Webbasierte Authentifikation mittels Zertifikaten

Studienarbeit, Mirko Mrowczynski

9. Dezember 1999

Abstraktum

Gegenstand dieser Arbeit ist es Sicherheitsaspekte in Netzwerken zu erläutern, eine Möglichkeit einer sicheren Verbindung im Internet vorzustellen und deren Realisierung unter Verwendung heute vorhandener Recourcen an Hard- und Software zu beschreiben.

Erstellt durch:	Mirko Mrowczynski
Betreuer:	Dipl. Inf. Sven Köhler
Betreuender Hochschullehrer:	Prof. Dr.-Ing. habil. Uwe Hübner
Eingereicht:	am 10. Dezember 1999 an der Technischen Universität Chemnitz Fakultät für Informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Aufgabenstellung

Ziel meiner Studienarbeit ist es eine webbasierte Authentifikation mittels Zertifikaten zu ermöglichen.

Diese Formulierung ist sehr knapp gehalten und aus eben diesem Grund eher unpräzise. Das sorgte dafür, daß ich beim Lösen der Aufgabe Aspekte beleuchten mußte, die für mögliche Realisierungen keine Bedeutung haben aber zum Verstehen des Sachverhaltes dienlich sind. Andererseits bekam ich durch diese Aufgabe einen Spielraum, der mir die Suche nach einer praktikablen Lösung erleichterte.

Im nächsten Kapitel möchte ich die in der Aufgabenstellung verwendeten Begriffe klären, um anschließend einen Einblick über das Zustandekommen und die Bedeutung der Aufgabe liefern zu können. Die darauffolgenden Kapitel befassen sich mit Grundlagen zu Bedrohungen, denen Informationstechnologie ausgesetzt ist und Präventivmaßnahmen, die ergriffen werden können, um diese Bedrohungen abzuschwächen. Im Anschluß daran wird an ausgewählten Beispielen der aktuelle Entwicklungsstand beschrieben.

Ab Kapitel acht wird die Umsetzung der Aufgabe auf einer von mir gewählten Plattform erläutert. Diese Umsetzung erfolgt unter Berücksichtigung der in den vorangegangenen Kapiteln beschriebenen theoretischen Erkenntnisse, so daß an dieser Stelle mehr Wert auf die Nachvollziehbarkeit der gezeigten Möglichkeiten gelegt wird.

Zum Schluß gebe ich noch einen kleinen Ausblick, welche weiteren Entwicklungsschritte im Umfeld dieser Aufgabe möglich sind und wie dies durch neue Anwendungen beeinflußt wird.

Kapitel 2

Begriffsklärung

2.1 Authentifikation

Eine Authentifikation ist der Nachweis eines potentiellen Kommunikationspartners, daß er der ist für den er sich ausgibt. Seine Identität wird durch die Überprüfung unverwechselbarer Merkmale festgestellt. Je nach Art dieses Merkmals handelt es sich um einen bestimmten Typ der Authentifikation. Man unterteilt hier in drei Gruppen.

Eine sehr beliebte Möglichkeit eine Maschine die Authentizität eines Nutzers prüfen zu lassen ist die Authentifikation durch Wissen. Das älteste mir bekannte Beispiel hierzu ist "Sesam öffne dich !" aus Alibaba und die vierzig Räuber. Eine etwas zeitgemäßere Variante der Authentifikation durch Wissen ist das Einloggen an einem Rechner mittels Nutzername und Passwort, wobei der Nutzername öffentlicher Parameter ist, also selbst nicht zum Geheimnis gehört und nur festlegt welcher "Berg" geöffnet werden soll.

Die zweite Art der Authentifikation ist das Anzeigen von Besitz. Der einfachen Handhabbarkeit wegen ist der Gegenstand mit dessen Hilfe die Authentifikation durch Besitz geschieht meist in seinen Abmessungen genormt. Das eigentliche Merkmal mit dem Identifiziert wird, ist zur optischen Erkennung auf die Oberfläche aufgebracht. Heute wird diese Methode, mit rein optischer Erkennung, gern da verwendet, wo nur ein geringes Sicherheitsbedürfnis besteht aber sich durch den Einsatz der Service erhöht oder statistische Auswertungen ermöglicht werden, die diesen Aufwand rechtfertigen. Das Medium ist dann meist eine bedruckte Plastikkarte, die über einen Barcodescanner erkannt werden kann. Früher waren Siegelringe eine Umsetzung dieser Authentifikationsform. Um die Sicherheit bei der maschinellen Erkennung zu erhöhen, wird die Merkmalspalette in vielen Fällen, wie bei der EC-Karte, durch Magnetstreifen und Speicherchips erweitert.

Eine dritte Art der Authentifikation ist durch Eigenschaften möglich. Die Eigenschaft wird dabei so gewählt, daß sie nicht übertragbar ist. Um die Authentizität eines Menschen zu prüfen müssen also biometrische Eigenschaften wie Augenhintergrund, Fingerabdrücke oder Stimme kontrolliert werden. Die Dresdner Bank testet seit wenigen Tagen Geldautomaten, die den Kunden über Videotechnik an seiner Iris erkennen [1]. Das besondere daran ist, daß die biometrische Eigenschaft hier nicht als Ergänzung zu anderen Authentifikationsmöglichkeiten verwendet wird.

Der Kunde braucht also weder Karte noch PIN. Ich persönlich bin sehr gespannt, wann damit die "Und wie möchten sie bezahlen ?" VISA Werbung getoppt wird.

Ein Problem, das bei der Authentifikation durch biometrische Eigenschaften auftritt ist, daß zum Erkennen dieser Eigenschaften spezielle Technik erforderlich ist und diese Technik lokal verfügbar sein muß. Nach dem Scannen der Eigenschaft wird diese in Form von Bits und Bytes vom Scanner zum auswertenden Rechner übertragen. Bei dieser Übertragung ist die mit der Eigenschaft assoziierte Information aber am ehesten mit Wissen vergleichbar und kann entsprechend dupliziert oder manipuliert werden. Diese Übertragung ist also entsprechend zu Schützen. Das kann durch den Einsatz von Zertifikaten erfolgen, was nun in einem etwas anderen Zusammenhang das Ziel meiner Studienarbeit ist. Die Authentifikation mittels biometrischer Eigenschaften sei daher hier nur der Vollständigkeit wegen genannt. Sie wird im weiteren Verlauf meiner Arbeit keine Rolle spielen.

Bei Wissen und Besitz stellt die Weitergabe kaum ein Problem dar, deshalb werden unterschiedliche Arten der Authentifikation oft miteinander kombiniert. Das ist zum Beispiel dann der Fall, wenn man mit dem eigenen Schlüssel die Haustür aufschließt (Besitz), an der Wohnungstür einen bestimmten Rhythmus klingelt (Wissen), um dann auf die Frage "Wer da ?" mit "ich bin's" zu antworten (Stimme als Eigenschaft).

Je nach vereinbarten Sicherheitsanforderungen kann es notwendig werden, daß sich alle Partner gegenseitig authentifizieren. Das schließt auch Maschinen ein. Selbst wenn nicht explizit vereinbart wurde, daß sich ein Computer identifizieren muß, kann ich sein "Wissen" prüfen. Ich kann versuchen mich mehrmals mit falschem Passwort anzumelden und verwende dann das Richtige. Wenn ich daraufhin Zugriff erhalte und alle vorherigen Fehleingaben als solche erkannt wurden scheint der Computer der zu sein, für den er sich ausgibt. Wenn der Nutzer sich einer strengen Authentifikation unterziehen muß sollte er von der Maschine jedoch das gleiche verlangen können ohne Gefahr zu laufen sein Login zu sperren. Prinzipiell kann das durch alle oben genannten Möglichkeiten geschehen.

2.2 webbasiert

In der Aufgabenstellung steht, daß diese Authentication webbasiert sein soll. Web ist nicht gleich World Wide Web, es kann sich also auch um ein lokales, isoliertes Netzwerk handeln. Die Prinzipien unterscheiden sich jedoch nicht. Die Realisierungen haben nur unterschiedliche Ausmaße. Eine webbasierte Lösung ist immer Client - Server basiert. Vom Client, dem Webbrowser eines Nutzers, wird eine Verbindung zu einem Server, in diesem Fall einem Webserver, aufgebaut um von dort Informationen zu beziehen. Andere Server mit denen ein Browser in Verbindung treten kann sind beispielsweise FTP- und Mail-Server. Mein Bericht bezieht sich jedoch ausschließlich auf Webserver, da ihre Funktionalität die eines FTP- Servers beinhaltet und Zertifikate zur Authentication bereits innerhalb von Mail's eingesetzt werden können.

2.3 Zertifikate

Der dritte zu beachtende Punkt der Aufgabenstellung ist, daß die webbasierte Authentifikation durch Zertifikate erfolgen soll. Ein solches Zertifikat ist ein Gutachten einer vertrauenswürdigen und aus diesem Grund normalerweise unparteiischen Instanz. Solche Instanzen heißen Certifikation Authorities. Im weiteren Bericht werde ich Certification Authorities mit CA abkürzen. In einem Zertifikat werden bestimmte Daten des Besitzers bestätigt. Diese Daten müssen natürlich vor der Ausstellung eines Zertifikates von der ausstellenden CA geprüft werden. Wenn sich nun ein Nutzer mit Hilfe eines Zertifikates bei einem Webserver authentifizieren möchte, muß der Server die Daten des Nutzers nicht selbst prüfen, sondern kann sich auf die Überprüfung der Daten durch die ausstellende CA verlassen. Der Server verwaltet deshalb eine Liste mit vertrauenswürdigen CA's. Es erfolgt also keine Prüfung von "Einzelschicksalen", sondern die Prüfung einer Gruppenzugehörigkeit. Wenn ein Nutzer dem Server ein für diesen gültiges Zertifikat vorweisen kann steht einer Kommunikation nichts mehr im Wege. Um die Sicherheit zu erhöhen sind Zertifikate zeitlich begrenzt. Der Server kann aber auch bei einem zeitlich noch gültigen Zertifikat bei der CA nachfragen, ob das Zertifikat eventuell zurückgezogen und für ungültig erklärt wurde. Das kann der Fall sein, wenn sich Daten des Besitzers geändert haben, ihm die Gruppenzugehörigkeit entzogen wurde oder das Zertifikat gestohlen ist.

Kapitel 3

Legitimation der Aufgabe

Zur Aufgabenstellung gibt es natürlich auch eine entsprechende Hintergrundgeschichte, welche die Bedeutung der Aufgabe und ihre Einbettung in die Chemnitzer Informatik erklärt. Seit 1995 gibt es an der Universität Chemnitz ein Internet-Aufbaustudium "Informations- und Kommunikationssysteme" [11]. Die Teilnehmer, der von der Universität Chemnitz und dem VDE angebotenen Kurse, müssen nur zur Anmeldung und zur Abschlußprüfung persönlich erscheinen. Die eigentliche Ausbildung erfolgt ausschließlich am PC, die Arbeitsmaterialien werden über das Internet bezogen. Da diese Ausbildung kostenpflichtig ist muß sichergestellt werden, daß nur "Zahlende Gäste" Zugang zu den entsprechenden Betriebsmitteln erhalten. Deshalb wird jeder Nutzer, der sich auf die Teilnehmerseiten von iuk.tu-chemnitz.de begibt, nach einer USER ID und dem zugehörigen Passwort gefragt.

Diese Art der Authentifikation ist, da sie leicht realisiert werden kann, weit verbreitet. Das bedeutet jedoch nicht, daß sie nicht verbessert werden kann, denn ein möglicher Angriffspunkt ist die Eingabe und Übertragung von Passwörtern. Das Problem, daß sensible Daten über einen unsicheren Kanal zwischen zwei Computern übertragen werden, kann man durch Verschlüsselung lösen.

Das Nutzer versuchen können aus einer unsicheren Umgebung heraus eine Verbindung aufzubauen ist ein weiteres Problem. Trotz aller Verschlüsselung darf nicht vergessen werden, daß die Kommunikation vom und zum Nutzer im Klartext verläuft und daran wird sich ohne den Einsatz von Transplantaten auch nichts ändern. Damit muß sich jeder, der keine Steckdose hinter dem Ohr trägt, bei der Authentifikation durch Wissen fragen, ob er sein Wissen in der jeweiligen Umgebung preisgeben möchte. Um das zu vermeiden bleibt einem nur die Möglichkeit ganz auf Passwörter zu verzichten und andere Authentifikationsformen zu wählen.

Dieses Problem gilt besonders in den für den Nutzer unbekannten Umgebungen. Während er in seinem gewohnten Umfeld einen Überblick über die verwendete Infrastruktur hat und ihm auch die in angrenzenden Büros arbeitenden Leute vertraut sein sollten, kann ein Besucher eines Internetcafes nicht mit Sicherheit sagen, welche Aufgaben ein kleiner grauer Kasten im Kabelgewirr hinter dem Rechner erfüllt und ob der Kasten gestern schon da war.

Es kann aber auch versucht werden, das Risiko durch einen administrativen Eingriff zu verringern. So kann zum Beispiel festgelegt werden, daß ein Nutzer nur von einem bestimmten

Rechner (seinem eigenen) einen Zugriff auf die Betriebsmittel des Webservers erhält. Eine solche Maßnahme beinhaltet aber eine starke Einschränkung der Verwendbarkeit.

