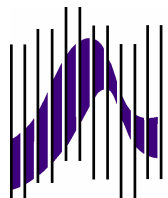


Diplomarbeit

Aufbau einer Public Key-Infrastruktur in
einem mittelständischen Unternehmen

von
Daniel Hirscher



Fachhochschule
Ravensburg-Weingarten

Sonderausgabe für das Internet

17. Februar 2002

Inhaltsverzeichnis

Einleitung	xi
1 Sicherheitsaspekte	1
1.1 Sicherheit der Software	1
1.1.1 E-Mail Dienst	1
1.1.2 Serverzugriffe auf das Extranet	3
1.1.3 Netzwerk	3
1.1.4 Daten auf Festplatten	3
1.1.5 Systemadministration über Browser	4
1.2 Physikalische Sicherheit	4
1.2.1 Hardware	4
1.2.2 Zutritt zum Gebäude	5
1.3 Angriffe	5
1.4 Zusammenfassung	6
2 Übersicht und Vergleich von Public Key Systemen	7
2.1 Vorstellung der Programme	7
2.1.1 Pretty Good Privacy (PGP)	8
2.1.2 Gnu Privacy Guard (GnuPG)	9
2.1.3 Secure Multipurpose Internet Mail Extension (S/MIME)	9
2.1.4 Weitere Systeme	9
2.2 Funktionsumfang	10
2.3 Vergleichstests	12

2.3.1	Verschlüsseln mit RSA Schlüsseln	13
2.3.2	Unterschreiben mit RSA Schlüsseln	13
2.3.3	Verwendung von DSS/DH Schlüsseln	14
2.3.4	Kompatibilität von PGP mit GnuPG	14
2.3.5	Unterschiede zwischen PGP und S/MIME	16
2.4	Zusammenfassung	18
3	Aufbau der Public Key Infrastruktur	21
3.1	Was ist eine Public Key Infrastruktur?	21
3.2	Mitarbeiterumfrage	22
3.2.1	Auswahl der Fragen	22
3.2.2	Ergebnisse	24
3.3	Einbindung von Public Key Verfahren in vorhandene E-Mail Programme	25
3.3.1	Programme	25
3.3.2	Zusammenfassung	28
3.4	Wahl des Public Key Verfahrens	28
3.4.1	Zahlen und Statistiken	28
3.4.2	Alternativen	29
3.4.3	Entscheidung für S/MIME	29
3.4.4	Weitere Möglichkeiten	30
3.5	Aufbau der Certificate Authority	31
3.5.1	Zertifikattypen	31
3.5.2	Schlüsselerzeugung beim Benutzer	32
3.6	Schulung der Mitarbeiter	33
3.7	Durchführung	34
4	Weitere Einsatzmöglichkeiten von Zertifikaten	39
4.1	Server mit SSL sichern	39
4.1.1	Extranet	39
4.1.2	Intranet	40

4.2	Unterschreiben von Programmen	40
4.2.1	Netscape's Signtool	41
4.2.2	JDK's Keytool	41
4.2.3	Microsoft's Signcode	42
4.3	Virtuelles Privates Netz (VPN)	42
5	Zusammenfassung und Ausblick	43
A	Zeitleiste	45
B	Technische Anleitung zum Aufbau der CA	47
B.1	Installation	47
B.1.1	OpenSSL	47
B.1.2	pyCA	47
B.2	Struktur der CA	50
B.2.1	PKI Definitionen	50
B.2.2	Zertifikatspeicherung	52
B.2.3	Zertifizierungsablauf	53
B.2.4	Benutzerzertifikate unterschreiben	56
B.2.5	Zertifikat zurückrufen	57
B.2.6	Konfigurationsdateien	58
B.3	Probleme	64
B.4	SSL Server Zertifikat auf Extranet Server einrichten	68
B.4.1	SSL Server Zertifikat generieren	68
B.4.2	Serverkonfiguration	69
B.5	Object Zertifikat zum Unterschreiben von Code	71
C	Protokolle und Verfahren	73
C.1	Symmetrische Verschlüsselungsverfahren	73
C.2	Asymmetrische Verschlüsselungsverfahren	74
C.2.1	RSA	75
C.2.2	El Gamal	76

C.3	Einwegfunktionen	76
C.4	Schlüsseltausch	77
C.4.1	Diffie-Hellmann	77
C.5	Zufallszahlen	78
C.6	X.509 Zertifikat	79
C.6.1	Versionen	79
C.6.2	Beispielzertifikat	79
C.6.3	PKCS	81
C.7	Verfahren, die X.509 Zertifikate benutzen	82
C.7.1	SSL	82
C.7.2	S/MIME	83
C.7.3	IPsec	83

