einem Linux-Rechner, in dem der Login-Bildschirm eines anderen Linux-Rechners angezeigt wird.

Startet man auf einem (entfernten) Rechner einen VNC-Server, so startet man dort einen Dienst, der auf einer bestimmten Portnummer auf einen Verbindungsaufbau wartet. VNC wählt dazu die Nummer $5900 + x$, wobei x der Displaynummer des VNC-Servers entspricht. Mit der Displaynummer identifiziert man den VNC-Server, denn es kann mehrere davon auf einem Rechner geben. Auf einem UNIX-System mit X11-Server teilen sich X11 und VNC die Displaynummern. Normalerweise wird der X11-Server als erster gestartet und bekommt dann die Displaynummer 0. Ein danach gestarteter VNC-Server bekäme dann die Displaynummer 1 und wäre auf der Portnummer 5901 erreichbar.

Weil VNC über eine Portnummer erreichbar ist, kann man es einfach mit SSH

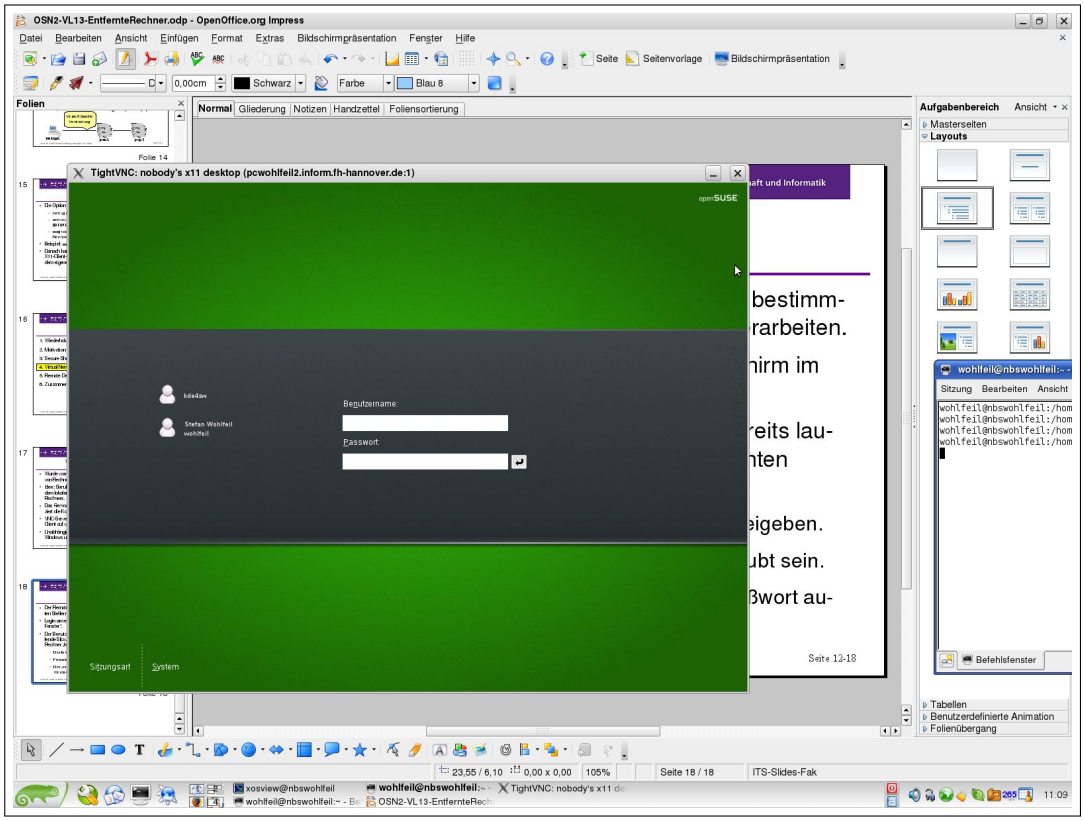


Abbildung 3.13: VNC-Verbindung zu einem entfernten Linux-Rechner

kombinieren und dadurch eine sichere Authentisierung und die Gewährleistung der Vertraulichkeit erreichen. Dazu muss auf dem entfernten Rechner (also dem Rechner mit dem VNC-Server) auch ein SSH-Server laufen. Dann kann man wie folgt vorgehen:

1. Man öffnet eine SSH-Verbindung zum Server, in der gleichzeitig auch ein SSH-Tunnel eingerichtet wird. Das Kommando hierfür lautet:

   ```
   ssh -L 6666:127.0.0.1:5901 entfernterRechner
   ```

   Es öffnet eine SSH-Verbindung zum Server `entfernterRechner` und es wird ein Tunnel eingerichtet, der auf dem lokalen Rechner an Portnummer 6666 endet. Diese Nummer kann im Prinzip frei gewählt werden. Am entfernten Ende der SSH-Verbindung wird der Tunnel dann an die Adresse `127.0.0.1:5901` weiterverbunden. Die IP-Adresse `127.0.0.1` bezeichnet dabei den entfernten Rechner selbst. Auf dem entfernten Rechner endet der Tunnel dann auf Portnummer 5901, dem VNC-Port.

2. Auf dem *lokalen Rechner* startet man einen VNC-Client und verbindet diesen mit dem Tunnelendpunkt aus Schritt 1 auf dem *lokalen Rechner*. Das Kommando auf einem Linux-Rechner hierfür lautet:

   ```
   vncviewer 127.0.0.1::6666
   ```

   Dem VNC-Client gibt man mit einem Doppelpunkt abgetrennt die DISPLAY-Nummer x des VNC-Servers auf dem entfernten Rechner an. Man kann auch die Portnummer $5900 + x$ direkt angeben, wenn sie wie im Kommando durch zwei Doppelpunkte abgetrennt ist. Für die willkürlich gewählte Nummer 6666 bietet sich dann die Schreibweise mit der Portnummer an.

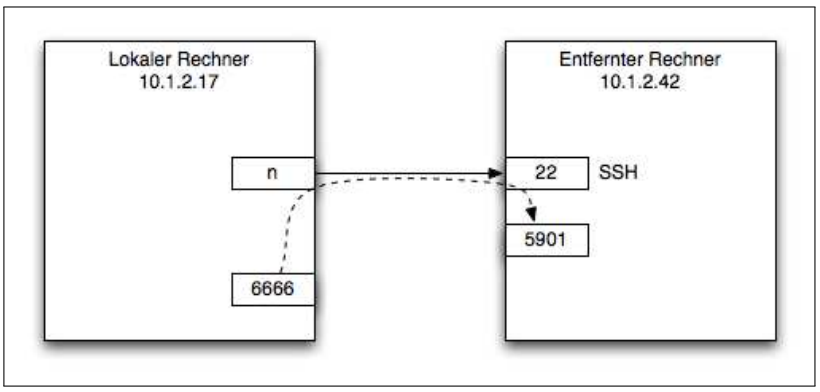


Abbildung 3.14: Beispiel VNC durch einen SSH-Tunnel

In Abbildung 3.14 wird gezeigt, wie man sich die Situation vorstellen kann. Vom lokalen Rechner wird die SSH-Verbindung zum entfernten Rechner aufgebaut. Normalerweise erwarten SSH-Server den Verbindungsaufbau auf Portnummer 22. Die o. g. Option erzeugt einen zusätzlichen Tunnel, der einmal auf Portnummer 6666 des lokalen Rechners und zum anderen vom SSH-Server des entfernten Rechners auf denselben Rechner und die Portnummer 5901 verbunden wird.

Einige Implementierungen von VNC-Clientprogrammen bringen diese Funktion direkt mit. Man muss also nicht mehr erst den Tunnel erstellen und dann den VNC-Client starten, sondern kann dem Client beim Start direkt mitteilen, dass ein SSH-Tunnel benutzt werden soll.

Man braucht auf der Client-Seite auf jeden Fall ein spezielles Programm, gerne auch *vncviewer* genannt. Möchte oder kann oder darf man auf dem lokalen Rechner aber keinen VNC-Client installieren, so gibt es trotzdem noch die Möglichkeit, VNC zu benutzen. Dazu braucht man auf der Client-Seite nur einen Webbrowser. Einige VNC-Server-Programme besitzen nämlich auch einen eingebauten Webserver. Dieser Webserver macht nichts anderes, als ein *Java-Applet* an den Client zu übertragen. Das *Java-Applet* ist dann der VNC-Viewer. Abbildung 3.15 zeigt, wie man dann den entfernten Rechner bedienen kann.

Sie können in Abbildung 3.15 sehen, wie im Firefox-Browser mehrere Tabs geöffnet sind. Im angezeigten Tab läuft das *Java-Applet*, das den Inhalt des VNC-Fensters im Browser anzeigt. Sie sehen dort einen typischen SuSE-Linux-Desktop.

3.4.3 Remote Desktop (rdesktop)

Remote Desktop Protocol (RDP)

Die Firma *Microsoft* hat ein eigenes Protokoll für den Zugriff auf Programme, die auf einem entfernten *MS Windows*-Server laufen, entwickelt. Das *Remote Desktop Protocol (RDP)* basiert auf den T.120-Protokollvorschlägen der *International Telecommunication Union (ITU)*. Diese Vorschläge wurden von *Microsoft* erweitert und in dem Programm *MS Windows Terminal Server* realisiert. Ein lokaler Client kann mit Hilfe dieses Protokolls:

1. die Ausgaben eines Programmes auf dem Server anzeigen.
2. Tastatur- und Maus-Ereignisse vom Client zum Server übertragen.