Da Computer im Internet anhand ihrer IP-Adresse identifiziert werden, darf sich diese bei einer fest eingetragenen Zugangsberechtigung von Rechner sowieso nicht verändern. Wenn sich aber jemand per Notebook und Modem anmelden möchte, wird die IP-Adresse, die sein mobiler Rechner bekommt, dynamisch vergeben. Er erhält möglicherweise bei jeder Anmeldung eine andere IP-Adresse. Ob das Verfahren mit dem Überprüfen der IP- Adresse eingesetzt werden kann, muß demnach von Fall zu Fall geprüft werden. Es kann sich deshalb höchstens um eine Ergänzung zu anderen Sicherheitsvorkehrungen handeln.

Eine andere Möglichkeit einen Verbindungsaufbau von einem potentiell unsicheren Rechner zu erschweren, ist der bereits beschriebene Einsatz von Zertifikaten. Ein Zertifikat stellt Besitz dar und der muß transportiert werden. Bei einem Rechnerwechsel wegen Umzug oder Aufrüstung kann ein Zertifikat problemlos per Diskette oder ähnlichem auf den neuen Rechner übertragen werden. Da eine solche Aktion gezielt erfolgen muß, können "Unfälle", durch die Sicherheitslücken entstehen, weitgehend vermieden werden. Niemand geht aus Versehen mit einer Diskette seines Zertifikates in ein Internetcafe, installiert es dort und baut eine Verbindung zum Webserver auf. Das eine solche Handlungsweise fahrlässig ist, sehen sicher auch die ein, die sich eventuell von dort per Nutzername und Passwort angemeldet hätten.

Die Aufgabe eine webbasierte Authentifikation mittels Zertifikaten zu ermöglichen, ist wie ich eben gezeigt habe, nicht nur theoretischer Natur. Lösungen dieser Art können sofort überall eingesetzt werden, um webbasiert auf sensible Daten in einem Netzwerk zuzugreifen. Die Nutzer müssen nur die Möglichkeit haben, ihr Zertifikat sicher auf ihrem lokalen Computer zu deponieren.

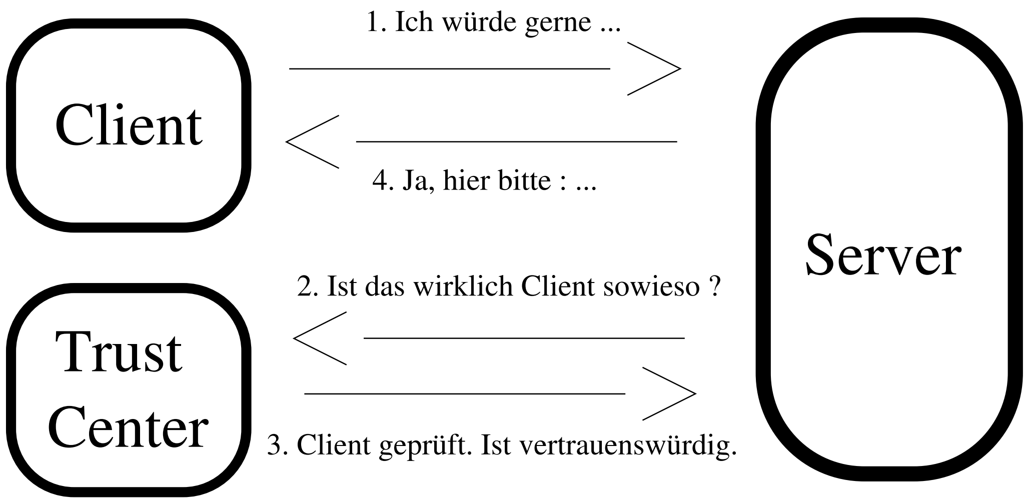


Abbildung 3.1: Prinzip webbasierte Authentifikation

Kapitel 4

Grundbedrohungen

Alle gezeigten Bedrohungen lassen sich auf drei Grundbedrohungen zurückführen.

4.1 Verfügbarkeit

Eine Bedrohung die lange Zeit unterschätzt wurde, ist der Verlust der Verfügbarkeit. Unterschätzt deswegen, weil angenommen wurde, daß nur Systeme mit sensiblen Daten bedroht werden können. Wenn jedoch die Seiten eines frei zugänglichen Webservers gelöscht werden, ist der Schaden oft nicht unerheblich. Das liegt daran, daß die Informationen bis zur Wiederherstellung eben nicht zur Verfügung stehen und die Wiederherstellung selbst natürlich auch einen Aufwand bedeutet. Die verursachten Kosten setzen sich überwiegend aus den vorbeugend einzusetzenden Backupmechanismen und den Verzögerungen durch das Fehlen von Information zusammen.

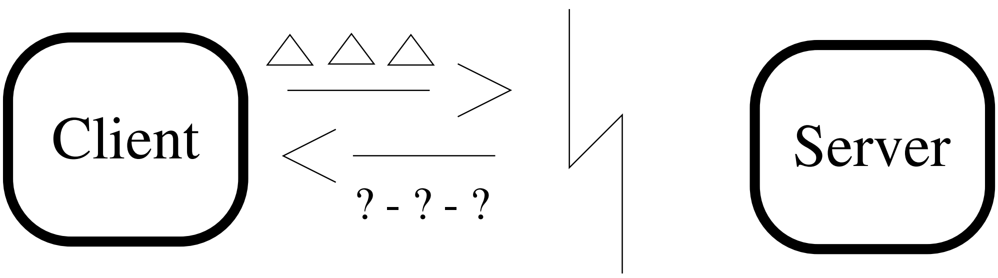


Abbildung 4.1: Verfügbarkeitsverlust

Als Beispiel möchte ich das Löschen oder Unbrauchbarmachen eines Bibliotheksrecherchesystemes nennen. Selbst bei einem Ausfall von nur wenigen Stunden kann die Summe der Wartezeiten leicht mehrere tausend Stunden betragen. Es gibt im wesentlichen zwei Arten des Verlustes der Verfügbarkeit. Zum einen können äußere Einflüsse oder höhere Gewalt für einen Ausfall zuständig sein. Das sind dann meist Schwierigkeiten mit zuviel oder zuwenig Strom, unterbrochenen Leitungen und ähnlichem. Relevant für meine Untersuchungen ist jedoch nur die andere Art von Verfügbarkeitsverlust, die entweder absichtlich oder fahrlässig herbeigeführt unter Zuhilfenahme eines Computers erfolgt. Es handelt sich hier meist um das Löschen, Überschreiben oder sonst irgendwie unbrauchbar Machen von Daten.

4.2 Integrität

Eine zweite Bedrohung ist der Verlust der Integrität. Im Unterschied zum oben beschriebenen Totalausfall, ist eine Veränderung von Daten oft schwerer zu erkennen. Solche Manipulationen können über einen längeren Zeitraum erfolgen und verringern dadurch die Chance einer erfolgreichen Wiederherstellung der Daten durch Backups. Je älter die Daten für die Rekonstruktion sind, desto geringer ist zwar die Wahrscheinlichkeit einer Manipulation aber desto unvollständiger bzw. ungenauer sind sie.

Das bekannteste Beispiel für Integritätsverlust ist die “man in the middle attack”, bei der ein zwischen die Kommunikationspartner geschalteter Angreifer, der sich bei beiden Beteiligten für den jeweils anderen Partner ausgibt.

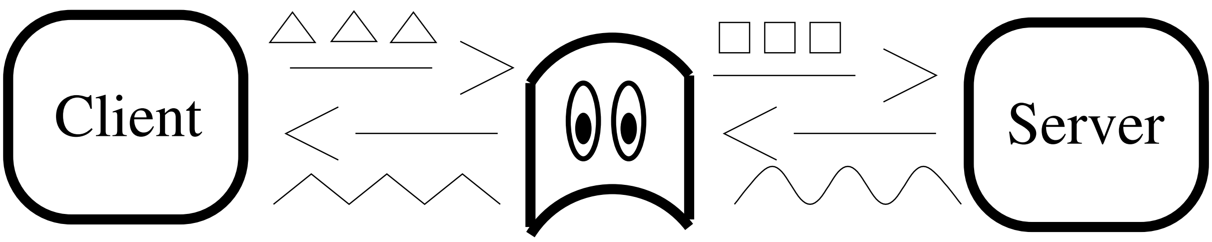


Abbildung 4.2: man in the middle attack

4.3 Vertraulichkeit

Die wohl am häufigsten auftretende Bedrohung ist der Verlust der Vertraulichkeit. Am häufigsten deswegen, weil die Vertraulichkeit oft genug unbeabsichtigt verlorengeht. Denn unabhängig davon, wie Daten übertragen werden, sind sie oft genug ohne besonderen Schutz "gelagert". Das gilt besonders für Informationen, die zwar elektronisch erstellt und bearbeitet werden, deren Archivierung dann aber auch auf Papier erfolgt. Jeder kennt sicher Schreibtische, auf denen sich aufgeschlagene Schnellhefter und Aktenordner stapeln. Da kann sich schon mal ein Zettel seinen Weg von einem Ordner zum nächsten bahnen. Ein noch größeres Problem sind jedoch Besucher, deren Erscheinen zwar einen legitimen Grund hat, die jedoch kaum gezwungen werden können, die ganze Zeit aus dem Fenster zu sehen. Solche Besucher können sicher leicht die eine oder andere Überschrift erhaschen. Es müssen nicht immer Passwörter sein. Die Gerüchteküche arbeitet schnell und ohne einen Quellennachweis zu fordern.

Nun ist der Computer oft Bibliothek, Büro und Aktentasche in einem Gerät. Grund genug etwas Sorgfalt walten zu lassen und dem Geschehen etwas Aufmerksamkeit zu schenken. Selbst bei einem gezielten Angriff zur Informationsbeschaffung, weiß das Opfer oft nicht, das es bestohlen wurde. Wir haben es schließlich mit nichts materiellem zu tun. Das jemand über bestimmte Informationen verfügt, läßt sich daher erst nachweisen, wenn diese verwendet werden und manchmal nicht einmal dann. Die Verwendung eines gestohlenen, erlauschten Passwortes, kann anhand der Protokolldateien des benutzten Computers ermittelt werden. Viele Systeme zeigen nach erfolgreicher Anmeldung wann und von wo die letzte Anmeldung erfolgte. Wenn aber eine gestohlene Information verwendet wird, um das Angebot eines Konkurrenten knapp zu über- oder unterbieten und einen Auftrag zu erhalten, ist ein Nachweis des Diebstahls anhand seiner Auswirkungen nicht möglich.

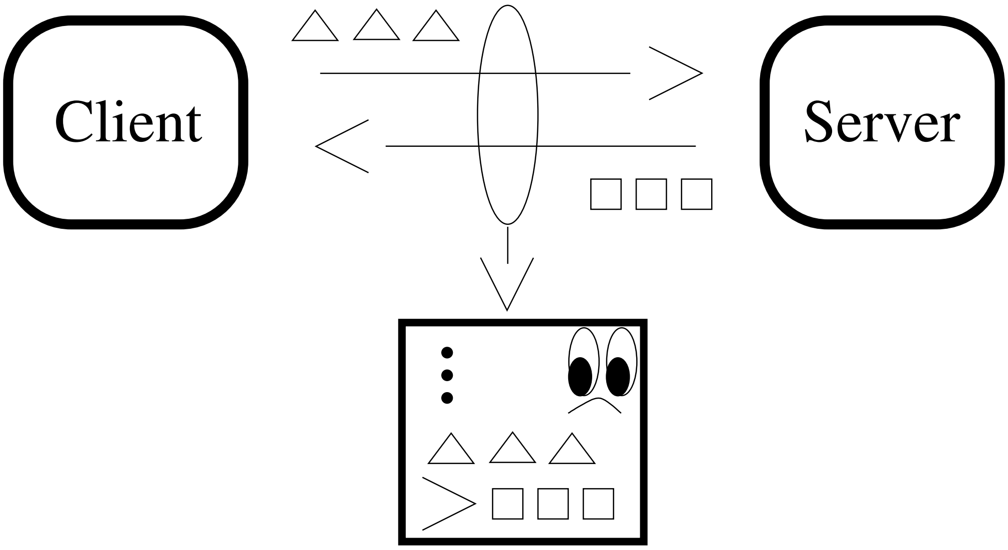


Abbildung 4.3: Vertraulichkeit geht verloren

4.4 Authentizität

In der Literatur wird oft von einem Verlust der Authentizität als weiterer Bedrohung gesprochen. Vom Erscheinungsbild her ist dieses Problem dem Verlust der Integrität sehr ähnlich. Bei einem Verlust der Authentizität wird versucht, Zugang mittels gefälschter Identität zu erlangen. Beim Verlust der Integrität werden Informationen verfälscht. Für den Empfänger von manipulierten Daten spielt es dabei keine Rolle, ob die Information als Falschmeldung erzeugt wurde oder eine reguläre Anforderung verändert wurde. Der Punkt ist, daß veränderte Informationen unbemerkt weitergegeben werden, was sonst weder beim Verlust der Vertraulichkeit noch beim Verlust der Verfügbarkeit der Fall ist. Der Verlust der Authentizität stellt damit zwar eine Bedrohung dar, ist jedoch keine Grundbedrohung.

4.5 Kombinationen

In der Praxis ist es häufig so, daß eine Bedrohung durch eine andere entsteht oder zumindest wahrscheinlicher wird. Ein Verlust von Integrität oder Verfügbarkeit ist oft auf einen Verlust von Vertraulichkeit zurückzuführen. Das Erlangen vertraulicher Informationen kann also als Einstieg in die Computerkriminalität betrachtet werden, da es hier eine sehr breite Grauzone gibt, in der nicht mit Sicherheit gesagt werden kann, ob sich ein Besitzer vertraulichen Materials, daß nicht für ihn bestimmt ist, krimineller Handlungen schuldig gemacht hat, um an dieses Material zu gelangen. Es sollten also vorbeugende Maßnahmen ergriffen werden, durch die entsprechende Situationen von vornherein auszuschließen sind. Für die Lagerung brisanter Daten besteht eine solche Maßnahme seit der Vor - Computer - Ära im wegschließen der Daten. Das trifft heute natürlich noch genauso zu und deshalb sollte ein LockScreen beim kurzzeitigen Verlassen eines Rechners im Pool genauso selbstverständlich sein, wie das verschließen des Büros beim Verlassen des selben. Das eigentliche Sicherungsproblem besteht aber bei der Übertragung von Informationen.