Abbildungsverzeichnis

2.1	PGP: Web-of-Trust.	17
2.2	S/MIME: Certificate Chain	17
3.1	Fragebogen im Intranet.	23
3.2	E-Mail Programme (alle Mitarbeiter).	25
3.3	E-Mail Programme (Mitarbeiter mit Kundenkontakt).	25
3.4	Zertifizierungshierarchie.	31
3.5	Übersicht der Zertifizierungsstellen.	34
3.6	Eingabemaske für den Zertifikatsantrag.	35
3.7	Eingabemaske für Suchanfrage.	36
3.8	Ergebnis der Suche.	36
4.1	Einstiegsseite im Extranet.	40
B.1	Zeitlicher Ablauf der Zertifizierung.	55
C.1	Schlüsseltausch nach Diffie-Hellman.	77
C.2	Ablauf des SSL Handshake	83
C.3	IPsec Betriebsmodi	84

Tabellenverzeichnis

2.1	PGP: Die wichtigsten Funktionen im Überblick.	11
2.2	PGP: Protokolle und Verfahren	12
3.1	Public Key Verfahren in E-Mail Programmen	26
B.1	LDAP: Feste Attribute für Innovations	51
B.2	Definition Zertifikate	51
C.1	Kodierungen von Zertifikaten	81
C.2	S/MIME Typen	84

Einleitung

Fehlende Datensicherheit stellt heutzutage ein nicht zu unterschätzendes Risiko dar. Unternehmen sollten darauf Wert legen ihre Daten zu schützen, besonders wenn sie mit Kunden kommunizieren und ihnen vertrauensvolle Daten zukommen lassen.

Im Mittelpunkt meiner Diplomarbeit steht der Aufbau einer PKI (Public Key Infrastruktur). Die Realisierung von sicherer E-Mail (Elektronische Post), sicheren Verbindungen zu Web-Servern und das digitale Unterschreiben von Java Programmen schließt sich daran an. Die dabei verwendeten Programme, Protokolle und Verfahren werden erklärt, um zu zeigen wie sie funktionieren und was sie leisten.

Die technische Realisierung findet in der Firma Innovations GmbH statt. Das Unternehmen wurde 1997 gegründet, hat ca. 50 Mitarbeiter und spezialisiert sich auf CRM (Customer Relationship Management). Mit dem Hauptprodukt *in-contact* können regelbasierte Entscheidungsbäume visuell erstellt und automatisch ausgeführt werden.

Kapitel 1

Sicherheitsaspekte

In diesem Kapitel berichte ich von der Bestandsaufnahme verschiedener Sicherheitsaspekte in der Firma Innovations GmbH. Überprüft wurden sowohl die Sicherheit der Software bzw. Angriffe darauf, als auch die Sicherheit der Hardware, wie z.B. die Zutrittskontrolle zum Gebäude.

1.1 Sicherheit der Software

1.1.1 E-Mail Dienst

E-Mail, ob in der privaten oder geschäftlichen Kommunikation, ist heutzutage einer der wichtigsten Dienste im Internet. Leider ist der Großteil der E-Mail-Nutzer gar nicht oder nur unzureichend über die Gefahren und Missbrauchsmöglichkeiten aufgeklärt. E-Mails können auf dem Weg zum Empfänger von anderen gelesen und geändert werden, ein Absender kann eine beliebige Identität vortäuschen. Der Empfänger kann sich nicht sicher sein, ob das was er liest in der Form vom Sender geschrieben wurde und wer der Sender eigentlich wirklich ist (nach [Kon00]).

Zur Sicherung des E-Mailverkehrs eignen sich digitale Unterschriften und Verschlüsselungsverfahren. Zusätzlich wird eine Zertifizierungsinfrastruktur benötigt und es muss auf geeignete Weise eine Vertrauensbasis geschaffen werden.

Gültigkeit einer E-Mail beweisen

Beispiel aus [Ryd00]: Eines Morgens ruft ein Kunde in der Firma an. Er ist sehr verärgert über die E-Mail die er gerade erhalten hat, in der steht, wieso

und warum das Geschäftsverhältnis mit ihm gekündigt wird. Alles was er am Telefon erzählt ist für den Mitarbeiter in der Firma völlig neu. Soweit er sich erinnern kann, wurde eine derartige Nachricht von niemandem aus der Firma versendet. Wie kann er beweisen, dass diese E-Mail nicht von ihm stammt?

1. *Der Mitarbeiter unterschreibt sorgfältig jede E-Mail digital wenn er sie sendet und die vom Kunde erhaltene enthält keine Unterschrift.* Er könnte sagen: „Warum sollte ich ein Nachricht verschicken, die unser Geschäftsverhältnis zerstört ohne sie digital zu unterschreiben? Jede andere E-Mail ist unterschrieben. Ich mache das um zu zeigen, dass ich der Sender bin. Darum bin ich nicht der Sender wenn die Unterschrift fehlt!“
2. *Der Mitarbeiter unterschreibt sorgfältig jede E-Mail digital und die vom Kunde erhaltene enthält eine gültige Unterschrift.* Offensichtlich wurde der Schlüssel geklaut. Darum sollte man noch einen anderen Mechanismus einbauen, z.B. könnten Kopien aller E-Mails an die Rechtsabteilung der Firma geschickt werden (womöglich sollte sie den gleichen Mailserver verwenden, damit die E-Mail das interne Netzwerk nicht verlässt). Somit ist jede E-Mail bekannt und kann sowohl für den Sender als natürlich auch gegen ihn verwendet werden.
3. *Der Mitarbeiter sendet eine Kopie jeder E-Mail an die Rechtsabteilung und die vom Kunde Empfangene ist eine andere.* Wenn das geschieht, handelt es sich um einen Man-in-the-Middle Angriff. Um zu beweisen, dass es sich um eine Fälschung handelt, kann die Rechtsabteilung herangezogen werden.