Funktionen

Die Protokollvorschläge erlauben mehrere virtuelle Känale zwischen Client und Server, so dass man auch von mehreren Clients ein einziges Serverprogramm

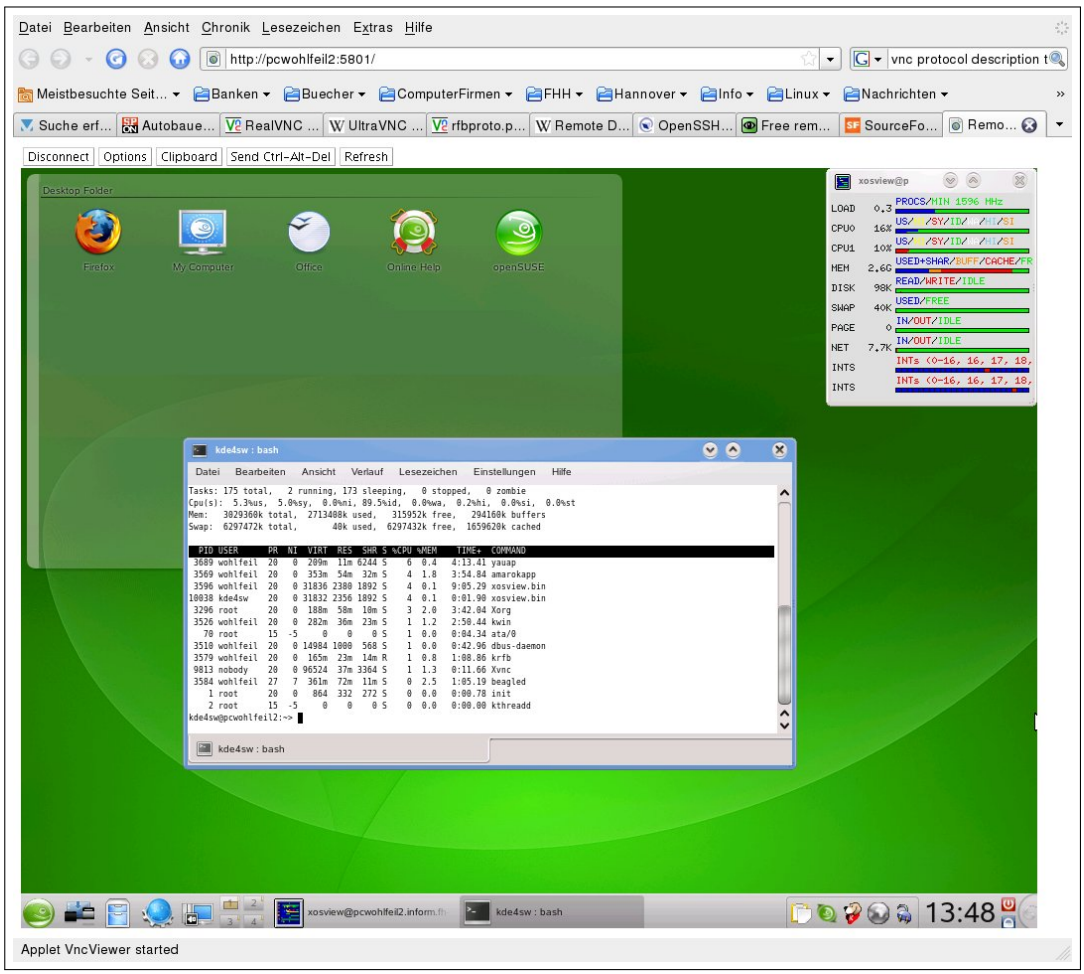


Abbildung 3.15: Beispiel VNC-Verbindung im Browser

betrachten und steuern kann. Damit wären auch virtuelle *White Boards* möglich, bei denen mehrere Benutzer (Clients) parallel auf eine gemeinsame Tafel (Serverprogramm) zugreifen. Ob die *Microsoft*-Implementierung diese Möglichkeit tatsächlich realisiert, ist im Einzelfall zu prüfen. Die virtuellen Kanäle können auch dazu benutzt werden, lokale Geräte (z. B. Drucker) dem Programm auf dem entfernten Server zugänglich zu machen oder Daten aus dem Programm auf dem entfernten Rechner durch „Copy und Paste“ in ein Programm auf dem lokalen Rechner einzufügen. *Microsoft* hat bei der Weiterentwicklung von RDP bisher die Schwerpunkte

- einfache Bedienbarkeit und
- Performanz

gesetzt. Benutzer sollen das entfernte Programm möglichst einfach und vollständig bedienen können. Dazu gehörte insb. der Wunsch nach „Copy und Paste“ zwischen lokal und entfernt laufenden Programmen sowie das Drucken aus einem entfernt laufenden Programm auf einem lokalen Drucker. Außerdem sollen Benutzer die zwangsläufigen Geschwindigkeitseinbußen möglichst wenig spüren. Daher wurden Maßnahmen ergriffen, die die erforderliche Bandbreite reduzieren, indem Daten komprimiert übertragen werden oder indem intensives *Caching* benutzt wird.

Sicherheitseigenschaften

Im Remote Desktop Protocol ist bereits vorgesehen, dass die Anzeigedaten

sowie die Tastatur- und Maus-Ereignisse verschlüsselt übertragen werden können. Dazu wird der Stromverschlüsselungsalgorithmus RC4 benutzt. Die Schlüssel können unterschiedlich groß sein. Man kann die Software allerdings auch so konfigurieren, dass nur der Datenverkehr einer Richtung (z. B. vom Client zum Server) verschlüsselt wird, während die andere Richtung im Klartext übertragen wird. Nach den Veröffentlichungen von Edward Snowden wurden Vermutungen geäußert, dass RC4 von der NSA in Echzeit „geknackt“ werden könne. Echte Belege hierfür gibt es aktuell (Stand: Februar 2014) allerdings nicht.

Implementierungen

Die Implementierung der Serverkomponente liefert *Microsoft* mit seinen Serverversionen von *MS Windows* aus. Die Clientkomponente ist für alle *MS Windows*-Versionen verfügbar. Es existiert auch eine Client-Implementierung für Linux unter dem Namen *rdesktop*. Damit kann man von einem lokalen Linux-Client aus Windows-Programme auf einem entfernten Windows-Server starten und bedienen.

3.4.4 SSL und die Kommandozeile

Wie bereits in Kapitel 1 gesagt, kann man mit dem Kommando `telnet` auch eine Verbindung zu einem konkreten Dienst auf einem Rechner aufbauen. Das Kommando `telnet pop.gmx.net 110` baut beispielsweise eine Verbindung zum POP3-Dienst auf dem Rechner `pop.gmx.net` auf. Dort kann man nun die POP3-Kommandos direkt eintippen und bekommt die Antworten des Servers direkt angezeigt.

Man muss sich als Benutzer aber natürlich authentisieren, was man bei POP3 durch ein Kommandopaar `user xxx` und danach `pass yyy` macht. Dabei werden die Benutzerkennung xxx und das Passwort yyy im Klartext durch das Internet übertragen und können leicht abgehört werden.

Man kann die POP3-Verbindung auch über eine SSL-Verbindung abwickeln. E-Mail-Clientprogramme können das bei entsprechender Konfiguration, ohne dass es dem Benutzer auffällt. Möchte man das nun auch in der Kommandozeile machen, so braucht man ein zusätzliches Programm, beispielsweise das Programm *openssl*. Bei den meisten Linux-Distributionen wird es mitgeliefert; *MS Windows*-Benutzer müssen es separat installieren.

Nach der Installation kann man mit dem Kommando

```
openssl s_client -connect pop.gmx.net:995
```

eine SSL-verschlüsselte Verbindung zum POP3-Dienst bei *GMX* aufbauen. Dieser Dienst wird bei *GMX* über die Portnummer 995 angeboten. Nun kann man die Kommandos im Klartext eintippen und *openssl* verschlüsselt die Daten, bevor sie an den Server übertragen werden.

Weiterhin kann man mit openssl auch beobachten, welche Algorithmen und Parameter beim handshake ausgehandelt werden. Der folgende Ausschnitt zeigt, wie ein Client und ein Server AES mit 256 Bit Schlüssel und SHA als Hash-Algorithmus ausgehandelt haben:

```
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: zlib compression
```

```
Expansion: zlib compression
SSL-Session:
    Protocol  : TLSv1
    Cipher    : AES256-SHA
    Session-ID: B731DBD1BBC885F36975E5BE67D6F5899AB14B1A9C0E...
    Session-ID-ctx:
    Master-Key: 1A0B9C2A288D5844FDAB53F3EDDDD655CF8DB534ADB0...
    Key-Arg   : None
```

Ausserdem haben sich beide Seiten darauf geeinigt, die Daten vor dem Verschlüsseln auch noch zu komprimieren und dabei die zlib zu benutzen.

Mit der Option `-cipher` kann man openssl auch mitgeben, dass nur bestimmte Algorithmen vom Client beim Client Hello an den Server geschickt werden. So kann man testen, ob ein Server perfect forward secrecy unterstützt. Dazu gibt man nur die beiden Diffie-Hellman-Varianten an, mit denen perfect forward secrecy möglich ist. Das Kommando hierfür lautet:

```
openssl s_client -cipher 'DH:ECDH' -connect server:portnummer
```

Dann steht oben in der Cipher-Zeile beispielsweise:

```
    Cipher    : DHE-RSA-AES256-SHA
```

In diesem Fall geht der Server davon aus, dass der Client nur diese beiden Schlüsselaustauschverfahren beherrscht und wählt daher einen davon aus.

Leider hat man heute (Stand Dezember 2013) keine Möglichkeit, eines dieser beiden Schlüsselaustauschverfahren zu erzwingen. Dafür gibt es im wesentlichen folgende Gründe:

1. Alle Browser-Programme und alle Web-Server-Programme müssen auch die Diffie-Hellman-Verfahren implementiert haben. Die aktuellsten Versionen der Programme sollten das können, aber manche Server-Betreiber haben noch ältere Versionen laufen. Dieser Grund sollte sich bald erledigt haben.