Um sich in Netzwerken gegen Bedrohungen zu schützen ist es wichtig, die Struktur des Netzes und der Komponenten, die mit dem Netz arbeiten, zu kennen. Möglichkeiten, wie Netzwerkstrukturen erforscht und visualisiert werden können, sind in [16] beschrieben.

Kapitel 5

Prinzipien der Verschlüsselung

5.1 Der sichere Kanal

Erste Ansätze um einem Verlust der Vertraulichkeit vorzubeugen, stammen aus dem militärischen und aus dem diplomatischen Bereich. Es wurde versucht durch entsprechend aufwendige Chiffrierung, eine Nachricht für jeden Nichteingeweihten unverständlich zu halten.

Das setzt natürlich einige Bedingungen voraus. So müssen sich die beiden miteinander kommunizierenden Partner vor dem Übertragen der Nachricht auf ein Chiffriersystem und eventuell die zugehörigen Schlüssel einigen. Die Unterhaltung um eine solche Einigung zu erzielen, muß über einen sicheren Kanal geführt werden. Mit einem solchen (Übertragungs-)Kanal ist das Medium gemeint, über das die Nachricht Übermittelt wird. Dieser Kanal muß abhörsicher sein, um einen Schlüsselmißbrauch zu verhindern.

In der Praxis gibt es einen solchen sicheren Kanal nicht, denn sonst könnte man sich die Verschlüsselung sparen und nur sichere Kanäle verwenden. Deshalb ist es notwendig, daß sich die Partner persönlich zu einem Schlüsselaustausch treffen. Darauf, daß dies einen erheblichen Nachteil des Systems darstellt, muß sicher nicht näher eingegangen werden.

5.2 Realisierbare Ziele

Auf die Verfügbarkeit hat die Verschlüsselung keinen Einfluß. Verfügbar ist entweder Alles, was zwischen zwei Computern übertragen werden soll, oder Nichts. Das liegt daran, daß Informationen zum Verbindungsaufbau nicht verschlüsselt werden können, da Router und Gates, über die eine Kommunikation läuft, die Adressen der Beteiligten kennen müssen. Für den mechanischen Aspekt der Verfügbarkeit gilt diese "geht oder geht nicht" Eigenschaft ebenfalls. Ein Computer ist verfügbar, wenn eine physische Verbindung aufgebaut werden kann, oder ist nicht verfügbar, weil z.B. die Leitung unterbrochen ist.

Gegen die anderen bereits beschriebenen Bedrohungen, kann die Verschlüsselung eingesetzt werden. Sie wird also verwendet, um Integrität und Vertraulichkeit zu sichern. Weil über das Anfertigen verschlüsselter Nachrichten auch der Besitz der zugehörigen Schlüssel nachgewiesen wird, kann über diesen Umweg auch eine Authentifikation realisiert werden, denn wer in der Lage ist eine gültige verschlüsselte Nachricht zu erzeugen, muß dazu im Besitz der richtigen Schlüssel sein und ist mit großer Wahrscheinlichkeit der richtige Kommunikationspartner.

Bei all den phantastischen Sachen, die mit Verschlüsselung getan werden können, ist zu beachten, daß sie Sicherheitsbedürfnisse befriedigen soll und das am besten unauffällig, schnell und vollautomatisch. Während früher die Entschlüsselung einer Nachricht mühselig von Hand erfolgte wird sie heute vom PC am Arbeitsplatz durchgeführt ohne das der Nutzer eine Verzögerung bemerkt. Was jedoch bei vielen Verfahren negativ auffällt, ist der hohe Verwaltungsaufwand. Besonders beim Schlüsselmanagement besteht Bedarf für eine sanftere Integration in bestehende Applikationen.

Kapitel 6

Kryptosysteme

6.1 Symmetrische Kryptosysteme

Verfahren, bei denen sich Sender und Empfänger einen geheimen Schlüssel teilen, bezeichnet man als symmetrische Kryptosysteme, da zur Codierung derselbe Schlüssel eingesetzt wird, wie zur Decodierung und nur der Algorithmus in umgekehrter Reihenfolge abgearbeitet wird oder bei der Decodierung genau das Gegenteil, von dem getan wird, was zur Codierung erforderlich war. Bei vielen dieser Kryptosysteme wurde nicht nur der Schlüssel geheimgehalten, sondern auch die Arbeitsweise des Systems selbst, um noch einmal die Sicherheit zu erhöhen.

Von allen heute anerkannten Kryptosystemen ist auch deren Arbeitsweise bekannt, da eine Anerkennung und die damit einhergehende Überprüfung sicherheitsrelevanter Eigenschaften nur durch Diskussion in einer breiten Öffentlichkeit zustande kommen kann. Trotzdem ist es nicht trivial vom Chiffrat auf das verwendete Kryptosystem zu schließen, was in etwa der Tatsache entspricht, daß verwendete System geheimzuhalten.

Schlüssel und Kryptosystem geheimzuhalten war nötig, da es keine mathematischen Untersuchungen zur Sicherheit dieser Kryptosysteme gab, also ein Angriff auf das System ohne Schlüsselkenntnis nicht ausgeschlossen werden konnte. Das änderte sich erst, als die Verbreitung von Rechentechnik den Datenaustausch mit vielen verschiedenen Partnern ermöglichte.

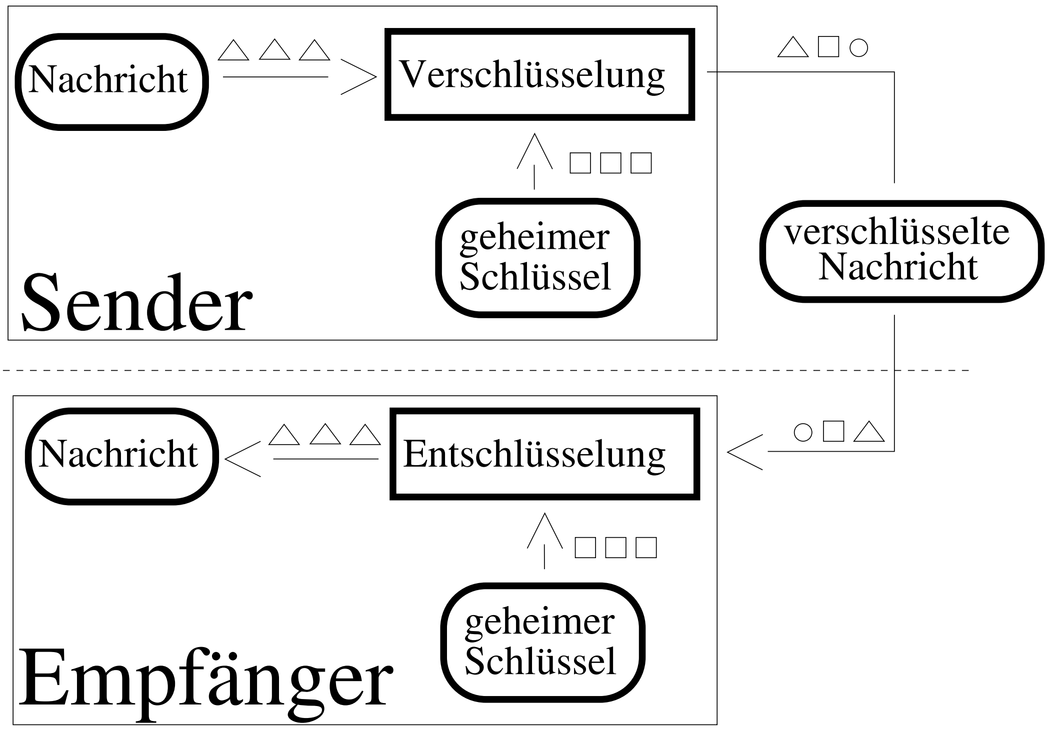


Abbildung 6.1: Funktionsweise symmetrischer Kryptosysteme

6.1.1 Standardisierung

Es war nötig, eine einheitliche Plattform zu schaffen, um auf deren Basis eine sichere Kommunikation zu ermöglichen. Eine solche Plattform bringt eine Reihe von Vorteilen mit sich. So wird zum Beispiel verhindert, daß viele verschiedene Algorithmen implementiert werden, mit deren Hilfe dann jeweils nur wenige Leute Daten austauschen können. Außerdem müßte irgendwann jeder diese Fülle von Algorithmen auf seinem Rechner haben. Die Implementierung vieler kleiner Systeme bindet, wie sich jeder vorstellen kann, so viele Kräfte, daß sie nur schlechter realisiert werden kann und mehr Sicherheitslücken enthält, als ein großes Kryptosystem. Das das Verfahren, nach dem dieses Kryptosystem arbeitet bekannt gegeben werden mußte, war nötig, weil Implementationen für verschiedene Rechnerarchitekturen realisiert werden mußten und diese Implementationen natürlich auch käuflich sein sollten. Ein Verstecken des Algorithmus ist also nicht möglich, denn der kann durch Analyse des Binärcodes ohnehin ermittelt werden.

Anfang der 70er Jahre wurde eine Liste mit Bedingungen für ein solches Kryptosystem erstellt. Bei IBM wurde zu dieser Zeit an einem System gearbeitet, daß bereits viele dieser Anforderungen erfüllte. Sie erhielten aus diesem Grund den Auftrag ihr System weiter zu entwickeln. Die Ergebnisse wurden von der NSA geprüft, getestet und 1977 unter dem Namen DES "Data Encryption Standard" von ANSI für die USA zum Standard erklärt.

Es wurden Hardwarerealisierungen gebaut, um zu Ermöglichen, daß der Durchsatz der Codierung und Decodierung dem Datentransfer in Computernetzen in nichts nachsteht. In vielen UNIX-Derivaten steht der Befehl "des" noch heute zur Verfügung, sollte jedoch nur noch unter

Vorbehalt verwendet werden.

6.1.2 Verwendbarkeit

"Bis heute ist keine ernsthafte Verwundbarkeit von DES aufgedeckt worden. Zudem hat die Hardware einen zur vollständigen Schlüsselraumsuche notwendigen Entwicklungsstand noch nicht erreicht." schrieb Oppliger 1992 [2]. Im ersten genannten Punkt hat sich bis heute nichts geändert. Kryptoanalytisch kann der DES heute noch als sicher betrachtet werden. Anders sieht das allerdings beim Durchprobieren der Schlüssel aus.

Beim letzten öffentlichen Test, an dem im Januar 1999 etwa 100000 Computer per Internet und ein speziell zu diesem Zweck entwickelter Computer mit dem Namen DeepCrack teilnahmen, wurde der richtige Schlüssel in einer Zeit von nur 22 Stunden und 15 Minuten gefunden [3]. Nun ist einem einzelnen Angreifer nicht dieses Potential an Hardware gegeben, jedoch weiß auch niemand, wieviele Rechner ala DeepCrack weltweit im Einsatz sind.

Verschlüsselung mittels DES sollte also nur noch dort eingesetzt werden, wo die Nachricht innerhalb von Stunden ihre Bedeutung verliert, oder dann ohnehin bekannt ist. Da wo heute der DES eingesetzt wird, erfolgt meist eine Mehrfachverschlüsselung mit unterschiedlichen Schlüsseln. Bei einem Einsatz von Triple-DES, also einer Dreifachverschlüsselung, steigt damit die Schlüssellänge auf 168 Bit. Die Anzahl möglicher Schlüssel steigt somit von 2^56 auf 2^168, einer Größe die momentan jedem Angriff standhalten sollte.

Wenn bei einer Ausschreibung jeder sein Angebot wenige Stunden vor dem Abgabetermin zuschickt, hat ein potentieller Konkurrent keine Möglichkeit sich daran zu orientieren, da er die Nachricht erst zu einem Zeitpunkt aus dem Chiffrat extrahieren kann, zu dem sie ihm nichts mehr nützt. Das erhöht leider nicht die Sicherheit des Kryptosystems selbst. Der hohe Datendurchsatz, der einst als Vorteil vom DES gepriesen wurde ist nun sein größter Nachteil, da dieser Vorteil mit, aus heutiger Sicht, zu kurzen Schlüsseln erkauft wird. Außerdem sind diese Schlüssel konstant 56 Bit lang. Ein skalieren der Schlüssellänge, wie bei anderen Systemen, ist leider nicht möglich.

6.2 Asymmetrische Kryptosysteme

Revolutioniert wurden Kryptosysteme seit 1976 nach dem veröffentlichen einer Arbeit durch Whitfield Diffie und Martin Hellman [DIF76]. Diese beiden Wissenschaftler legten den Grundstein für eine neue Art von Kryptosystemen, den Public-Key-Kryptosystemen. Zwar gilt heute als ziemlich sicher, daß die Idee eines solchen Verfahrens bereits vor Veröffentlichung der Arbeit bei einigen Institutionen bekannt gewesen ist [4], die Forschung auf diesem Gebiet setzte jedoch erst zu diesem Zeitpunkt ein.

Der Name Public-Key-Kryptographie rührt daher, daß es hier keinen geheimen Schlüssel für Codierung und Decodierung gibt, den sich alle beteiligten Kommunikationspartner teilen, sondern zur Kommunikation zwei Schlüssel erforderlich sind. Einer dieser Schlüssel wird Private-Key genannt, der andere Public-Key, daher auch der Name. Public-Key-Kryptosysteme werden heute zumeist als Asymmetrische-Kryptosysteme bezeichnet, um gleich in der Namensgebung eine Unterscheidung zu Systemen mit einem gemeinsamen Schlüssel, wie dem DES, zu erhalten.

Wenn jemand seinem Gegenüber eine verschlüsselte Nachricht zukommen lassen möchte und dazu ein asymmetrisches Verfahren einsetzt, dann verschlüsselt er seine Nachricht mit dem öffentlichen Schlüssel (public key) des Empfängers. Nachdem er das getan hat, besteht für ihn selbst keine Möglichkeit mehr, die Nachricht aus dem Chiffrat zu extrahieren. Alle potentiellen Angreifer, die nur über den öffentlichen Schlüssel verfügen, können dies natürlich ebenfalls nicht.

Der einzige Weg wieder an die Nachricht zu gelangen, führt über den privaten Schlüssel (private key) und der sollte sich nur beim rechtmäßigen Besitzer befinden. Die Verantwortung für das Schlüsselmanagement, also die Generierung und Aufbewahrung des Schlüsselpaares, unterliegt jedem selbst. Ein Gefahrenpotential, wie es das bei symmetrischen Systemen durch die Kenntnis mehrerer Leute von einem bestimmten Schlüssel gab, existiert nicht mehr.

Ein großer Vorteil, den es mit sich bringt, wenn ein Schlüssel zum codieren und einer zum decodieren eingesetzt wird, ist der, daß sich die Kommunikationspartner nicht persönlich treffen müssen, um sich auf Schlüssel zu einigen. Wenn jeder seine beiden Schlüssel generiert und den öffentlichen an geeigneter Stelle da hinterläßt, wo ihn mögliche Kommunikationspartner finden können, steht einer gesicherten Übertragung nichts mehr im Weg. Ein Verlust der Vertraulichkeit muß auf jeden Fall nicht befürchtet werden, wenn die Authentizität der Schlüssel sichergestellt ist.

Es genügt also nicht, den Schlüssel irgendwo gut erreichbar zu deponieren, wenn es an dieser Stelle eine übergeordnete Einrichtung gibt, die den Schlüssel austauschen könnte. Eine solche administrative Instanz hat aber fast jeder bei seiner Homepage, einem FTP-Verzeichnis oder ähnlichem. Eine mögliche aber umständliche Lösung ist es, seinen Schlüssel an vielen verschiedenen Orten zu deponieren und die Liste dieser Orte beizufügen. Wenn diese Depots in unterschiedlichen Einflußbereichen liegen, sollte es einem Angreifer schwerfallen alle Schlüssel auszutauschen. Ein Kommunikationspartner kann dagegen mit erträglichem Aufwand die Gleichheit der Schlüssel prüfen.

Die praktikablere Lösung ist dagegen, sich die Echtheit seines Schlüssels von einer dritten Instanz bestätigen zu lassen. Im einfachsten Fall kann das die Datenbank eines Trust-Centers sein, in der jeder Interessierte nach öffentlichen Schlüsseln möglicher Kommunikationspartner suchen kann. Wenn man diese Idee zu Ende denkt, kommt man jedoch unweigerlich zu Zertifikaten und die sind von den Certification Authorities mit Bedacht auszustellen, können dann aber bedenkenlos gelagert werden.

Neben der Vertraulichkeit steht aber auch die Integrität ganz oben auf dem Wunschzettel der Anwender. Nach der jetzigen Beschreibung asymmetrischer Kryptosysteme hätte ein Angreifer die Möglichkeit eine Nachricht abzufangen, deren Zieladresse und damit den öffentlichen Schlüssel des Empfängers zu ermitteln. Danach könnte er eine eigene Nachricht verfassen und diese eventuell im Namen des Absenders an den Empfänger weiterleiten. Um das zu verhindern, besteht bei asymmetrischen Kryptosystemen die Möglichkeit seine Nachricht digital zu signieren.

Es ist nicht vorgeschrieben, welcher der beiden Schlüssel der öffentliche und welcher der private ist. Der Nutzer kann vor dem ersten Einsatz wählen, welchen von beiden er bekanntgibt. Was mit einem Schlüssel codiert wurde kann immer mit dem anderen decodiert werden. Dadurch ist es möglich, eine Nachricht mit seinem eigenen privaten Schlüssel zu codieren. In einem uncodierten Teil, der an die Nachricht angefügt wird, wird dann beschrieben, von wem die Nachricht stammt und wo der zugehörige öffentliche Schlüssel zu finden ist. Der Empfänger erhält so die Möglichkeit den Absender zu überprüfen und mit dessen öffentlichem Schlüssel die Nachricht zu extrahieren.

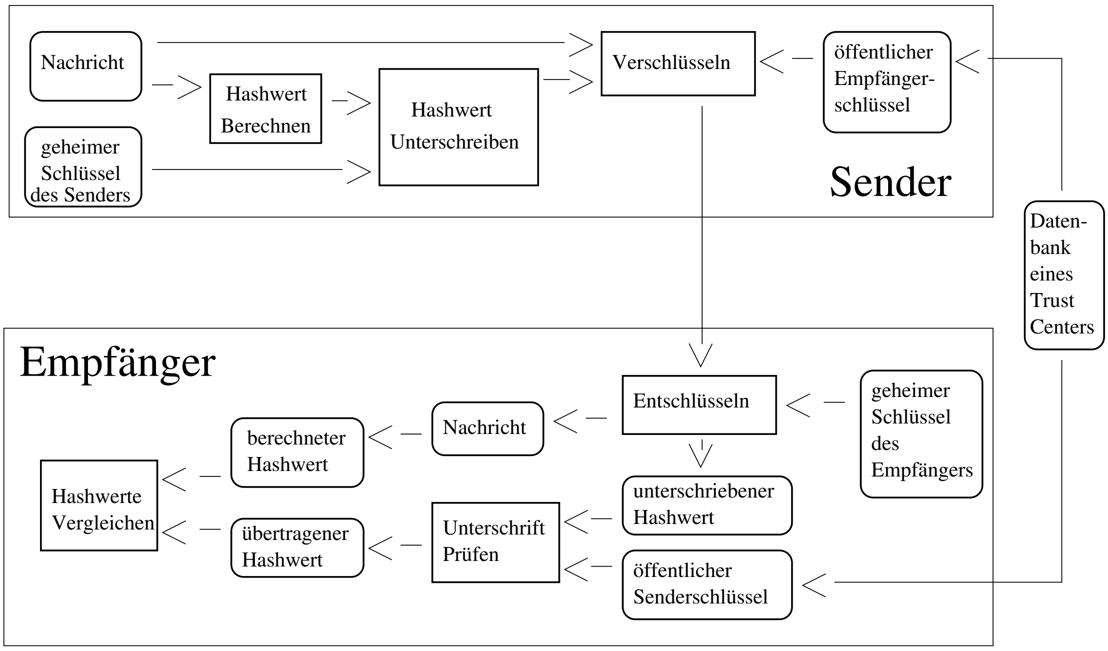


Abbildung 6.2: Asymmetrisches Kryptosystem mit Signatur

Da asymmetrische Kryptosysteme sehr rechenintensiv sind, wird meist nicht die gesammte Nachricht, sondern ein aus der Nachricht zu berechnender Hashwert unterschrieben. Das bringt vor allem bei Informationen, die nicht vertraulich sind und öffentlich bekannt gegeben werden den Vorteil, daß nachträgliche Veränderungen nicht dem ursprünglichen Autor untergeschoben werden können. Nach einer Änderung der Nachricht stimmen alter und neuer Hashwert nicht mehr überein, wodurch ein erneutes Signieren des Hashwertes mit einem geheimen Schlüssel erforderlich wird. Die Empfänger der Nachricht, also mögliche News Leser, können dann selbst entscheiden, wie mit einer Nachricht zu verfahren ist, der kein gültiger Absender zugeordnet werden kann. Eine Schwierigkeit die dann noch besteht, ist, daß nur schwer ermittelt werden kann, wer ursprünglicher Autor einer Nachricht ist, denn Zurückdatieren und neu Verpacken kann jeder.

Die Integrität einer Nachricht kann also gewährleistet werden. Maßnahmen zum Erhalt der Integrität und Vertraulichkeit lassen sich auch miteinander kombinieren. So kann eine Nachricht nach ihrer Erstellung erst mit dem eigenen privaten Schlüssel unterschrieben werden um eine nachträgliche Manipulation zu vermeiden. Anschließend codiert man die Nachricht samt Unterschrift mit dem öffentlichen Schlüssel des Empfängers, um die Vertraulichkeit zu wahren.

6.2.1 NP-vollständige Probleme

Bis jetzt habe ich über die Anwendungsmöglichkeiten asymmetrischer Kryptosysteme geschrieben, im Folgenden soll der theoretische Hintergrund beleuchtet werden. Die Idee, die allen solchen Systemen zu Grunde liegt, ist die, daß es mathematische Einwegfunktionen gibt. Das sind Funktionen, deren Lösung einfach zu realisieren ist, wogegen ihre Umkehrung nahezu unmöglich ist. Einfach zu realisieren bedeutet, daß die Lösung mit polynomialem Aufwand berechnet werden kann. Nahezu unmöglich heißt dagegen, daß der Aufwand zur Lösung des Problems mindestens ein exponentielles Wachstum im Vergleich zur Eingabe aufweißt. Probleme deren Lösungsaufwand jenseits des polynomialen liegt, nennt man NP-vollständig.

Eines der wohl bekanntesten Probleme dieser Art, und das am meisten eingesetze in der Kryptographie, ist das Faktorisierungsproblem. Es ist einfach zwei Zahlen miteinander zu multiplizieren, jedoch sehr schwierig, vom Produkt auf die verwendeten Faktoren zu schließen. Da das Ermitteln der Faktoren stark erleichtert wird, wenn das Produkt aus vielen Faktoren besteht, werden zur Berechnung Primzahlen verwendet. Durch den Einsatz von Primzahlen wird gewährleistet, daß das Produkt nur aus exakt zwei Faktoren besteht.

Seit der Verwendung des Faktorisierungsproblems in der Kryptographie wurden eine Reihe von Algorithmen geschrieben, um den Bedarf an großen Primzahlen zu decken. Die Primzahlen müssen schließlich dort berechnet werden, wo daraus die beiden Schlüssel für das public-key Kryptosystem entstehen sollen, und das ist normalerweise der lokale Computer des Nutzers.

Im Gegenzug wird versucht, im Rahmen der Kryptoanalyse, Wege zur Faktorisierung großer Zahlen zu finden. Bislang wurde keine generelle Lösung gefunden, die gefundenen Ansätze schränken allerdings die Wahl der zu verwendenden Primzahlen ein. Neben der Aussage, daß die verwendeten Primzahlen möglichst groß sein sollen, um ein durchprobieren zu verhindern, sollten sie unter anderem auch weit auseinanderliegen, da es sich sonst aus Primzahlen handelt, die ähnlich groß wie die Wurzel der zu faktorisierenden Zahl sind, was auch wieder eine erhebliche Eingrenzung des Suchraumes ermöglicht.

6.2.2 RSA

Die verbreiteteste Implementierung, eines Kryptosystems auf der Basis des Faktorisierungsproblems ist das RSA Verfahren. Es wurde zwei Jahre nach der Arbeit von Diffie und Hellmann vorgestellt und ist nach dessen Entwicklern Rivest, Shamir und Adleman benannt. Der Tatsache, daß es sich um eine sehr frühe Implementierung eines asymmetrischen Kryptosystems handelt und bis heute kein erfolgreicher Angriff bekannt wurde, verdankt dieses Kryptosystem seine große Popularität. Weil die Schlüssellänge vom Benutzer festgelegt wird und nicht fest vorgegeben ist, konnte in den letzten 20 Jahren mit der Verbesserung der Hardware auch immer die Schlüssellänge erhöht werden.

Das RSA Verfahren wird heute in vielen Bereichen eingesetzt da es sowohl zu Signaturzwecken als auch zur Verschlüsselung eingesetzt werden kann. Ein Beispiel für die Verwendung von RSA ist sicher PGP. PGP steht für Pretty Good Privacy und wird zum Schutz von E-Mail und Dateien eingesetzt. In PGP kann mit dem eigenen Private-Key unterschrieben und mit dem Public-Key des Empfängers verschlüsselt werden. Großer Vorteil des PGP Paketes ist, daß es so in einige der gängigen Mailprogramme integriert werden kann, das die Benutzer mit PGP fast nicht mehr in Berührung kommen und der Arbeitsaufwand dadurch minimiert wird. Es lassen sich globale Einstellungen treffen, in denen festgelegt wird, ob Mails digital zu unterschreiben und wenn möglich zu verschlüsseln sind.

6.3 hybride Kryptosysteme

Größter Nachteil asymmetrischer Kryptosysteme und damit auch Nachteil von RSA, ist die hohe Rechenleistung, die bei einem Einsatz benötigt wird. Das wird durch die Verarbeitung auf Universalprozessoren, die oft einen für diese Art von Kryptosystemen ungeeigneten Befehlssatz haben, noch gesteigert. RSA wird eben zunehmend auch im privaten Bereich eingesetzt und da spielt Bearbeitungszeit eine untergeordnete Rolle. Um dennoch akzeptable Resultate zu erzielen, ist ein möglicher Weg, die Vorteile symmetrischer und asymmetrischer Kryptosysteme zu vereinigen. Eine solche Kombination beider Systeme nennt man hybrides Kryptosystem. Es handelt sich dabei um keine neue Art von Kryptosystem, sondern um das zeitlich versetzte Abarbeiten eines asymmetrischen und danach eines symmetrischen Verfahrens.