2. Normale Benutzer eines Browsers, die „nur“ im Web surfen wollen, sind mit der Konfiguration von Verschlüsselungsparametern i. d. R. überfordert. Die Programmierer der Browser bieten daher (für den „normalen“ Benutzer) keine solchen Konfigurationsmöglichkeiten an. Browser sollten von ihren Programmierern so konfiguriert werden, dass Diffie-Hellman präferiert wird. Auch das sollte bald überall so eingestellt sein.

3. Letztlich entscheidet aber der Server, welche Algorithmen beim Handshake ausgehandelt werden. Bietet der Client Diffie-Hellman und die anderen Verfahren an und der Server wählt kein Diffie-Hellman aus, dann wird es nicht benutzt. Bietet der Client nur Diffie-Hellman an und der Server will das nicht unterstützen, dann schlägt der SSL-Handshake fehl und es wird gar nicht verschlüsselt.

   Dieser Grund liegt außerhalb des Einflussbereichs des Benutzers. Man kann daher nur hoffen, dass immer mehr Server-Betreiber ihre Server so konfigurieren, dass auch sie Diffie-Hellman präferieren.

3.4.5 Zusammenfassung: Entfernte Rechner

Für den Zugriff auf entfernte Rechner sollte man niemals *telnet* benutzen, weil hier Benutzerkennung, Passwort und alle Daten unverschlüsselt übertragen werden. In der Praxis ist SSH inzwischen das Standardwerkzeug, um an entfernten Rechnern zu arbeiten. In SSH sind sichere Verschlüsselungsmethoden implementiert, die vertrauliche und authentische Kommunikation ermöglichen.

Zusammen mit einem X11-Server auf dem lokalen Rechner kann man auch einzelne grafische Programme auf dem entfernten Rechner einfach, schnell und sicher benutzen. In der Praxis reicht das für die meisten Anwendungsfälle aus. Die Geschwindigkeit ist selbst bei DSL-Verbindungen noch akzeptabel.

Möchte man zusätzlich auch den kompletten Bildschirm eines entfernten Rechners auf einem lokalen Rechner darstellen, so kann man Systeme wie *VNC* oder *Rdesktop* einsetzen. Da VNC keine Verschlüsselung anbietet und der Algorithmus von Rdesktop (RC4) nicht mehr sicher sein könnte, sollte man sie sicherheitshalber nur durch einen SSH-Tunnel benutzen. Da hierbei sehr viele Daten übertragen werden, ist ein sehr schnelles Netz besonders wichtig. Sonst sind die Reaktionen im VNC-Fenster so träge, dass man nicht mehr vernünftig arbeiten kann.

Bevor man SSH einsetzen kann, muss man Schlüsselpaare generieren und auf einem sicheren Weg auf den betroffenen Rechnern installieren. Bei großen Rechnerzahlen kann dieser Aufwand nicht unerheblich sein.

3.5 Schutz des privaten PCs

In diesem Abschnitt geht es darum, wie ein privater Benutzer seinen PC mit Internetzugang sichert. Der erste und wichtigste Schritt besteht darin, sich der Gefahren bewusst zu sein und bei allen Aktionen am eigenen Rechner, den gesunden Menschenverstand zu benutzen. Schadsoftware die man gar nicht startet, egal wie „nett“ oder trickreich sie den Benutzer dazu bringen will es doch zu tun, richtet auch keinen Schaden an.

Weiterhin gibt es aber auch verschiedene technische Teilaspekte, die zu beachten sind und in den folgenden Unterabschnitten besprochen werden. Immer dann, wenn man Dateien mit anderen austauscht, geht man das Risiko ein, dass eine Datei einen Virus oder andere Schadsoftware enthält. Dieser Virus sollte

Virenscanner

erkannt und möglichst auch gleich entfernt werden. Das Thema **Virenscanner** wird in Abschnitt 3.5.1 besprochen.

Wenn ein Rechner an das Internet angeschlossen ist, dann kann i. d. R. jeder andere Rechner im Internet eine Verbindung zu diesem Rechner aufbauen. Außerdem kann der angeschlossene Rechner selbst beliebige Daten an andere Rechner im Internet versenden. Die Kontrolle dieser Datenflüsse wird durch eine

Firewall

Firewall sichergestellt. In Abschnitt 3.5.2 werden Firewalls für den Einsatz auf PCs besprochen.

Allgemeine Hinweise zur sicheren Konfiguration von Windows-Systemen finden sich bei [WWS02]. Weiterhin ist es erforderlich, dass auch das Betriebssystem selbst sicher konfiguriert ist. In Abschnitt 3.5.3 wird besprochen, worauf man bei der Konfiguration von Windows 7 oder einem seiner Nachfolger achten sollte. Wird im Kurstext an einige Stellen von Funktionen für Windows 7 gesprochen, so sind diese Funktionen i. d. R. auch in den Nachfolgeversionen verfügbar. Schnappschüsse von Bildschirminhalten im Kurstext sind dann evtl.

nicht mehr aktuell. Es sollte aber einfach sein, in der aktuellen Windows-Version die Funktion in einem ähnlichen Fenster oder einer Kachel ausführen zu können.

3.5.1 Virenscanner

Immer wieder hat es während der letzten Jahre größere Probleme mit Viren, Würmern oder anderer Schadsoftware gegeben. Beispiele hierfür sind: Beispiele

Blaster: Dieser Wurm nutzte im Jahr 2003 eine Schwachstelle im RPC-Dienst von Windows aus. Er wurde auch *Lovesan* oder *MSBlaster* genannt und ist sozusagen ein Vorfahre von *Conficker*.

Beagle: Ein Wurm, der sich im Jahr 2004 per E-Mail verbreitete und parallel eine Hintertür im Betriebssystem Windows öffnete.

Conficker: Dieser Wurm hat im Jahr 2008 eine Lücke im Remote-Procedure-Call-Dienst von Windows ausgenutzt und sich weltweit verbreitet.

Man schätzt die Schäden, die durch diese Schadsoftware verursacht wurden, auf mehrere Milliarden Euro. Die Größenordnung erscheint plausibel, wenn man bedenkt, welche Auswirkungen ein Virus oder ein Wurm in der Geschäftswelt hat. Auswirkungen

- Alle Rechner in einer Firma müssen überprüft und ggf. muss der Virus/Wurm entfernt werden. Hierbei fällt Aufwand durch die Prüfung selbst an. Weiterhin steht der Rechner während dieser Zeit nicht mehr für seinen eigentlichen Zweck zur Verfügung.

- Um die Ausbreitung des Virus/Wurms zu stoppen, muss ein lokales Netz möglicherweise komplett vom Internet getrennt werden. Verbreitet sich ein Virus/Wurm per E-Mail, so müssen also evtl. die E-Mail-Server abgestellt werden. Hierdurch steht das Kommunikationsmedium E-Mail nicht mehr zur Verfügung. Dadurch fällt möglicherweise Umsatz weg. Weiterhin muss sich ein Administrator um den E-Mail-Server kümmern und die durch den Virus generierten Nachrichten löschen. Allerdings dürfen keine anderen Nachrichten gelöscht werden. Das macht diese Arbeit wiederum sehr aufwendig.

Wie kann man sich als Benutzer nun gegen Viren schützen? Hierbei muss man die verschiedenen Virentypen unterscheiden. Gegen **Bootsektor-Viren** kann man sich wie folgt schützen: Bootsektor-Viren

Reihenfolge der Bootgeräte beachten: Das **BIOS** (oder in neueren PCs das **UEFI**) eines PCs enthält Informationen über den Programmcode, der beim Einschalten gestartet werden soll. Normalerweise hat man hier die Auswahl zwischen verschiedenen Geräten, von denen der **Lader** gestartet werden soll. Hier sollte man die folgende Reihenfolge einstellen: BIOS UEFI Lader

1. Festplatte
2. DVD-Laufwerk
3. USB-Stick

Falls beim Einschalten des Rechners eine DVD mit einem Bootsektor-Virus im Laufwerk liegt oder ein verseuchter USB-Stick eingesteckt ist, so wird das nicht beachtet.

Immer dasselbe Bootmedium angeschlossen lassen: Eine andere Möglichkeit für den Heim-PC ist, wenn man grundsätzlich von derselben Boot-DVD oder demselben USB-Stick startet. Dieses Medium enthält dann möglicherweise einen **Bootmanager**, der es erlaubt verschiedene Betriebssysteme, die parallel auf der Festplatte installiert sind, zu starten. Dieses Medium sollte den Schreibschutz aktiviert haben und ständig angeschlossen sein.

Bootmanager

Zusätzlich sollte man ein „Rettungs-Bootmedium“ erstellen und bereithalten. Einige Zeitschriften bieten hierzu eine spezielle Linux-Distribution mit einem integrierten Virenscanner an. Mit diesem Bootmedium muss man den Rechner starten, wenn man befürchtet, dass der Rechner von einem Bootsektor-Virus befallen ist.

Dateiviren
Würmer

Gegen die wesentlich weiter verbreiteten **Dateiviren** und auch gegen **Würmer** gibt es die folgenden Schutzmechanismen:

Antivirensoftware wird von verschiedenen Herstellern (*Avast, Avira, Bitdefender, Kaspersky Lab, McAfee, Norman, Symantec* und viele mehr) angeboten. Diese Programme sind nicht sehr teuer und dürfen für den privaten Einsatz teilweise sogar kostenlos benutzt werden (*AVG* oder *Avira*). Im Internet oder verschiedenen Computerzeitschriften können Sie nachlesen, welche Versionen der Programme gerade aktuell sind und wie gut sie die folgenden wichtigen Eigenschaften einer Antivirensoftware erfüllen:

Eigenschaften

Erkennung der meisten bekannten Viren/Würmer: Das Programm sollte all die Viren/Würmer, die schon einmal weit verbreitet waren, erkennen. Es ist nicht unwahrscheinlich, dass sich diese Viren/Würmer über ungeschützte Rechner ein weiteres Mal in großem Stil verbreiten. Davor sollte die Antivirensoftware natürlich schützen.