Bei dieser Vorgehensweise erfolgt mittels public-key Kryptosystem der Schlüsselaustausch für den anschließenden Einsatz eines viel schnelleren symmetrischen Verfahrens. Es spielt dabei keine Rolle, ob eine stehende, interaktive Verbindung zu einem Rechner aufgebaut werden soll oder es sich um eine reine Datenübertragung ohne Beteiligung des Empfängers handelt. Man kann also dem Empfänger einer Mail RSA chiffriert den DES Schlüssel mitteilen, mit dem der eigentliche Inhalt der Mail chiffriert ist. Dazu ist lediglich die Kenntnis des öffentlichen Schlüssels des Empfängers nötig.

Ein Programmpaket, bei dem dieses hybride Verfahren eingesetzt wird um Verbindungen zu anderen Computern aufzubauen, ist SSH. Die Abkürzung bedeutet SecureSHell und weist schon darauf hin, daß zu übertragende Datenpakete unter einer sicheren Schale verpackt werden. Dabei können zur Authentifizierung die unterschiedlichsten Verfahren eingesetzt werden und nur als letzte Möglichkeit wird nach einem Passwort gefragt, daß aber auch nur chiffriert übertragen wird. Auf prinzipielle Möglichkeiten zur Authentifizierung, wurde in früheren Abschnitten bereits eingegangen. Für die technische Realisierung bei SSH möchte ich den interessierten Leser an die Manual-Pages verweisen, da hier neben der Beantwortung syntaktischer Fragen auch auf das Umfeld beim Einsatz eingegangen wird.

SSH wurde entwickelt, um eine Reihe weit verbreiteter aber unsicherer Netzwerkprogramme abzulösen oder zu ergänzen. Zu einer sicheren Datenübertragung ala FTP wird innerhalb des Paketes das Programm SCP (Secure CoPy) bereitgestellt. Bei der Verwendung von unsicheren Programmen auf einem entfernten Computer, bei denen lediglich die Ein- und Ausgabe vom lokalen Computer aus erfolgt, übernimmt SSH selbst die Sicherung der Daten. SSH bildet hier im einfachsten Fall eine sicher Alternative zu Telnet, dem Klassiker der seit den Anfängen von Computernetzen die Möglichkeit zum Ausführen von Programmen auf entfernten Computern liefert. SSH sichert dabei auch alle Folgeverbindungen, die im Rahmen der bestehenden Verbindung aufgebaut werden. Das betrifft vor allem die Ein- und Ausgaben von an den lokalen Computer weitergeleiteten X11-Fenstern für die sonst keine anderen Sicherungsmöglichkeiten bestehen.

Programmpakete wie SSH sind dabei in der Lage unterschiedliche symmetrische Kryptosysteme einzusetzen weil die Gesetzgebung von Land zu Land verschieden ist und nicht überall jedes Kryptosystem eingesetzt werden darf. Außerdem liegt es im Bereich des Möglichen, daß ein Kryptosystem total gebrochen wird, d.h. ein Weg gefunden wird mit erträglichem (polynomialem) Aufwand ohne Schlüsselkenntnis in den Besitz der Nachricht zu gelangen. Um in einem solchen Fall schnell Reagieren zu können verfügt die aktuelle Version von SSH über die Möglichkeit, die Daten nach der Authentifizierung mit IDEA, BLOWFISH, ARCFOUR, DES und 3DES zu chiffrieren.

Die Mechanismen zur Authentifizierung bei SSH sind darauf abgestimmt zu überprüfen, ob der Nutzer auf dem Zielrechner ein login hat und demzufolge in der Lage ist Prozesse zu starten oder auf das Dateisystem zuzugreifen. Der Nutzer muß also auf beiden kommunizierenden Computern über eine Reihe von Betriebsmitteln verfügen können und das unabhängig davon, ob diese bei der Kommunikation tatsächlich verwendet werden.

Der Aufbau von sicheren Verbindungen zum Zwecke der Informationsbeschaffung mit Unterstützung von SSH ist also nur möglich, wenn der Nutzer auf dem Zielrechner ein login hat. Das ist aus vielerlei Gründen nicht möglich. Man stelle sich nur vor, daß alle n Webserver weltweit jeden der m potentiellen Nutzer kennen müßte. Bei einer webbasierten Authentifikation müssen demnach andere Wege gegangen werden.

Um eine sichere Verbindung im Web zu realisieren, müssen also andere Wege beschritten werden. Die bislang beschriebenen Techniken können dabei natürlich eingesetzt werden. Lediglich zur Authentifizierung müssen andere Maßnahmen eingesetzt werden.

Im einfachsten Fall erfolgt die Datenbeschaffung von einem Web-Server durch einen Verbindungsaufbau auf Port 80 des Servers, gefolgt von einer Anfrage zum Erhalt eines Dokumentes. Nach der Übertragung der geforderten Information, oder einer Meldung, daß dies nicht möglich war, wird die Verbindung durch den Server wieder geschlossen. Das für die Übertragung verwendete Protokoll ist so einfach, das es unter Umständen ohne Browser durch die Verwendung von Telnet realisiert werden kann.

```
mmr@pumuckl:~ > telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta name="GENERATOR" content="Mozilla/4.5 [en] (X11; I; Linux 2.0.36
i586) [Netscape]">
</head>
```

<body background="/foggy2.gif" nosave>

You are connected to pumuckl.csn.tu-chemnitz.de

</body>
</html>
Connection closed by foreign host.
mmr@pumuckl:~ >

Kapitel 7

Secure Socket Layer

Um nun dem bestehenden Sicherheitsbedürfnis gerecht zu werden, ist von der Netscape Communications Corporation das SSL Protokoll entwickelt wurden. SSL steht für Secure Socket Layer. Dieses Protokoll ist speziell für den Datenaustausch im WWW entwickelt wurden. Das zeigt sich schon beim Verbindungsaufbau, denn bei SSL authentifiziert sich der Server gegenüber dem Clienten. Das ist etwa so, als müßte derjenige, der angerufen wird zuerst seinen Namen nennen, ohne zu wissen wer am anderen Ende ist. Im WWW hat diese Reihenfolge aber ihre Berechtigung, da Information zwar an jeden weitergegeben wird, aber trotzdem die Quelle prüfbar sein muß.

Als nächstes einigen sich beide Teilnehmer auf ein gemeinsames Kryptosystem, mit dem Daten chiffriert werden sollen. Je nach eingestellter Sicherheitsstufe kann der Server anschließend auch eine Authentifikation vom Clienten verlangen. Wenn das erfolgreich verläuft müssen sich Server und Client nur noch auf den Schlüssel zum vereinbarten Kryptosystems einigen um mit der Kommunikation zu beginnen. Diesen Schlüssel nennt man auch verteiltes Geheimnis, weil nur diese beiden entfernten Kommunikationspartner Kenntnis davon haben.

Man unterscheidet innerhalb des SSL Protokolls zwischen zwei Teilen. In dem Teil, der für den Verbindungsaufbau zuständig ist, erfolgt die Authentifikation und werden Maßnahmen getroffen, um später die Integrität der Daten zu gewährleisten. Der zweite Teil übernimmt dann nach erfolgreichem Verbindungsaufbau den Datentransfer.

Da das SSL-Protokoll die praktische Grundlage für meine Arbeit liefert, möchte ich nun detailiert auf den dabei verwendeten Verbindungsaufbau, also dem ersten Teil des Protokolls eingehen. Die Grundlagen hierzu lieferte Netscape selbst [5].

1. Der Client sendet dem Server seine SSL Versionsnummer, Angaben zu möglichen Chiffren, Zufallsdaten und einige weitere Daten, die jedoch auch ohne die Verwendung von SSL nötig wären.

2. Der Server teilt dem Clienten seine Versionsnummer mit, wählt das stärkste symmetrische Kryptosystem aus, mit dem beide arbeiten können. Er überträgt Zufallsdaten und sein eigenes Zertifikat an den Clienten. Wenn der Client ein Betriebsmittel anfordert, für das eine Authentifikation erforderlich ist, wird das Zertifikat des Clienten angefordert.

3. Der Client benutzt die vom Server erhaltenen Informationen, um diesen zu authentifizieren. Das geschieht erfolgreich, wenn alle der folgenden Fragen mit "ja" beantwortet werden. - Ist das aktuelle Datum innerhalb der Gültigkeitsperiode des Serverzertifikates ? - Kann der ausstellenden CA vertraut werden ? - Kann mit Hilfe des öffentlichen Schlüssels der CA die Unterschrift des Zertifikates bestätigt werden ? - Stimmt der im Zertifikat eingetragene Domainname mit dem tatsächlich verwendeten überein ? Wenn der Server nicht authentifiziert werden kann, erhält der Nutzer eine Warnung und muß selbst entscheiden, ob der Verbindungsaufbau fortgesetzt werden soll.

4. Vom Clienten wird, in Abhängigkeit des vom Server ausgewählten Kryptosystems, ein Geheimnis erzeugt, aus dem dann später der Sitzungsschlüssel generiert wird. Dieses Geheimnis wird dann mit dem öffentlichen Schlüssel des Servers, der dessen Zertifikat entnommen wurde, verschlüsselt und dem Server zugesandt.

5. Wenn der Server die Authentifikation des Clienten fordert, dann unterschreibt der Client einige vom Server empfangene Zufallsdaten und sendet das Unterschriebene zusammen mit seinem Zertifikat an den Server.

6. Falls gefordert, versucht der Server nun den Clienten zu authentifizieren. Dazu prüft er die unterschriebenen Zufallsdaten mit dem öffentlichen Schlüssel aus dem Zertifikat des Clienten und stellt so fest, ob das Zertifikat wirklich seinem Gegenüber gehört. Alle weiteren zu beantwortenden Fragen sind ähnlich denen der Serverauthentifikation. - Ist das aktuelle Datum innerhalb der Gültigkeitsperiode des Clientzertifikates ? - Kann der ausstellenden CA vertraut werden ? - Kann mit Hilfe des öffentlichen Schlüssels der CA die Unterschrift des Zertifikates bestätigt werden ? Wenn der Client nicht authentifiziert werden kann, dann wird die Sitzung beendet. Anderenfalls verwendet der Server seinen privaten Schlüssel, um an das vom Clienten übertragene Geheimnis zu gelangen.

7. Aus diesem Geheimnis wird sowohl vom Server als auch vom Clienten der Sitzungsschlüssel des symmetrischen Kryptosystems generiert, das zur Sicherung der übertragenen Daten verwendet werden soll.

8. Der Client sendet eine Nachricht um dem Server mitzuteilen, daß ab jetzt alle Nachrichten verschlüsselt ausgetauscht werden und eine weitere verschlüsselte Nachricht, um dem Server mitzuteilen, daß aus Sicht des Clienten der Verbindungsaufbau abgeschlossen ist.

9. Der Server sendet daraufhin seinerseits diese beiden Nachrichten.

Die Verbindung steht nach Abarbeitung dieser Punkte, der erste Teil des Protokolls wurde erfolgreich beendet und es kann an dieser Stelle begonnen werden, Informationen unter Verwendung des Sitzungsschlüssel auszutauschen.

Kapitel 8

Einrichtung des Clienten

8.1 Browser

Um bei meiner Arbeit auf das SSL-Protokoll zurückgreifen zu können, muß ich mich für einen Clienten entscheiden, der dieses Protokoll beherrscht. An dieser Stelle ist es naheliegend den Netscape Navigator zu verwenden, da hier die zu erwartenden Probleme am geringsten sein sollten. Im Web browsen können heute die meisten Filemanager. Die hier gestellten Anforderungen gehen aber deutlich darüber hinaus.

8.2 Zertifikate

Neben allgemeinen Einstellungen um den Browser in Betrieb zu nehmen, muß man ihm auch das Zertifikat des Clienten zukommen lassen, mit dem man sich später beim Server authentifizieren möchte. Dazu gibt es beim Netscape Navigator zwei Möglichkeiten.

8.2.1 Keygen

Die erste Möglichkeit, den Netscape Navigator mit einem Zertifikat zu versorgen, ist ihn per HTML-Tag anzuweisen, die für das Zertifikat erforderlichen Schlüssel zu generieren [6]. In diesem Fall steht irgendwo im Quelltext des zum Erhalt eines Zertifikates auszufüllenden Fragebogens :

```
<KEYGEN NAME="name"
        CHALLENGE="challenge string"
        KEYTYPE="type"
        PQG="pqg-params">
```

Die Parameter NAME und CHALLENGE sind immer zu übergeben. Mit KEYTYPE kann zwischen dem RSA und dem DSA Verfahren gewählt werden und der PQG Parameter wird nur benötigt, um Einstellungen für DSA vorzunehmen. Wird kein Challenge String unterstützt, dann wird ein IA5STRING der Länge Null übergeben. Im einfachsten Fall sieht eine Anforderung also folgendermaßen aus :

```
<KEYGEN NAME="name">
```

Zu beachten ist, daß in den Netscape Exportvarianten nur Schlüssel mit einer Länge von 512 Bit generiert werden, während die USA-Version die sonst übliche Länge von 1024 Bit unterstützt.

Alle weiteren Angaben sind vom Nutzer einzugeben und werden zusammen mit dem vom Browser generierten öffentlichen Schlüssel an ein Programm auf dem Server übergeben. Bei diesem Programm handelt es sich üblicherweise um ein CGI- oder Perl-Script, das die Erstellung des Zertifikates durch die Ausführung einer Reihe von SSL-Befehlen übernimmt.

Größtes Problem dabei ist es, sicherzustellen für wen das gerade generierte Zertifikat bestimmt ist, denn die Kommunikation zum Erhalt des Zertifikates läuft zwar über https aber eventuell ohne Client Authentifikation, weil dessen Zertifikat gerade erst generiert wird.

Verisign [15], eine amerikanische CA, bietet potentiellen Kunden ein Testzertifikat mit einer Gültigkeitsdauer von sechs Wochen an. Die CA prüft dabei nur, ob der Nutzer in der Lage ist, die Post an eine von ihm anzugebende E-Mail Adresse zu lesen. Es wird nach dem Antrag im WWW eine Art Passwort per Mail verschickt und wenn das wieder korrekt auf der Webseite von Verisign eingegeben wird, erhält der Nutzer sein Zertifikat.

Diese Vorgehensweise ist von vielen Seiten kritisiert wurden [7]. Diese Zertifikate enthalten zwar den Vermerk, daß sich der Nutzer lediglich mit einer E-Mail Adresse ausgewiesen hat, aber das in Umlauf Bringen unsicherer Zertifikate schadet dem Einsatz eines ansonsten sehr nützlichen Verfahrens doch mehr, als es den wenigen Nutzern innerhalb der Testphase nützt.

Das Generieren von Zertifikaten aus dem Browser heraus sollte also nur eingesetzt werden, wenn der Nutzer sich gegenüber der ausstellenden Instanz so sicher authentifizieren kann, wie das später mit seinem neuen Zertifikat möglich wird. Ein mögliches Einsatzgebiet ist das aktualisieren von Zertifikaten kurz vor ihrem Verfallsdatum. Hier werden zwar meist die Schlüssel beibehalten, es kann sich aber durchaus erforderlich machen, deren Länge zu erhöhen und deshalb neue zu wählen.

8.2.2 Zertifikate importieren

Eine unter dem Gesichtspunkt der Authentifikation praktikablere Lösung für das Erstellen von Zertifikaten ist das Zertifizieren bei der CA mit anschließendem Import einer Zertifikatsdatei in den Browser.

Da Zertifikate eine Grundlage für eine sichere Kommunikation oder Authentifikation in Netzwerken bilden ist es nicht verwunderlich, daß bei der Authentifikation zur Erlangung eines Zertifikates andere, konventionellere Wege gegangen werden müssen. Viele CA's schreiben daher in ihrer Policy, dem Regelwerk das ihrem Handeln zugrundeliegt, vor, daß die Identität eines Nutzers durch sein persönliches Erscheinen und Vorlegen amtlicher Dokumente zu prüfen ist. Da CA's einen großen Einzugsbereich haben können, wird teilweise auf vertrauenswürdige Registrierungsinstanzen zurückgegriffen, die lokal die Identität und Authentizität von Nutzern prüfen können. Als Beispiel hierfür sei die Policy des Individual Network e.V. aufgeführt [8]. Diese Policy liegt auch der CA der Firma IKS GmbH Jena zugrunde und ermöglichte es mir mich hier in Chemnitz auszuweisen und dadurch die Fahrt nach Jena zu vermeiden.

Kapitel 9

Eigene CA erstellen

9.1 OpenSSL

Nachdem die CA den Nutzer überprüft hat, kann damit begonnen werden das eigentliche Zertifikat zu erstellen. Dazu ist eine SSL-Implementierung nötig, die in der Lage ist, Schlüssel für ein asymmetrisches Kryptosystem zu generieren, Nutzerzertifikate zu erstellen, zu verwalten und in spezielle Formate zu konvertieren. Erste Versuche mit SSLeay schlugen leider Fehl, da die von mir verwendete Version leider nicht das pkcs12 Format verstand, das für den Import in aktuelle Versionen (4.x) des Netscape Navigators nötig ist. Alle anderen Schritte liefen zwar reibungslos ab, aber was nützt das, wenn nicht der komplette benötigte Funktionsumfang der SSL Version 3 implementiert ist. Ich mußte mich also nach einem alternativen Programmpaket umsehen und fand es in OpenSSL, das seit August in der Version 0.9.4 vorliegt.

Weil das Clientzertifikat von einer CA ausgestellt werden muß und hier gezeigt werden soll wie das geschieht, brauchen wir eine CA. Das OpenSSL Paket enthält eine Reihe von Skripten, um für den Administrator absehbare Routinetätigkeiten zu vereinfachen und dadurch die Fehlerquote zu reduzieren. Der folgende Dialog zeigt, wie mit Hilfe des Skriptes CA.sh eine Certification Authority angelegt wird :

```
root@pumuckl:/usr/local/ssl/misc > /usr/local/ssl/misc/CA.sh -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Using configuration from /usr/local/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....+++++
....+++++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
```

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
—–
Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Technische
Universitaet Chemnitz
Organizational Unit Name (eg, section) []:CSN
Common Name (eg, YOUR name) []:CA pumuckl.csn.tu-chemnitz.de
Email Address []:mmr@informatik.tu-chemnitz.de
root@pumuckl:/usr/local/ssl/misc >

Die benötigten Verzeichnisse werden vom Skript angelegt, es muß also mit root Rechten gestartet werden, um in /usr/local/... schreiben zu dürfen. Bedauerlich ist, daß die Zugriffsrechte für private Dateien nicht automatisch gesetzt werden, dadurch ist erst einmal jeder mit einem login auf dem Computer in der Lage, die komplette CA zu kopieren. Das ist einer der Gründe, warum in den meisten CA Policy's steht, daß der Computer auf dem Zertifikate ausgestellt werden in kein Netzwerk integriert sein darf. Das Skript CA.sh stammt übrigens von Tim Hudsen, einem der Entwickler von SSLeay.

Der Inhalt des Zertifikates, der im letzten Teil des abzuarbeiteten Skriptes eingegeben wird, ist fest vorgeschrieben. Er dient bei Client- und Serverzertifikaten der Authentifizierung. Da root-CA's keine übergeordnete Instanz haben, bei der der Wortlaut geprüft werden kann und der Inhalt mit keinem Hostnamen übereinstimmen muß, kann bei einem selbstunterschriebenen Zertifikat etwas freier bei der Namensgebung verfahren werden. Nutzer am Clienten müssen nämlich manuell bestätigen, ob und in welchem Rahmen sie der jeweiligen CA vertrauen und auf dem Server müssen die Zertifikate vertrauenswürdiger CA's ohnehin in einem bestimmten Verzeichnis abgelegt werden.

9.2 Clientzertifikat generieren

Nachdem nun eine CA existiert, kann mit der Erstellung eines Zertifikates für eines Nutzer begonnen werden. Die Nutzerdaten wurden bereits überprüft, nun ist das RSA Schlüsselpaar zu generieren.

root@pumuckl:/usr/local/ssl/misc > openssl genrsa -out mmr.key 1024
1112 semi-random bytes loaded

Generating RSA private key, 1024 bit long modulus
.......+++++
....+++++
e is 65537 (0x10001)
root@pumuckl:/usr/local/ssl/misc >

Die einzelnen Schritte erfolgen direkt mit OpenSSL und sind deshalb gut nachvollziehbar. Die eben angelegte Datei mmr.key , die das Schlüsselpaar enthält wird im nächsten Schritt in eine Request-Datei, einen Antrag auf Erhalt eines Zertifikates eingebaut.

root@pumuckl:/usr/local/ssl/misc > openssl req -new -out mmr.req -key
mmr.key
Using configuration from /usr/local/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
—–
Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Technische
Universitaet Chemnitz
Organizational Unit Name (eg, section) []:CSN
Common Name (eg, YOUR name) []:Mirko Mrowczynski
Email Address []:mmr@informatik.tu-chemnitz.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@pumuckl:/usr/local/ssl/misc >

Diese Request-Datei enthält exakt die Daten, die zertifiziert werden sollen. Wird die Konfigurationsdatei /usr/local/ssl/openssl.cnf nicht verändert, wird an jedes so erzeugte Zertifikat der Kommentar "OpenSSL Generated Certificate" angehängt. Einige CA's tragen hier Informationen zur Art des vorliegenden Zertifikates ein. Verisign weist an dieser Stelle bei Testzertifikaten darauf hin, daß sich der Besitzer nur durch seine E-Mail Adresse authentifiziert hat.

In einige der auszufüllenden Felder wurde lediglich ein Punkt eingetragen. Diese Felder sind für dieses Zertifikat nicht relevant und werden im fertigen Zertifikat nicht enthalten sein. Welche Felder unbedingt ausgefüllt werden müssen und wo eine Eingabe optional ist, wird auch in openssl.cnf festgelegt. Eine Überprüfung findet jedoch erst im nächsten Schritt statt, eine Fehlermeldung enthält man also auch erst dort. Nun wird die Request-Datei der CA zur Zertifizierung vorgelegt.

```
root@pumuckl:/usr/local/ssl/misc > openssl ca -policy policy_anything -out
mmr.crt -infiles mmr.req
Using configuration from /usr/local/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'de'
organizationName      :PRINTABLE:'Technische Universitaet Chemnitz'
organizationalUnitName:PRINTABLE:'CSN'
commonName            :PRINTABLE:'Mirko Mrowczynski'
emailAddress          :IA5STRING:'mmr@informatik.tu-chemnitz.de'
Certificate is to be certified until Oct 12 18:00:42 2000 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@pumuckl:/usr/local/ssl/misc >
```

Dem Administrator wird der Inhalt des Antrags angezeigt, nachdem er das Passwort, mit dem die Request-Datei gesichert ist, eingegeben hat. Danach wird er gefragt, ob das Zertifikat unterzeichnet werden soll.

Die Sicherung per Passwort erfolgt, weil alle bis zu diesem Punkt unternommenen Schritte auch auf einem anderen Computer als dem der CA ausgeführt werden können. In vielen Fällen müssen sie sogar anderswo ausgeführt werden, da die CA sonst den privaten Schlüssel, der zum Zertifikat gehört kennt. Das Passwort soll also die zu übertragende Request-Datei auf ihrem Weg zur CA schützen.

Wird bei der Zertifikatserstellung festgestellt, daß die Daten der Request-Datei unvollständig sind, bricht OpenSSL mit einem Hinweis auf die fehlende Information ab.

```
root@pumuckl:/usr/local/ssl/misc > openssl pkcs12 -export -in mmr.crt -inkey
mmr.key -certfile demoCA/cacert.pem -name "Mirko" -out mmr.p12
```

Enter Export Password:
Verifying password - Enter Export Password:
root@pumuckl:/usr/local/ssl/misc >

Der Antragsteller sollte sich also gut informieren, welche Informationen benötigt werden, da durch einen Fehler oder Unvollständigkeit nochmaliges persönliches Erscheinen notwendig werden kann.

Wenn der Administrator also seinen Segen erteilt hat, legt die CA eine .crt Datei an. In diese Datei werden die mit dem privaten Schlüssel der CA signierten Nutzerdaten geschrieben. Diese Datei enthält also das, was später vom Clienten zur Authentifikation an den Server übertragen wird.

Um jedoch ein komplettes Client-Zertifikat zu bekommen, das aus einer Datei in den Browser geladen wird muß in die .crt Datei noch der private Schlüssel des Clienten integriert werden. Das sollte dort erfolgen, wo das RSA Schlüsselpaar erzeugt wurde. Wenn der Nutzer mit der Zertifikatsanforderung zur CA gegangen ist, sollte er die .crt Datei mit dem Zertifikat mit zu seinen Computer nehmen und den letzten Schritt vor dem importieren in den Browser dort erledigen.

root@pumuckl:/usr/local/ssl/misc > openssl pkcs12 -export -in mmr.crt -inkey mmr.key -certfile demoCA/cacert.pem -name "Mirko" -out mmr.p12
Enter Export Password:
Verifying password - Enter Export Password:
root@pumuckl:/usr/local/ssl/misc >

Bei den drei hierfür benötigten Dateien handelt es sich um das Clientzertifikat, die Datei mit dem Schlüsselpaar des Clienten und um das Zertifikat der CA, das natürlich frei zugänglich ist. Aus diesen Daten wird dann die .p12 Datei generiert, die in den Browser geladen wird.

Im Netscape Navigator können solche Zertifikate unter dem Menüpunkt Security, Certificates, Yours geladen werden. Wenn die Datenbank noch keine Zertifikate enthält, muß nach dem Laden ein Passwort zur Sicherung eingegeben werden. Bei jeder Änderung der Datenbank und dem ersten Zugriff nach starten des Browsers wird dieses dann geprüft. Also Vorsicht, jeder an der Konsole kann die Zertifikate nutzen, wenn eins in dieser Sitzung bereits verwendet wurde.

9.3 Pro und Kontra eigene CA

Ich habe mich dafür entschieden eine eigene CA aufzubauen, um Server und Client mit Zertifikaten zu versorgen. Eine andere Möglichkeit wäre es gewesen, die Zertifikate bei bestehenden, anerkannten CA's anzufordern.

Für HTTP-Server innerhalb der TU-Chemnitz ist das kein Problem, da das URZ eine eingene CA betreibt [10]. Das hätte auch den Vorteil, daß man über Web-Seiten des Universitätsrechenzentrums das Zertifikat dieser CA in den Browser laden kann und der dann in der Lage, ist die

Authentizität von HTTP-Servern mit einem Zertifikat dieser CA zu prüfen.