Mehrere Betriebsmodi: Die Antivirensoftware sollte auf der einen Seite im Batch-Betrieb laufen können. Das bedeutet, dass der Benutzer die Antivirensoftware manuell starten kann und alle Dateien auf der lokalen Festplatte oder den angeschlossenen Laufwerken geprüft werden. In Abbildung 3.16 startet man die Prüfung durch Anklicken von „Computer scannen“.

Auf der anderen Seite sollte die Antivirensoftware auch im Hintergrund laufen können. Das bedeutet, dass die Software ständig läuft und jeden Dateizugriff überwachen kann. Neben Zugriffen auf die lokale Festplatte sollte die Software auch Netzzugriffe überwachen. In Abbildung 3.16 zeigt der Text „On Access Scans: Aktiviert“ im Statusbereich an, dass diese Überprüfungen stattfinden. Was bei der Konfiguration der On Access Scans eingestellt werden kann, zeigen die Fenster in Abbildung 3.17. Man kann einstellen, ob bei Schreibzugriffen, Lesezugriffen, usw. eine Überprüfung durchgeführt werden soll und welche Dateien (oder Dateitypen) denn konkret betrachtet werden sollen.

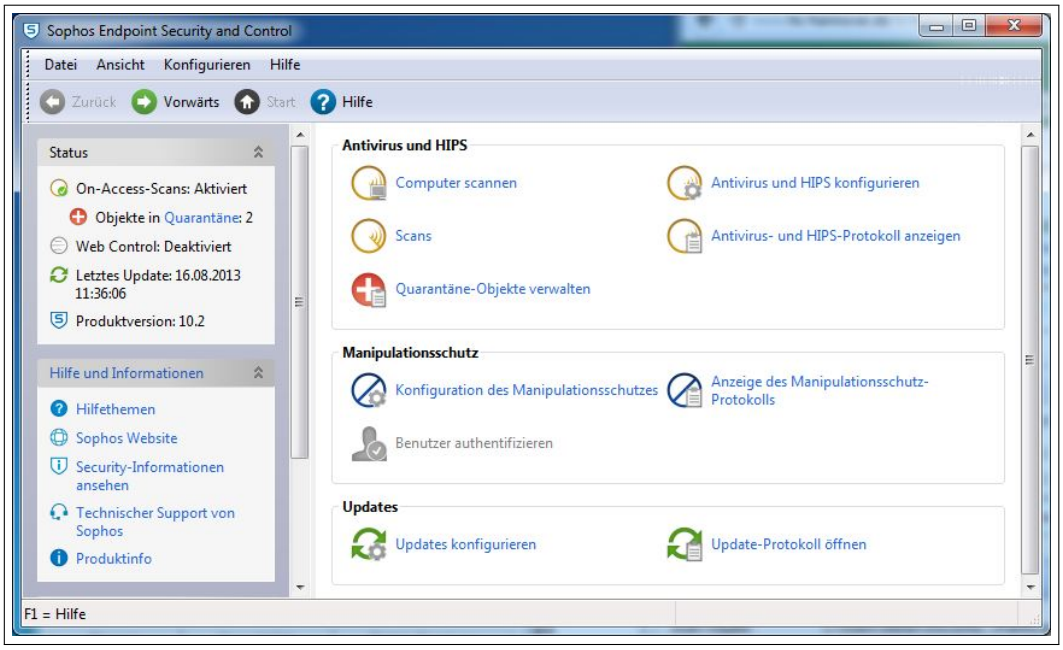


Abbildung 3.16: Startbildschirm der Sophos Antivirensoftware

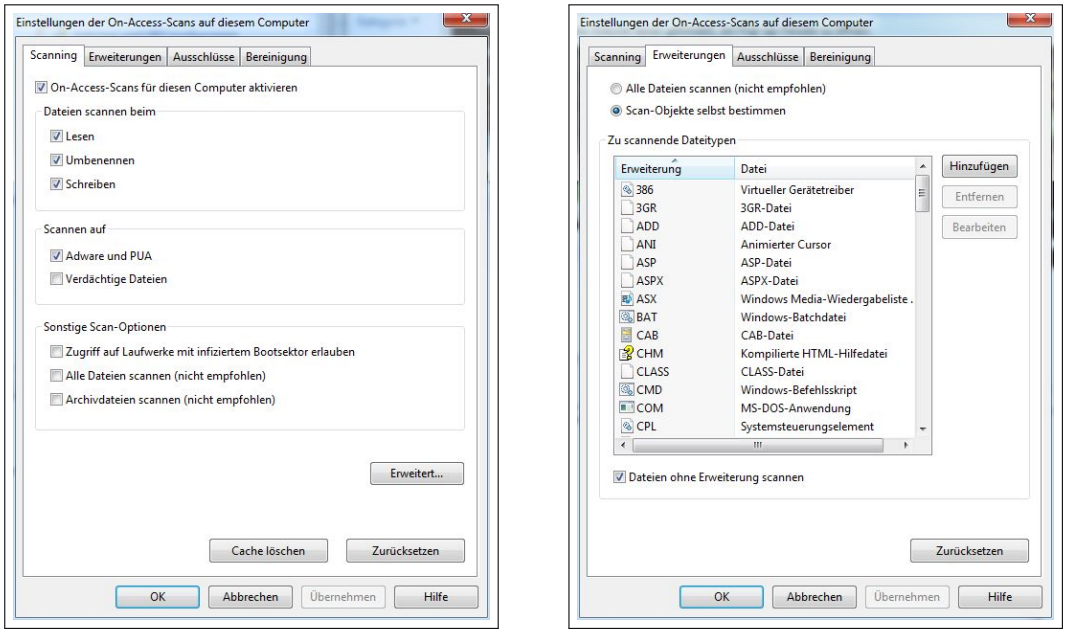


Abbildung 3.17: Sophos Antivirensoftware, Einstellungsoptionen für Hintergrundprüfungen

Automatische Updates über das Internet: Eine gute Antivirensoftware muss in der Lage sein, neue Viren zu erkennen. Daher ist es erforderlich, dass die Template-Dateien, in denen die Pattern beschrieben sind, an denen man einen Virus erkennt, von der Software automatisch aktualisiert werden können. Diese Pattern nennt man auch Virensignatur. Die meisten Hersteller bieten zu diesem Zweck einen Server im Internet an, von dem sich die Antivirenprogramme immer die aktuellen Templates laden können.

Virensignatur

Alternativ kann man so einen Server auch im lokalen Netz installieren. Dann braucht man nur diesen einen lokalen Server mit den aktuellen Templates über das Internet aktualisieren. Die einzelnen Clients verbinden sich dann mit diesem lokalen Server und aktualisieren die Templates. Dies spart Kosten für den Internetzugang. Abbildung 3.16 zeigt im Statusbereich, wann das letzte Update durchgeführt wurde.

Makrovirenschutz in Office-Programmen einschalten: Die aktuellen Office-Programme können so eingestellt werden, dass ein Schutz vor Makroviren gegeben ist. Diese Option sollte man einschalten.

Man muss hier allerdings beachten, dass dieser Schutz nur rudimentär ist. Die Mächtigkeit der Makrosprachen und das ActiveX-Konzept ermöglichen es bösartigen Programmen diese Option wieder zu deaktivieren, ohne dass der Benutzer es merkt.

Keine E-Mail-Anhänge öffnen, die ausführbare Programme enthalten: Diese Grundregel unterbricht die Verbreitung von vielen Viren oder Würmern. Ausnahmen von dieser Regel sollte man nur dann machen, wenn man wirklich sicher ist, dass das Programm keine Schadensfunktion enthält. Die Tatsache, dass man den Absender kennt und ihm vertraut, bedeutet *nicht*, dass die E-Mail sicher ist. Der Absender könnte selbst Opfer eines Virus oder Wurms geworden sein, das in fremdem Namen E-Mails versendet und sich dadurch verbreitet.

Besonders wichtig ist hier, dass auch das E-Mail-Programm entsprechend konfiguriert ist. Einige E-Mail-Programme wollen es dem Benutzer besonders einfach machen und starten Anhänge (engl. **attachment**), die Programme enthalten, automatisch. Diese Option sollte unbedingt deaktiviert sein.

3.5.2 Personal Firewalls

Eine Personal Firewall soll den privaten PC vor Angriffen aus dem Internet schützen, vergleichbar zu einer „richtigen“ Firewall, die ein Unternehmensnetz vor Angriffen aus dem Internet schützt. Das Grundprinzip einer Firewall wird später in Kapitel 4 erklärt. Dort werden Firewalls als eigenständige Systeme vorgestellt, die zwischen ein lokales Netz und das Internet geschaltet sind. Sie können somit jeglichen Datentransfer überwachen.

Ein typischer Privat-PC ist heute häufig über einen Router mit dem Internet verbunden. Je nach Internet Service Provider ist in den Router ein DSL-Modem, ein Kabel-Modem oder eine Funkkarte zusätzlich eingebaut. Auf diesem Router kann somit eine Firewall installiert werden. Sie sollte in der Standard-Einstellung jeglichen Verbindungsaufbau aus dem Internet zu Rechnern im lokalen Netz unterbinden. Verbindungsaufbau von innen ins Internet sollte

erlaubt sein. Auch ohne eigene Firewall sind PCs in diesem Szenario vor Angriffen aus dem Internet geschützt.[9]

Hat man sein internes Netz durch ein WLAN realisiert, oder muss man aus anderen Gründen damit rechnen, dass Angreifer auch an das interne Netz angeschlossen sein könnten, dann muss man die internen PCs zusätzlich schützen. Der Begriff **Personal Firewall** bezeichnet in diesem Fall ein Programm auf dem Privat-PC, das die Funktion einer Firewall möglichst gut nachbilden soll. Verschiedene Firmen (*Bitdefender, F-Secure, Kaspersky, Norman, Norton, Sophos, ZoneAlarm* und viele mehr) bieten solche Software an. Sie alle können als Paketfilter arbeiten, einige zusätzlich auch als Stateful-Inspection-Filter oder Application-Level-Gateway. Daneben liefern aber auch die Hersteller der Betriebssysteme inzwischen eine Firewall zusammen mit dem Betriebssystem aus. Benutzer von Windows, Mac OS X oder Linux haben automatisch auch eine Firewall auf ihrem Rechner. Der Zweck einer Personal Firewall besteht im Wesentlichen darin, die folgenden Funktionen zu erfüllen:

Personal Firewall

- Angriffe von außen auf den Privat-PC sollen unterbunden werden.
- Schädliche Programme, die möglicherweise auf den PC gelangt sind, sollen keine Daten vom PC in das Internet schicken können.
- Schädliche Programme werden erkannt, wenn sie auf dem Rechner installiert werden sollen.