Etwas schwieriger gestaltet sich die Sache bei den Clientzertifikaten. Es ist natürlich kein Problem für potentielle Nutzer des Servers, sich bei einer kommerziellen CA (Verisign, Thawte [12], Easterngraphics GmbH) ein Zertifikat zu besorgen. Der Server kann aber bei der Authentifikation nur eine Gruppenzugehörigkeit prüfen, müßte also erst einmal jedem Besitzer eines solchen Zertifikates den Zugang gewähren. Eine Möglichkeit dann bestimmte Verzeichnisse freizuschalten, könnte über die Auswertung von Logfiles mit anschließender Änderung von .htaccess Files erfolgen. Der Apache ist so vorkonfiguriert, daß er den Inhalt von Clientzertifikaten zusammen mit Datum und Uhrzeit des Zugriffs und der IP Adresse des Clienten in der Datei /var/log/ssl_log abspeichert. Diese Variante bietet jedoch eine ganze Reihe von Angriffsmöglichkeiten, so das nicht annähernd die Sicherheit bietet, die mit Zertifikaten von nur einer CA erreicht wird.

Langfristig wird aber daran weitergearbeitet werden müssen, Nutzerzertifikate unterschiedlichster CA's zu akzeptieren, da sonst jeder Nutzer eine Reihe von Zertifikaten verwalten muß. Ähnlich wie bei Schlüsseln oder Chipkarten kann der Besitzer einem Zertifikat dann nicht mehr die volle Aufmerksamkeit schenken, was einen Mißbrauch erleichtert. Peinliche Situationen können bereits entstehen, wenn aus Versehen das falsche Zertifikat an den Server geschickt wird. Das ist dann etwa so, als fällt einem beim Bezahlen die Rabattkarte des Konkurrenzunternehmens aus der Tasche.

Kapitel 10

Server Einrichten

10.1 Installation

Beim Einrichten eines HTTP-Servers habe ich mich an das gehalten, was mir mit der SuSE 6.0 in die Hand gegeben wurde. Der bekannteste und wohl auch ausgereifteste HTTP-Server unter Linux, der natürlich auch auf der SuSE Distribution [13] nicht fehlen darf, ist der Apache. Der Apache, bei mir in der Version 1.3 im Einsatz, zeichnet sich durch eine Reihe von Eigenschaften aus, die ihm zu einer Vormachtstellung unter Web-Servern verholfen hat und ihn Dank dieser Eigenschaften auch für meine Zwecke geeignet macht.

Größter Pluspunkt bei einem kommerziellen Einsatz ist sicher, daß der Apache bei laufendem Betrieb gewartet werden kann. So ist es möglich, die Konfiguration zu ändern während Verbindungen zum Server bestehen und dadurch eine nahezu ununterbrochene Verfügbarkeit zu gewährleisten. Für meine Aufgabe steht jedoch der modulare Aufbau des HTTP-Servers im Vordergrund. Diese Modularität ermöglicht es in Abhängigkeit vom Verwendungszweck, die Funktionalität des Servers zu ändern.

Eines dieser Module ist mod_ssl. Mit seiner Hilfe wird der Apache HTTP-Server SSL-fähig. Da unabhängig von den gezeigten Serverinhalten etwas gegen mögliche Kommunikationsbedrohungen getan werden sollte, ist der Einsatz von SSL-fähigen Servern dringend zu empfehlen. Bei SuSE wird aus diesem Grund neben dem normalen Apache, der natürlich auch durch Module ergänzt werden kann, ein bereits vorkonfigurierter SSL-fähiger Apache angeboten.

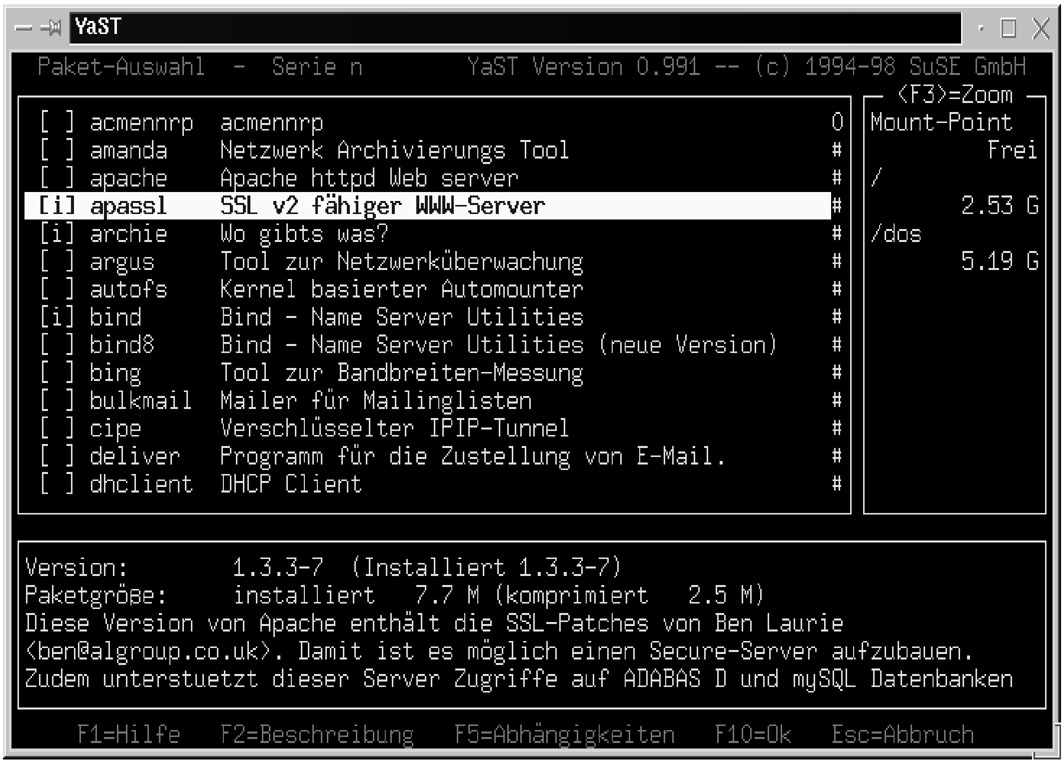


Abbildung 10.1: vorkonfigurierte Apache Webserver

Nach der Installation des apassl Paketes kann sowohl über Port 80, den HTTP-Port als auch über Port 443, für Secure HTTP, eine Verbindung zum Server aufgebaut werden. Das Wurzelverzeichnis für Dokumente ist in beiden Fällen /usr/local/httpd/htdocs/ . Das kann aber für Bereiche in denen die Verwendung von HTTPS zwingend erforderlich ist so umkonfiguriert werden, daß ein Besucher über HTTP auf Port 80 die Meldung bekommt, daß es sich um einen geschützten Server handelt und die Verwendung von HTTPS mit anschließender Authentifikation per Zertifikat erforderlich ist. Diejenigen, deren Authentizität geprüft wurde, erhalten dann Zugriff auf das Wurzelverzeichnis von Port 443.

10.2 Konfiguration

Unter Beibehaltung der Standardeinstellungen erfolgt keine Überprüfung des Clienten. Es wird davon ausgegangen, daß Informationen öffentlich zur Verfügung gestellt werden. Um nun eine Authentifizierung zu erzwingen, muß die Konfiguration des Servers geändert werden. Alle damit in Zusammenhang stehenden Dateien befinden sich in /etc/httpd/ . Der HTTP-Server lädt beim Start oder einem Konfigurationsupdate die Dateien access.conf, httpd.conf und srm.conf . In welcher Datei welche Informationen enthalten sind spielt für den Server keine Rolle. Die Aufteilung erfolgt, um das Auffinden und Verändern bestimmter Einstellungen zu erleichtern.

Was als Segen gedacht war kann aber leicht zum Fluch werden. Die Möglichkeit Ergänzungen an beliebiger Stelle vornehmen zu können, sollte innerhalb der Konfigurationsdateien mit Vorsicht behandelt werden, da durch solche Ergänzungen eventuell vorher getroffene Einstellungen überlesen werden oder die Ergänzungen selbst durch spätere Anweisungen ihre Gültigkeit verlieren. Vorteilhaft bei dieser Art des Überlesens ist jedoch, daß grundsätzlich erst einmal Alles verboten werden kann und anschließend nur gewünschte Sachen freigeschaltet werden. Dadurch können Sicherheitslücken vermieden werden ohne das ein konkreter Verdacht besteht.

Der Apache HTTP-Server verfügt über vier Varianten der Clientauthentifikation per SSL. Ausgewählt wird eine der Möglichkeiten über die Variable SSLVerifyClient in der Konfigurationsdatei httpd.conf :

```
#  set client verification level: [RECOMMENDED]
#  0|none:            no certificate is required
#  1|optional:        the client may  present a valid certificate
#  2|require:         the client must present a valid certificate
#  3|optional_no_ca:  the client may  present a valid certificate
#                     but it is not required to have a valid CA
SSLVerifyClient 2
```

Voreingestellt ist das System nach der Installation auf none. Es erfolgt keine Überprüfung des Clienten, alle Informationen sind frei zugänglich. Für die Varianten bei denen der Client ein Zertifikat vorzeigen kann, aber nicht muß, konnte ich keinen Nutzen feststellen. Mit der Aussage des Servers "wenn du mir unbedingt sagen möchtest wer du bist dann tue es, aber ich möchte es eigentlich nicht wissen" läßt sich die Sicherheit des Systems einfach nicht verbessern. Die Möglichkeiten 1 und 3 haben für meine Arbeit keinerlei Bedeutung. Die einzige Variante den Server zu zwingen, die Authentizität des Clienten zu prüfen, ist es SSLVerifyClient auf zwei oder require zu setzen. Eine Beschreibung der einzelnen Direktiven und Hinweise zu Zusatzmodulen des Apache sind unter [14] zu finden.

Bei dieser Einstellung wird nach der Kontrolle, ob das Zertifikat zeitlich gültig ist und der Client den zum Zertifikat gehörenden privaten Schlüssel besitzt auch überprüft, ob der ausstellenden CA vertraut wird. Für den Server gibt es an dieser Stelle nur zwei Varianten. Ihm ist das Zertifikat der CA bekannt und er vertraut ihr, oder er kennt es nicht und lehnt den Clienten ab. Da der Server mit Clientzertifikaten unterschiedlichster CA's umgehen muß, gibt es im Konfigurationsbereich des HTTP-Servers ein Verzeichnis, in dem die Zertifikate aller für ihn vertrauenswürdigen CA's untergebracht sind.

Um nun den HTTP-Server zu veranlassen, den Client mit dem von uns ausgestellten Zertifikat zu akzeptieren, muß das Zertifikat unserer eigenen CA in den SSLCACertificatePath des Servers kopiert werden. Der Name, den die auf dem Server abgelegte Datei hat, spielt keine Rolle. Er sollte nur aus Gründen Der Übersichtlichkeit auf .crt enden. Der Server ermittelt den benötigten Dateinamen über eine Hashfunktion aus den vom Clienten übermittelten Informationen über die ausstellende CA. Um dem Server den Zugriff zu ermöglichen, wird ein Link mit dem über die Hashfunktion ermittelten Namen angelegt, der auf die Zertifikatsdatei verweist. Die Hashfunktion ist in OpenSSL implementiert und ihr Resultat wird bei der Erzeugung des

Links als Dateiname verwendet.

```
root@pumuckl:/etc/httpd/ssl.crt > ln -s demoCA.crt 'openssl x509 -noout
-hash -in demoCA.crt'.0
root@pumuckl:/etc/httpd/ssl.crt > ll
total 14
-rw-r--r--   1 root   root    887 Dec 12  1998 Makefile
-rw-r--r--   1 root   root   1385 Dec 12  1998 README.CRT
lrwxrwxrwx   1 root   root     10 Oct 13 19:32 c21a8722.0 -> demoCA.crt
-rw-r--r--   1 root   root   1334 Oct 13 19:00 demoCA.crt
-rw-r--r--   1 root   root   4022 Oct 13 19:10 server.crt
-rwxr--r--   1 root   root     40 Sep 24 11:04 view
root@pumuckl:/etc/httpd/ssl.crt >
```

Da es bei der Anwendung von Hashfunktionen vorkommen kann, daß unterschiedliche Zertifikate den selben Hashwert liefern, wird bei diesen Dateien die Dateinamenserweiterung durchnumeriert. Alle Dateien bekommen aus diesem Grund die Kennung .0 , alle doppelt vorkommenden dann .1 und so fort.

10.3 Serverzertifikate

Jetzt wo der Server der CA vertraut und damit die Clientauthentifikation gewährleistet ist, benötigt der HTTP-Server natürlich selbst auch ein Zertifikat. Seine Authentifikation gegenüber dem Clienten ist schließlich sein erster Schritt beim Verbindungsaufbau. Zum erstellen des Zertifikates wird die CA verwendet, mit der auch das Clientzertifikat generiert wurde. Das muß natürlich nicht sein, aber wenn Server- und Clientzertifikat von der selben CA erstellt wurden, sollte es dem Nutzer am Clienten leichter fallen auf "akzeptieren" zu klicken.

Als erstes muß wieder ein RSA Schlüsselpaar generiert werden [9]. Das kann auf einem anderen Rechner erfolgen, als dem der CA. Dann muß jedoch auch die Zertifikatsanforderung dort erstellt werden.

```
root@pumuckl:/usr/local/ssl/misc > openssl genrsa -out server.key 1024
1112 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.....+++++
.+++++
e is 65537 (0x10001)
root@pumuckl:/usr/local/ssl/misc >

root@pumuckl:/usr/local/ssl/misc > openssl req -new -out server.req -key
server.key
```

Using configuration from /usr/local/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
—–
Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Technische
Universitaet Chemnitz
Organizational Unit Name (eg, section) []:CSN
Common Name (eg, YOUR name) []:pumuckl.csn.tu-chemnitz.de
Email Address []:mmr@informatik.tu-chemnitz.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@pumuckl:/usr/local/ssl/misc >

root@pumuckl:/usr/local/ssl/misc > openssl genrsa -out server.key 1024
1112 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.....+++++
.+++++
e is 65537 (0x10001)
root@pumuckl:/usr/local/ssl/misc >

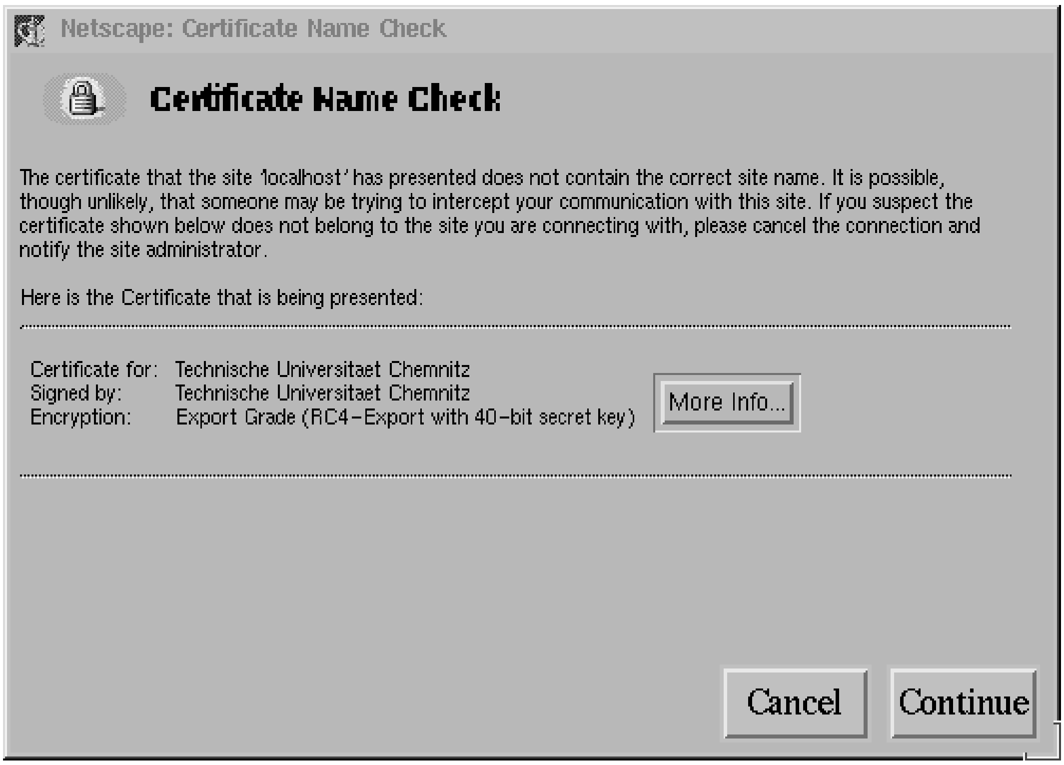
root@pumuckl:/usr/local/ssl/misc > openssl req -new -out server.req -key
server.key
Using configuration from /usr/local/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

—–

Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Technische Universitaet Chemnitz
Organizational Unit Name (eg, section) []:CSN
Common Name (eg, YOUR name) []:pumuckl.csn.tu-chemnitz.de
Email Address []:mmr@informatik.tu-chemnitz.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@pumuckl:/usr/local/ssl/misc >

Bei "Common Name" muß exakt der Name des HTTP-Servers eingegeben werden, wie er sonst in der URL steht. Wenn ich hier also pumuckl.csn.tu-chemnitz.de schreibe, dann lautet die URL https://pumuckl.csn.tu-chemnitz.de . Eine Fehleingabe würde letztendlich zu einem Zertifikat für einen anderen Server führen. Der Netscape Navigator gibt eine Warnung aus, wenn der Server sich mit einem Zertifikat authentifizieren möchte und die Namen nicht übereinstimmen. Eine solche Warnung wird auch ausgegeben, wenn ein Nutzer am pumuckl.csn einfach nur https://localhost eingibt.



Also Vorsicht mit Aliases. Der Browser kann nur den Namen prüfen, nicht aber die IP.

Der hoffentlich richtig ausgefüllte Antrag für ein Zertifikat wird dann der CA zum signieren vorgelegt. Der Administrator prüft dann den Inhalt und OpenSSL mittels Konfigurationsdatei die Vollständigkeit, bevor beide ihr ok geben.

```
root@pumuckl:/usr/local/ssl/misc > openssl ca -policy policy_anything -out
server.crt -infiles server.req
Using configuration from /usr/local/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'de'
organizationName      :PRINTABLE:'Technische Universitaet Chemnitz'
organizationalUnitName:PRINTABLE:'CSN'
commonName            :PRINTABLE:'pumuckl.csn.tu-chemnitz.de'
emailAddress          :IA5STRING:'mmr@informatik.tu-chemnitz.de'
Certificate is to be certified until Oct 12 17:10:14 2000 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@pumuckl:/usr/local/ssl/misc >
```

```
root@pumuckl:/usr/local/ssl/misc > openssl ca -policy policy_anything -out
server.crt -infiles server.req
Using configuration from /usr/local/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'de'
organizationName      :PRINTABLE:'Technische Universitaet Chemnitz'
organizationalUnitName:PRINTABLE:'CSN'
commonName            :PRINTABLE:'pumuckl.csn.tu-chemnitz.de'
emailAddress          :IA5STRING:'mmr@informatik.tu-chemnitz.de'
```

Abbildung 10.2: Kein Zertifikat für localhost

Certificate is to be certified until Oct 12 17:10:14 2000 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@pumuckl:/usr/local/ssl/misc >

Dieses fertige Zertifikat muß nun zusammen mit der auf dem Server verbliebenen Schlüsseldatei in die Serverkonfiguration integriert werden. Die Datei server.key mit den RSA Schlüsseln wird nach /etc/httpd/ssl.key/ kopiert. Dieses Verzeichnis ist nur für Schlüssel des Apache HTTP-Servers vorgesehen und sollte, da dieser mit root-Rechten gestartet wird auch nur mit diesen Rechten lesbar und ausführbar sein.

Das Zertifikat kommt zu den anderen, also nach /etc/httpd/ssl.crt/ , wo bereits das Zertifikat unserer CA auf seinen Einsatz wartet. Links müssen diesmal nicht erzeugt werden, die Dateinamen werden über Direktiven in den drei Konfigurationsdateien festgelegt.

Jetzt muß der HTTP-Server nur noch einmal neu gestartet werden und zwischen dem Server und dem Clienten kann eine webbasierte Authentifikation mittels Zertifikaten erfolgen.

Kapitel 11

Ausblick

Die Authentifizierung wird an Bedeutung gewinnen, daß gilt besonders für die Authentifizierung gegenüber einer Maschine. Das liegt daran, daß die Anzahl der informationsverarbeitenden Maschinen steigt und keine Mechanismen zum Erinnern an vorangegangene Prüfungen der Authentizität eingesetzt werden. Bei einem Notar können nach erfolgter Identifikation mehrere voneinander unabhängige Sachverhalte geklärt werden, weil sich der Notar die Daten zur Identifikation merkt und mit diesen Daten bei jedem neuen Sachverhalt die Authentizität prüft. Er klärt also, ob mit der betreffenden Person über den Sachverhalt gesprochen werden darf. Maschinen dagegen nehmen bei einer erforderlichen Prüfung der Authentizität auch eine Prüfung der Identität vor. Typischer Vertreter dieser Philosophie ist der Bankomat. Implementierungen mögen sich zwischen einzelnen Kreditinstituten unterscheiden. Nach jeder am Automaten durchgeführten Transaktion die Karte und die PIN zu verlangen, ist aber gängige Praxis.

Ziel künftiger Realisierungen bezüglich der Authentifikation sollte es daher sein, etwas mehr von der Zurückhaltung walten zu lassen, die man von menschlichen Kommunikationspartnern gewohnt ist. Das bedeutet Prüfung der Authentizität, ohne das es einem vorkommt, als würde einem ständig ein Betrugsversuch vorgeworfen.

Bei der Mensch-Maschine Kommunikation läßt sich eine solche sanftere Authentifikation durch eine kontinuierliche Prüfung der Identität erreichen. Das ist jedoch mit einer Einschränkung der Identifikationsmerkmale verbunden. Besitz kann verlegt und vom Finder mißbraucht werden und Wissen kontinuierlich zu prüfen geht nicht, weil darunter der eigentliche Zweck der Kommunikation leidet. Bleibt also die Identifikation per biometrischer Eigenschaft. Im Bezug auf Sicherheit stellen biometrische Eigenschaften kein Problem dar. Schwierigkeiten treten erst im Zusammenhang mit den benötigten Geräten auf, da beim Feststellen der Eigenschaft hohe Ansprüche an die Qualität der Erkennung gestellt werden. Wenn aber irgendwann die Maschine den Nutzer kontinuierlich per Kamera und Mikrofon prüft und diese Geräte eventuell auch noch zur Eingabe bei der Kommunikation verwendet werden, steigt nicht nur der Service, sondern es wird auch etwas zum Schutz der Integrität getan, ohne das der Nutzer einen Vertrauensbruch spürt.

Das eben beschriebene Szenario greift weit in die Zukunft voraus. Das WWW wird zu diesem Zeitpunkt durch Softwareengineering und höheren Datendurchsatz eine andere Form haben als heute. Die nächsten Schritte auf dem Weg in eine schöne neue Welt werden daher erst einmal mit der gesellschaftlichen Etablierung wissenschaftlicher Erkenntnisse zu tun haben. Vorangetrieben wird in unserer Gesellschaft eine Etablierung, egal welcher Art, am ehesten durch wirtschaftliche Überlegungen. So ist es nicht verwunderlich, daß es innerhalb der Europäischen Gemeinschaft Bestrebungen zur Anerkennung digitaler Signaturen gibt. Eine Gleichsetzung solcher Signaturen mit Unterschriften auf Papier wäre eine Absicherung für alle Arten elektronisch getätigter Geschäfte und würde die Bereitschaft per Computer einzukaufen stark erhöhen. In diesem Zusammenhang bekämen auch die CA's der Trust Center starken Zulauf, da über sie die Zugehörigkeit von Signaturen zu Personen geklärt wird.

Wenn sich die Leute an die Benutzung von Zertifikaten gewöhnt haben, wird es Bestrebungen geben, diese Zertifikate von verschiedenen Computern aus zu nutzen. Für den Computer am Arbeitsplatz und dem Daheim funktioniert das eventuell noch per Transport des Zertifikates auf Disk. Bei gelegentlicher Nutzung auf fremden Computern muß aber der private Schlüssel geschützt werden. In einer solchen Situation ist es nicht mehr möglich das Zertifikat auf einem komplett lesbaren Speichermedium zu transportieren. Abhilfe könnten hier PDA's (Persönliche Digitale Assistenten) schaffen, die jeder bei sich trägt und bei Bedarf mit dem Computer, an dem er arbeitet, verbindet. Diese PDA's würden dann das Zertifikat des Nutzers enthalten und die komplette, mit asymmetrischen Verfahren durchzuführende, Verschlüsselung übernehmen. Eine Prozessorkarte, die nur den Bruchteil des Geldes für einen PDA kostet, kann diese Aufgabe leider nicht übernehmen, da sie keine direkte Ein- Ausgabe zum Nutzer hat. Der PDA sollte nämlich bei jeder Anforderung einer Unterschrift seinen Nutzer fragen, ob wirklich unterschrieben werden soll. Bei einer Prozessorkarte könnte der Computer, an den diese angeschlossen ist, alles Mögliche unterschreiben lassen, ohne das der Nutzer davon erfährt.

Abkürzungsverzeichnis

ANSI	American National Standards Institute
CA	Certification Authority
CGI	Common Gateway Interface
DES	Data Encryption Standard
EC	Euro Cheque
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
ID	Identifier
IP	Internet Protocol
NP	Nichtdeterministisch Polynomial
NSA	National Security Agency
PC	Personal Computer
PDA	Persönlicher Digitaler Assistent
PGP	Pretty Good Privacy
PIN	Personal Ident Number
RSA	Rivest Shamir Adleman
SCP	Secure Copy
SSH	Secure Shell
SSL	Secure Socket Layer
SuSE (GmbH)	(Gesellschaft für) Software- und System-Entwicklung
URL	Uniform Resource Locator
VDE	Verband deutscher Elektrotechniker
WWW	World Wide Web

Literaturverzeichnis

[1] http://de.news.yahoo.com/991129/11/e54s.html

[2] Oppliger, R. (1992): Computersicherheit : eine Einführung. Viehweg, Braunschweig, Wiesbaden

[3] http://www.rsa.com/rsalabs/des3/

[4] http://www.research.att.com/~smb/nsam-160/

[5] http://developer.netscape.com/docs/manuals/security/sslin/contents.htm

[6] http://home.netscape.com/eng/security/comm4-keygen.html

[7] http://archiv.tu-chemnitz.de/pub/1997/0032/certification.htm

[8] http://archiv.tu-chemnitz.de/pub/1997/0043/vortrag/sontag/policy.html

[9] http://www.tu-chemnitz.de/urz/ca/howto.html

[10] http://www.tu-chemnitz.de/urz/ca/

[11] http://iuk.tu-chemnitz.de

[12] http://www.thawte.com/about.html

[13] http://www.suse.de

[14] http://www.apache.org/docs/mod/core.html

[15] http://www.verisign.com/client/index.html

[16] Stöckel,R. (1999): Interaktive 3D-Visualisierung von Netzwerkstrukturen. Technische Universität Chemnitz

[17] Gleißner W., Grimm R., Herda S., Isselhorst H. (1989): Manipulation in Rechnern und Netzen. Addison-Wesley, Bonn, München

Selbständigkeitserklärung

Ich erkläre, daß ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Chemnitz, den 09. Dezember 1999