Angriffe von außen: Eine Personal Firewall soll dabei helfen, Angriffe von außen zu verhindern. Diese Angriffe bestehen zunächst einmal einfach aus IP-Paketen, die an den Privat-PC geschickt werden. Diese enthalten als Adressangabe ein Paar (IP-Adresse, Portnummer). Ist der PC also so konfiguriert, dass IP-Pakete an bestimmte Portnummern ignoriert werden, so kann ein Angriff auf diese Portnummer nichts ausrichten. Eine Personal Firewall kann den Benutzer an dieser Stelle also darin unterstützen, die offenen und geschlossenen Ports zu verwalten.

In Windows sind häufig die lokalen Festplatten für den Zugriff von außen frei geschaltet. Ein normaler Benutzer weiß nun nicht unbedingt, wie und in welchem Windows-Dialog er das abstellen kann. In diesem Fall kann eine Personal Firewall die Einstellung für den Benutzer vornehmen.

Zielt ein Angriff von außen allerdings auf einen Port, auf dem der Privat-PC normalerweise Pakete entgegennimmt, so kann eine Personal Firewall hier selten helfen. Solche Angriffe nutzen Schwachstellen in der Software aus, die die Pakete an diesem Port entgegennimmt und verarbeitet. Der Microsoft-Internet-Information-Server (ein Webserver) enthielt einige Schwachstellen, die von *Buffer-Overflow*-Angriffen ausgenutzt wurden. Solche spezielleren Angriffe werden in Kurs *(01867) Sicherheit im Internet 2* detaillierter vorgestellt. Da der Rechner als Webserver eingesetzt wurde, wurden Pakete an Port Nummer 80 von der Firewall durchgelassen. Diese Pakete führten dann zum Absturz des Internet-Information-Servers.

[9] Das gilt natürlich nur so lange, wie der Router keine Schwachstellen enthält. Sonst könnte ein Angreifer den Router und somit die darin enthaltene Firewall umkonfigurieren und beispielsweise die Firewall abschalten.

Unerlaubter Datentransfer nach außen: Um unerlaubten Datentransfer zu verhindern, muss die Personal Firewall überwachen, welche lokalen Programme eine Verbindung nach außen aufbauen. Beim versuchten Verbindungsaufbau gibt die Firewall eine Meldung auf dem Bildschirm aus und der Benutzer kann entscheiden, ob die Verbindung aufgebaut werden darf oder nicht. Diese Entscheidung muss der Benutzer anhand des Namens des Programms treffen. Woher soll der durchschnittliche Benutzer aber wissen, welchen Programmen er den Zugriff erlauben darf? Mit dem Webbrowser möchte man Internetseiten anschauen, dieses Programm soll bestimmt zugreifen dürfen. Aber was ist mit anderen Programmen? Der Virenscanner soll seine Vorlagendatei immer aktuell halten. Er wird also bestimmt auch Verbindungen ins Internet aufbauen dürfen. Wie wird sich ein durchschnittlicher Benutzer wahrscheinlich entscheiden, wenn ein Trojanisches Pferd sich den Namen *Internet Explorer 11.5.1* gibt und dann die Nachricht „Darf das Programm *Internet Explorer 11.5.1* eine Verbindung ins Internet aufbauen?“ auf dem Bildschirm steht?

Eine Schadsoftware muss die Daten aber möglicherweise gar nicht selbst versenden. Statt dessen kann die Schadsoftware andere Programme aufrufen und diese dann die Daten versenden lassen. Das setzt natürlich voraus, dass sich Programme von anderen Programmen starten und entsprechend steuern lassen.

Eine Personal Firewall muss sich einige Konfigurationsinformationen dauerhaft merken. Beispielsweise ist die Liste aller Programme, die Verbindungen ins Internet aufbauen dürfen eine solche Information. Sie stehen in einer Konfigurationsdatei. Hat eine Schadsoftware nun Zugriff auf diese Konfigurationsdatei, so kann sie sich selbst in die Liste eintragen und somit die Erlaubnis geben, Verbindungen ohne Rückfragen an den Benutzer aufzubauen.

Ist in der Personal Firewall nun konfiguriert, dass der E-Mail-Client ohne Rückfrage eine Verbindung ins Internet aufbauen darf, dann kann diese Personal Firewall keine E-Mail-Würmer mehr kontrollieren. Das sind Programme, die sich Schwachstellen von E-Mail-Clients zu Nutze machen und sich massenhaft an Adressen aus dem Adressbuch des Benutzers versenden. Die Personal Firewall kann diese E-Mails nicht von normalen E-Mails unterscheiden. Und der E-Mail-Client darf ja Verbindungen ins Internet aufbauen.

Ein andere Möglichkeit Daten vom Privat-PC ins Internet zu übertragen besteht in der „ungewöhnlichen“ Nutzung von erlaubten Protokollen. Ein Plug-In in einem Webbrowser wird bestimmt auch einen HTTP-Request ins Netz senden dürfen. In diesen Request kann das Plug-In vertrauliche Informationen einbetten. Eine URL könnte beispielsweise die folgende Form haben: `http://xxx.com/index.html?Passwort=asdfg`

Alle Zeichen hinter dem Fragezeichen können vertrauliche Informationen sein, die der empfangende Webserver nicht auswertet, sondern nur speichert. Die Antwort des Webservers ist also völlig unabhängig vom hinteren Teil der URL. Der Benutzer bekommt hiervon nur schwerlich etwas mit. Und auch eine Personal Firewall kann nicht entscheiden, ob dieser HTTP-Request nun erlaubt sein soll oder nicht.

Zusammenfassung Personal Firewalls: Alle modernen Betriebssysteme werden bereits mit einer Firewall ausgeliefert. Benutzen Sie diese Firewall und verbieten Sie den Verbindungsaufbau von außen zu Ihrem PC zunächst einmal. Tragen Sie bei Bedarf Ausnahmen ein, z. B. für den Zugriff auf ihren PC mit

SSH. Den unerlaubten Datentransfer nach außen können diese Firewalls nur schwer bzw. gar nicht verhindern. Eine zusätzliche Personal Firewall ist dann meist entbehrlich.

3.5.3 Sichere Windows-Konfiguration

Das Betriebssystem Windows erlaubt es, Rechte für einzelne Benutzer zu vergeben. Somit ist es möglich, dass ein Benutzer keine Möglichkeit mehr hat, die Daten anderer Benutzer zu sehen. Außerdem kann man so verhindern, dass ein Benutzer (absichtlich oder unabsichtlich) weitere Programme auf dem System installiert oder Systemeinstellungen verändert. Wichtig für die Umsetzung dieser Mechanismen sind die folgenden Punkte:

1. Das System muss verschiedene Benutzer kennen und auseinander halten können. Im Allgemeinen spricht man von **Subjekten** (siehe auch Kurs *(01802) Betriebssysteme* oder Kurs *(01868) Sicherheit im Internet 1 – Ergänzungen*) und nicht von Benutzern. Subjekte wollen Aktionen ausführen. Deshalb sind Programme in diesem Sinne auch Subjekte.

   Subjekten

2. Das System muss **Rechte** verwalten und überprüfen können. Rechte beziehen sich häufig auf **Objekte**, also Ressourcen im System, auf denen Aktionen ausgeführt werden können. Dateien, die gelesen werden sollen, Ordner, die durchsucht werden sollen, Programmdateien, die ausgeführt werden sollen oder Betriebssystemfunktionen, die ausgeführt werden sollen, sind Beispiele für Objekte[10].

   Rechte
   Objekte

3. Das System muss bestimmte Aktionen kennen und ermöglichen. Hierbei handelt es sich um die typischen Aktionen, die Subjekte auf Objekten normalerweise ausführen. Beispiele sind: Lesen, Schreiben und Ausführen.

Damit nicht für jeden Benutzer und für alle Objekte des Systems die Rechte explizit definiert werden müssen, gibt es das Konzept der **Benutzergruppen**. Einzelne Benutzer werden Mitglied einer Gruppe und erhalten dadurch die Rechte, die für die Gruppenmitglieder definiert wurden. Windows unterscheidet zwischen Administratoren und normalen Benutzern. Mitglieder der Gruppe *Administratoren* haben sehr weitgehende Rechte. Sie können das System verwalten, d. h. neue Benutzer anlegen, Rechte administrieren und Programme installieren. „Normale“ Benutzer können das nicht bzw. nur sehr eingeschränkt für sich selbst.

Benutzergruppen

Neben den genannten Gruppen existieren weitere Gruppen, beispielsweise für Support, Remote-Verwaltung des Systems usw. Ein Administrator kann weitere Gruppen anlegen und diesen Gruppen spezielle Rechte zuordnen.

Die eigentlichen Rechte werden in Windows in Form von **Access-Control-Lists (ACL)** verwaltet. Eine ACL wird einem Objekt zugeordnet, z. B. einer Datei oder einem Ordner. In der ACL stehen mehrere Einträge, genannt **Access-Control-Entries (ACE)**. Sie spezifizieren für einen Benutzer (oder eine Benutzergruppe) die Zugriffsrechte auf das Objekt. In einem ACE können die Aktionen entweder erlaubt oder explizit verboten werden. Dabei haben Verbote Vorrang. Darf beispielsweise eine Benutzergruppe X das Objekt lesen,

Access-Control-Lists (ACL)
Access-Control-Entries (ACE)

[10] Programme sind letztlich also Subjekte und Objekte zugleich. Sie führen Aktionen aus und man kann Aktionen auf ihnen ausführen.

dem Benutzer Y ist das aber explizit verboten, so darf Y die Datei auch dann nicht lesen, wenn er Mitglied der Gruppe X ist.

Zur Vereinfachung der Verwaltung kann man Rechte auch vererben. Damit kann dann ein kompletter Verzeichnisbaum (ein Ordner inklusive aller Dateien und Unterordner) mit denselben Zugriffsrechten versehen werden. Der Administrator muss die Rechte nur dem Ordner zuweisen und sie als vererblich kennzeichnen. Dann werden den Dateien und Unterordnern automatisch diese Rechte zugewiesen.

Benutzer anlegen: Um die Sicherheitsfunktionen von Windows zu benutzen, muss man also Benutzer anlegen. Bereits während der Installation von Windows kann man Benutzer anlegen. Diese Benutzer werden jedoch automatisch der Gruppe der Administratoren zugeordnet. Man sollte an dieser Stelle also noch nicht die normalen Benutzer anlegen. Nach der Installation des Betriebssystems gibt es verschiedene Möglichkeiten, weitere Benutzer anzulegen. Mit Hilfe der Kommandozeile und dem Kommando `net user` kann man neue Benutzerkonten anlegen oder existierende Benutzerkonten bearbeiten. Das Kommando `net help user` liefert eine Erklärung der Optionen des Kommandos. Daneben gibt es auch ein grafisches Werkzeug. Öffnet man das Fenster „Systemsteuerung“, so erscheint das in in Abbildung 3.18 gezeigte Fenster. Dort kann man den Punkt

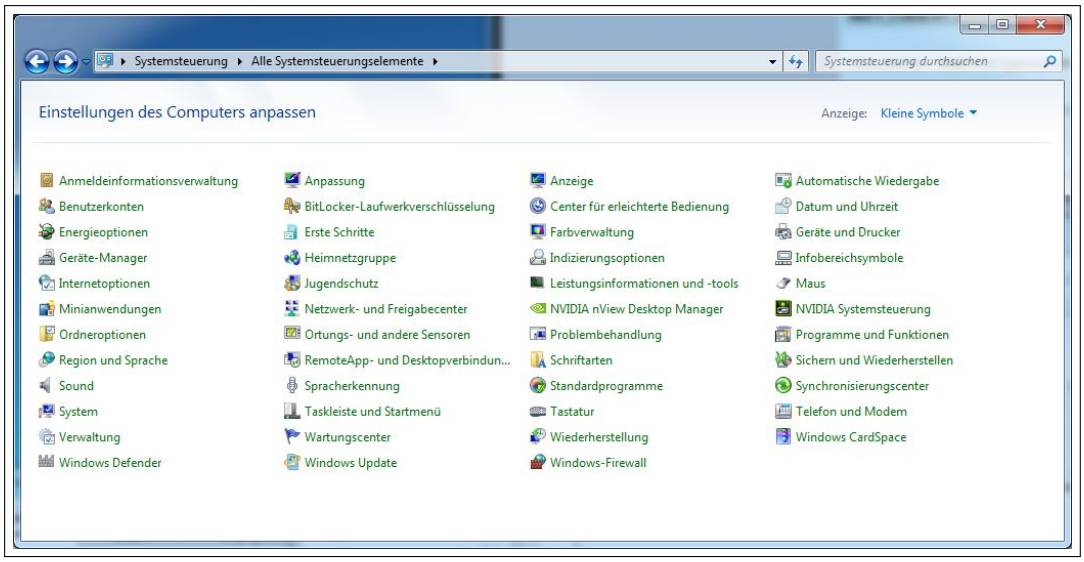


Abbildung 3.18: Systemsteuerung in Windows

„Benutzerkonten“ auswählen und dann Benutzerkonten anlegen, bearbeiten oder auch löschen.

Damit sich auch tatsächlich nur die vorgesehenen Benutzer unter einem Benutzerkonto anmelden, sollte man Benutzerkonten grundsätzlich mit Passwörtern versehen. Insbesondere das Administratorkonto und alle Benutzerkonten der Gruppe *Administratoren* müssen mit einem guten Passwort geschützt werden.

ACL definieren: Nachdem man Benutzerkonten und Benutzergruppen eingerichtet hat, sollte man die Zugriffsrechte für die Ordner mit den persönlichen Daten der Benutzer anpassen. Hierzu gibt es wiederum mehrere Möglichkeiten. Mit Hilfe des Kommandos `cacls` kann man Zugriffsrechte für Dateien oder

Ordner setzen. Alternativ dazu kann man auch mit der graphischen Umgebung *Windows Explorer* arbeiten. Klickt man mit der rechten Maustaste auf eine Datei oder einen Ordner und wählt im Kontextmenü dann den Eintrag Eigenschaften, so öffnet sich ein Fenster wie in Abbildung 3.19. Während bei

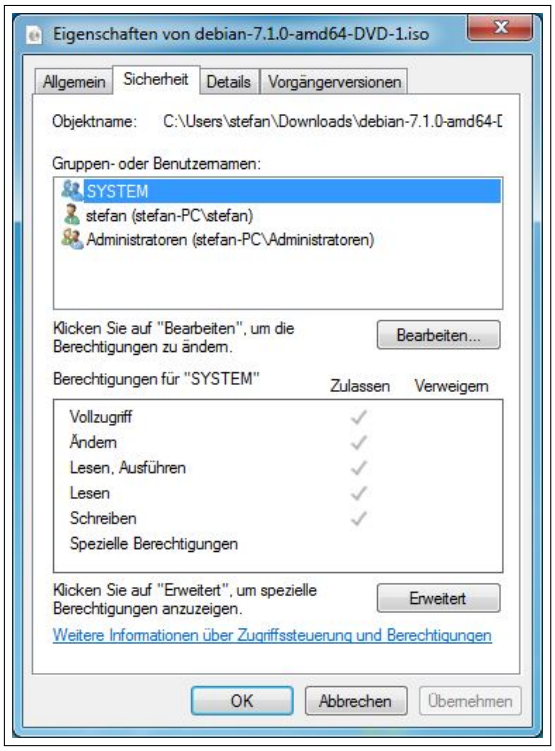


Abbildung 3.19: Einstellen der ACL in Windows

klassischen UNIX-Systemen nur drei Berechtigungen (Lesen, Schreiben, Ausführen) vorkommen, sind es bei Windows mehr. In der graphischen Oberfläche wie in Abbildung 3.19 muss man die Berechtigungen nicht einzeln anklicken. In der Oberfläche gibt es auch „Sammelbegriffe“ wie Vollzugriff. Setzt man dort den Haken bei Zulassen, dann werden „alle“ expliziten Berechtigungen gesetzt.

Damit das Betriebssystem nun bei Zugriffen auf Dateien oder Ordnern die Rechte auch tatsächlich prüfen kann, müssen die ACLs im Dateisystem gespeichert sein. Dies ist nur für das Dateisystem **NTFS** der Fall. Aktuell kann man Windows nur in eine NTFS-Partition installieren. Auf USB-Sticks oder weiteren angeschlossenen Festplatten könnte aber noch das ältere FAT-, bzw. FAT32-Dateisystem benutzt werden. Dort kann man keine Zugriffsrechte setzen.

NTFS

Leider ist der *Windows Explorer* so vorkonfiguriert, dass daraus Sicherheitsprobleme entstehen. Der *Windows Explorer* zeigt bei bekannten Dateinamenerweiterungen, wie beispielsweise `.doc`, `.txt` usw. diese Endungen nicht mit an. Anhand dieser Endungen wird jedoch das Programm definiert, das Dateien mit dieser Endung öffnen soll. Durch einen Doppelklick auf die Datei wird automatisch das zugehörige Programm gestartet und lädt die Datei. Eine Datei mit dem vollständigen Namen `harmlos.txt.exe` wird vom Explorer dann ohne die Endung `.exe` angezeigt. Der Benutzer sieht den Namen `harmlos.txt` und klickt darauf. Es wird aber *nicht* der Editor gestartet, sondern ein Programm. Dieses Programm kann nun im Namen des Benutzers Unheil anrichten. Deshalb sollte man die Ordneroptionen im *Windows Explorer* durchgehen und passend setzen (siehe Abbildung 3.20).

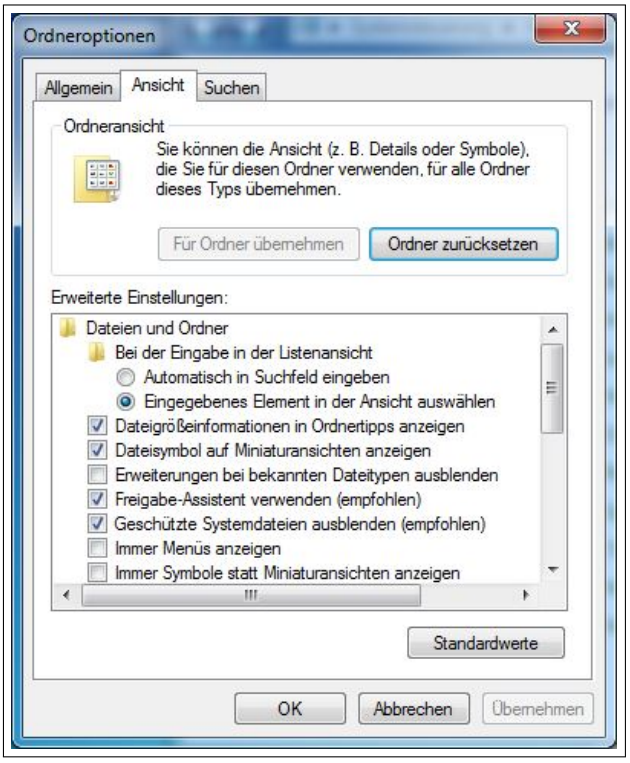


Abbildung 3.20: Explorer-Einstellungen für Windows-Ordner

Konkret sollte man auf die folgenden Punkte achten:

- Namen immer komplett anzeigen, d. h. keinen Haken bei „Erweiterungen bei bekannten Dateitypen ausblenden“ setzen.
- Detail-Liste anzeigen an Stelle von Bildchen.
- Keinen Autostart von eingelegten Medien (CD, USB-Stick oder Diskette) erlauben.

Das automatische Abspielen von eingelegten Datenträgern, beispielsweise CDs, ist recht bequem, wenn Musik-CDs eingelegt werden. Andererseits ist es auch gefährlich, denn auch bei anderen CDs wird dann nach automatisch startbaren Dateien auf der CD gesucht. Wird eine solche Datei gefunden, dann wird sie automatisch gestartet. Bei CDs, die vielen Computerzeitschriften beiliegen, sind das häufig spezielle Browserprogramme. Als Benutzer sollte man solche Programme aber immer nur manuell starten und das auch nur dann, wenn man tatsächlich sicher ist, dass man das Programm starten möchte. Unter Windows kann man in der Systemsteuerung „Automatische Wiedergabe“ auswählen. In dem Fenster aus Abbildung 3.21 können Sie dann einstellen, ob CDs oder DVDs automatisch abgespielt werden sollen oder nicht.

Zustand überwachen: *Microsoft* bietet ein kostenloses Programm mit dem Namen *Microsoft Baseline Security Analyser (MBSA)* an. Dieses Programm untersucht Computer auf verschiedene Schwachstellen, wie:

- Fehlende Sicherheits-Updates für das Betriebssystem oder für installierte Microsoft-Programme.
- Zu schwache oder gar fehlende Passwörter einzelner Benutzerkennungen.

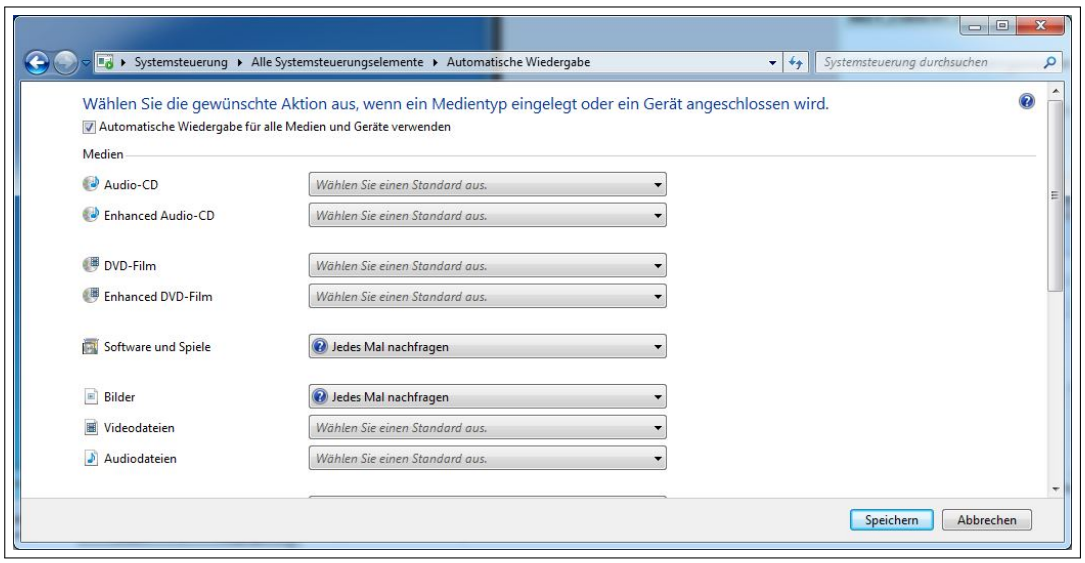


Abbildung 3.21: Ausschalten des automatischen Abspielens eingelegter Datenträger

- Fehlende oder deaktivierte Firewall.
- Fehlende oder deaktivierte Virenscanner.
- Anmeldemöglichkeiten über das Netz (evtl. ohne Passwort).
- Freigaben von Laufwerken.
- Ausgewählte Dateisysteme.

Das Programm ist frei verfügbar und kann aus dem Internet geladen werden. Es liefert eine gute Zusammenfassung des Systemzustands, und anhand der gemeldeten Probleme weiß der Anwender, welche Verbesserungen der Systemsicherheit erforderlich und machbar sind.

Benutzerkontensteuerung

User Account Control (UAC): Mit Windows Vista hat *Microsoft* das Konzept der Benutzerkontensteuerung (engl. **User Account Control**) eingeführt. Es ist auch in den Nachfolgerversionen vorhanden. Hintergrund hierfür war die Tatsache, dass sehr viele Benutzer von Windows XP unter einer Benutzerkennung mit Administratorrechten gearbeitet haben. Somit konnte bösartige Software, sobald einmal gestartet, die komplette Kontrolle über den PC übernehmen. Denn jedes Programm, das der Benutzer (auch unbemerkt und unabsichtlich) gestartet hat, hat die Administratorrechte bekommen.

Die Grundidee von UAC ist es, dass der Benutzer jedesmal explizit gefragt wird, wenn eines der Programme, das er gestartet hat, eine sicherheitskritische Funktion ausführen will. Dabei werden alle Funktionen, die Administratorrechte brauchen, als sicherheitskritisch angesehen. Auch wenn der Benutzer mit einer Kennung aus der Administratorengruppe angemeldet ist, wird die kritische Funktion nicht einfach ausgeführt, sondern es erscheint ein Dialogfenster, in dem der Benutzer die Ausführung explizit bestätigen muss.

Startet ein normaler Benutzer so ein Programm, so muss er in dem Bestätigungsfenster zusätzlich ein Administratorpasswort eingeben. Abbildung 3.22 zeigt, wie diese beiden Fenster aussehen können.

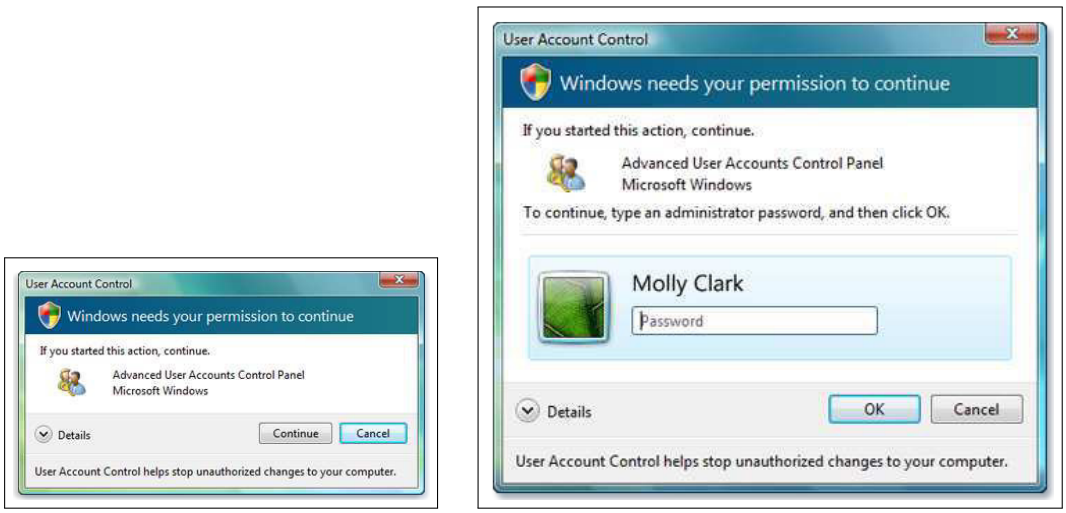


Abbildung 3.22: Kontrollfenster eines User-Accounts in Windows-Vista

Dieser Mechanismus sorgt dafür, dass schädliche Software nun keine unbemerkten Änderungen am System mehr vornehmen kann. Der Benutzer muss dazu im Bestätigungsfenster erst *Continue* oder *OK* anklicken. Wichtig ist dabei, dass kein Programm diese Klicks an Stelle des Benutzers ausführen kann. Außerdem muss der Benutzer wissen, ob die Rückfrage durch eine absichtliche Aktion seinerseits (beispielsweise weil er ein Programm installieren möchte) ausgelöst wurde oder ob ein im Hintergrund laufendes Programm die Ursache ist.

Nachteil

Der Nachteil dieses Mechanismus ist, dass Benutzer möglicherweise sehr oft gefragt werden, ob eine sicherheitskritische Aktion zugelassen werden soll oder nicht. Je öfter jemand dazu befragt wird, umso eher neigt derjenige dazu, immer auf *OK* zu klicken. Schließlich will man mit seiner Arbeit am PC vorankommen. Die Rückfragen bremsen an dieser Stelle.

Außerdem muss sich ein Benutzer nun auch über die Konsequenzen dieses Mechanismus im klaren sein. Jeder Benutzer muss wissen, ob er gerade etwas Sicherheitskritisches tut (dann muß er *OK* oder *Continue* anklicken) oder ob ein Programm im Hintergrund versucht, etwas Böses zu tun (dann muss er *Abbrechen* anklicken). Diese Unterscheidung kann man nur mit einem gewissen Maß an Wissen richtig treffen.

Letztlich muss man natürlich auch sicherstellen, dass schädliche Software nicht den Mechanismus selbst einfach abschalten kann. *Microsoft* hat es vorgesehen, dass Benutzer UAC auch wieder ausschalten können. Abbildung 3.23 zeigt, wie man über die Benutzerkontensteuerung die Benachrichtigungen im Rahmen von UAC einstellen kann. Stellt man „nie benachrichtigen“ ein, dann bekommen Programme ohne Rückfrage die Administrations-Rechte. Somit kann prinzipiell auch ein Programm diese Einstellung vornehmen und faktisch UAC ausschalten, um danach weitere kritische Änderungen am System vorzunehmen.

3.5.4 Zusammenfassung: Schutz des privaten PCs

Um einen privaten PC, der mit dem Internet verbunden ist, wirkungsvoll zu schützen, sind verschiedene Maßnahmen erforderlich. Grundsätzlich gilt, dass der Benutzer sich der Problematik bewusst sein muss und außerdem das erforderliche Wissen zur Sicherung des privaten PCs besitzen muss.

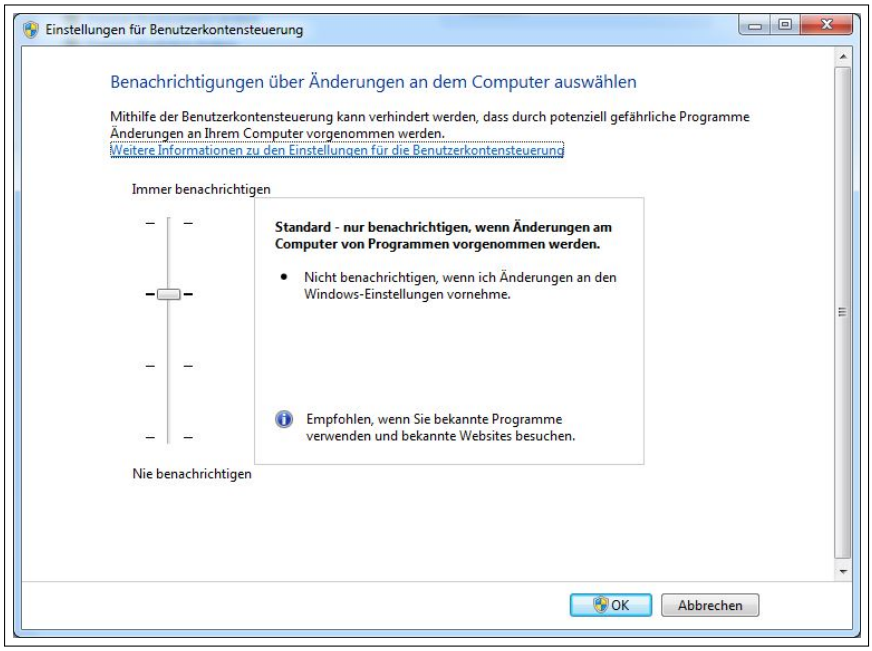


Abbildung 3.23: Einstellungen für UAC in Windows

Antivirensoftware

Mit Hilfe von Antivirensoftware kann man Viren erkennen und auch wieder vom Rechner entfernen. Wichtig bei der Antivirensoftware ist, dass sie (1) tatsächlich alle bekannten Viren erkennen kann, (2) verschiedene Betriebsmodi (Hintergrund und Batch) kennt und (3) über das Netz mit neuen Viren-Templates aktualisiert werden kann.

Firewall

Die mit dem Betriebssystem mitgelieferte Firewall sollte aktiviert und korrekt konfiguriert werden. Zusätzlich kann Personal-Firewall-Software den Schutz eines Rechners ergänzen. Wichtig ist hierbei, dass diese Software einen Rechner nicht automatisch sicher macht. Vielmehr muss der Benutzer genau über die Gefahren informiert sein und die prinzipiellen Grenzen von Personal-Firewall-Software kennen. Nur dann ist eine sinnvolle Konfiguration der Software möglich. Aber auch in diesem Fall lässt sich ein Rechner mit Personal Firewall immer noch angreifen. Prinzipiell können Firewalls den Datentransfer von außen nach innen sehr gut kontrollieren, während die Kontrolle des Datentransfers von innen nach außen viel schwieriger (wenn nicht unmöglich) ist.

sichere Konfiguration

Auch die sichere Konfiguration des Betriebssystems selbst ist wichtig. Richten Sie für die verschiedenen Arbeiten am Rechner verschiedene Benutzer ein, die unterschiedlichen Gruppen zugeordnet sind. Führen Sie keine alltäglichen Arbeiten als Administrator aus und deaktivieren Sie das automatische Starten von Programmen, falls eine CD/DVD eingelegt wird oder ein USB-Stick angeschlossen wird.

Sicherheits-Updates

Schließlich ist es wichtig, Sicherheits-Updates des Betriebssystems und der Systemsoftware regelmäßig zu installieren. Man kann das entweder manuell machen oder das System so einstellen, dass automatisch nach Sicherheits-Updates gesucht wird. Eine regelmäßige Überprüfung des Systemzustandes hilft bei der Entdeckung und Behebung von Schwachstellen.

3.6 Zusammenfassung

Nach dem Durcharbeiten dieser Kurseinheit sollten Sie die folgenden Fragen beantworten können:

- Welche Möglichkeiten und Systeme zur sicheren E-Mail-Kommunikation im Internet existieren?
- Wie wird Vertraulichkeit und Authentizität beim Surfen im WWW möglich?
- Wie erkenne ich, ob ich eine sichere Verbindung zu einem Webserver habe?
- Welche Probleme können Cookies bereiten und wie kann ich diesen Problemen begegnen?
- Wie kann ich sichere Verbindungen zu einem entfernten Computer aufbauen, z. B. um dort Administrationsaufgaben zu erledigen?
- Wie kann ich meinen privaten PC sichern, so dass der PC möglichst wenig gefährdet ist?

Lösungen der Übungsaufgaben

Übungsaufgabe 3.1 Sie trauen dem Schlüssel von X, denn er ist von A signiert. Da Sie glauben, dass A das nur dann tut, wenn er wirklich sicher ist, so glauben Sie, dass der Schlüssel von X korrekt ist. Die zusätzliche Unterschrift von C trägt nicht zur Steigerung Ihres Vertrauens bei, aber auch nicht zur Verminderung.

Dem Schlüssel von Y trauen Sie nicht, denn er ist nur von einer Person unterschrieben, der Sie weniger Vertrauen entgegen bringen. Für das Vertrauen in den Schlüssel von Y wären aber zwei Personen mit der Vertrauensstufe *usually trusted to sign* erforderlich.

Übungsaufgabe 3.2 Zu den MIME-Content-Types gehört auch der Typ *Application*, der ein ausführbares Programm bezeichnet. Startet der E-Mail-Client ein als Teil einer E-Mail empfangenes Programm automatisch (vergleichbar zum automatischen Anzeigen der empfangenen Grafiken), können beliebige Schäden angerichtet werden. Das Programm könnte beispielsweise die Festplatte löschen, einen Virus installieren (siehe Abschnitt 1.4.1) o. ä.

Übungsaufgabe 3.3 Wenn der öffentliche Schlüssel dem Browser bekannt sein muss, so ergeben sich zumindest zwei Probleme:

1. Der Schlüssel muss irgendwo gespeichert sein. Das kann im Programmcode selbst sein (der Schlüssel wurde „eincompiliert“) oder in einer eigenen Datei. In beiden Fällen könnte ein Angreifer eine modifizierte Version des Browsers verteilen, in der diese öffentlichen Schlüssel gefälscht sind. Konkret bedeutet das, dass sie durch Schlüssel ersetzt wurden, zu denen der Angreifer den privaten Schlüssel besitzt. Der Angreifer kann nun Zertifikate fälschen.

2. Jeder öffentliche Schlüssel, d. h. jedes Zertifikat, ist nur eine begrenzte Zeit gültig. Anschließend muss es durch ein Neues ersetzt werden. Daher müssen die dem Browser bekannten Zertifikate auch einmal ausgetauscht werden. Das kann dazu führen, dass der Benutzer einen neuen Browser installieren muss.

Übungsaufgabe 3.4 Ein Risiko besteht darin, dass man sich nicht sicher sein kann, ob das Cookie tatsächlich von dem Benutzer stammt, von dem der Empfänger glaubt, dass es kommt. Cookies werden ja im Klartext über das Netz geschickt. Sie können also abgehört werden, und ein Angreifer könnte sich das Cookie eines Fremden besorgen, auf seinen Rechner kopieren und dann die betreffende Webseite aufrufen.

Ein weiteres Risiko ist, dass Cookies zunächst einmal den Rechner identifizieren und nicht zwangsläufig den Benutzer. Auf einem PC mit vielen Benutzern könnte beispielsweise der zweite Benutzer für seinen Vorgänger gehalten werden, wenn der erste Benutzer das Cookie geladen hat und der zweite Benutzer später noch einmal die Webseite besucht, von der das Cookie stammt.

Übungsaufgabe 3.5 In dem beschriebenen Fall wäre ein *Man in the middle (MITM) Angriff* möglich. Ein Angreifer A kann den Verbindungsaufbau eines Clients C zum Server S abfangen und seinerseits die Verbindung mit C aufbauen. Nun will der Client seine Gegenstelle authentisieren (er denkt ja, dass er mit S

kommuniziert). Der Client erwartet also die digitale Signatur der SSH-Session-ID x, die er mit der Gegenstelle (also A) zusammen erzeugt hat.

Um das zu berechnen kann A nun seinerseits eine SSH-Verbindung zum Server S aufbauen. Wenn A hierbei erreichen kann, dass auch diese SSH-Session-ID den Wert x erhält, so authentisiert sich der Server S gegenüber A mit dem digital signierten x. Der Angreifer A muss das nun nur noch an den Cient weiterleiten und der Client glaubt, dass er mit dem echten Server S verbunden wäre, obwohl er mit dem Angreifer A verbunden ist.