Besondere E-Mail Adressen

Durch den inzwischen sehr dichten Namensraum der Internet Domänen kann es passieren, dass Fehladressierungen nicht erkannt werden. Durch Tippfehler in der E-Mail Adresse ist es möglich, dass ein anderer gültiger Empfänger erreicht wird. Der Server der Domain muss keine Meldung zurückschicken, so dass man den Fehler nicht sofort bemerkt. Das betrifft auch die Firma Innovations GmbH mit der Domäne `innovations.de`, denn es gibt z.B. auch `innovation.de` und `inovations.de` bzw. mit anderen Top-Level Endungen `innovations.ch`, auch `.at`, `.org`, `.com`, `.net` (sogar mit E-Mail Service) und andere.

Die universelle E-Mail-Adresse `everyone@innovations.de` könnte zu lästigen E-Mail Sendungen verwendet werden, wenn diese Adresse in falsche

Hände gerät. Abgesehen davon kann sie leicht erraten werden. Es sollte von außerhalb nicht möglich sein, E-Mails an diese Adresse zu schicken, bzw. sollten Sendungen von außerhalb sofort gelöscht werden.

Folgender Fall ist aufgetreten: Eine Person von außerhalb schickte eine E-Mail an die Adresse `everyone@innovations.de` mit der Bitte, ihm einen Zeitungsbericht zuzuschicken. Die E-Mail erreichte alle Mitarbeiter der Firma. Der Initiator hoffte nun, dass wenigstens ein Mitarbeiter unvorsichtig ist und ihm das gewünschte zuschickt. Zwei Tage später gab es eine Mitteilung, dass der Person dieser Bericht nicht zugesendet werden darf. Mit dieser Methode könnte ein Außenstehender relativ leicht an innere Daten gelangen, wenn er z.B. seine E-Mail Adresse ähnlich wählt wie ein tatsächlicher Mitarbeiter, um weniger aufzufallen.

1.1.2 Serverzugriffe auf das Extranet

Für Kunden steht ein Extranetzzugang zur Verfügung. Sie können sich dort über ein einfaches Username/Passwort System in einen Rechner in der DMZ (Demilitarized Zone) einwählen. Sie haben dann Zugriff auf alles was dort gespeichert ist. Üblicherweise werden dort Daten speziell für sie bereitgelegt.

Es wäre angebracht auf dem *extranet* Rechner mindestens eine SSL (Secure Socket Layer) Funktionalität einzurichten. Damit ist ein sicherer Zugang und eine verschlüsselte Übertragung möglich (siehe Abschnitt [4.1.1](#)).

1.1.3 Netzwerk

Firewall

Eine Firewall soll heutzutage eine Vielzahl von Aufgaben wahrnehmen, von denen der Schutz des lokalen Netzes die Wichtigste ist. Dies lässt sich mit dem Einsatz von Paketfiltern erreichen. Die Firma Innovations GmbH betreibt eine Firewall auf der Basis eines Linux Systems. Bis jetzt ist es allerdings schwierig, Einbrüche zu erkennen. Ein sogenanntes Intrusion Detection System ist noch nicht installiert.

1.1.4 Daten auf Festplatten

Durch das Anmelden (mounten) von Festplattenpartitionen eines anderen Betriebssystems hat man den vollen Lesezugriff ohne Beachtung der Zugriffsrechte der dort liegenden Daten.

Beispiel: Auf eine von z.B. Windows aus gemountete Linux Partition im `ext2` Format kann problemlos lesend zugegriffen werden. Umgekehrt, auf eine von Linux aus gemountete NTFS (NT File System) Partition kann ebenfalls problemlos lesend zugegriffen werden. Dabei spielen die ursprünglichen Dateirechte keine Rolle; alles ist sichtbar und in jedes Verzeichnis kann verzweigt werden.

Das Problem entsteht dadurch, dass es z.Z. bei der Firma Innovations GmbH erlaubt ist, beide Betriebssysteme auf einem Computer zu installieren. Die Firmenphilosophie besagt, dass Windows auf jedem Rechner installiert sein muss. Beide Betriebssysteme sind teilweise erforderlich. Windows für die Entwicklung und den Schriftverkehr in der Verwaltung und Linux um das Produkt auf einem UNIX ähnlichen System zu testen, da die Kunden verschiedenste Hardwareplattformen besitzen.

1.1.5 Systemadministration über Browser

Die Systemadministration, egal welcher Software, über einen Webbrowser, d.h. in HTML (Hypertext Markup Language) Darstellung ist unsicher, solange keine verschlüsselte Verbindung besteht, wie z.B. mit SSL. Alle übertragenen Daten können leicht abgehört werden. Das ist besonders kritisch, da die hierbei übertragenen Passwörter Administratorrechte erlauben.

1.2 Physikalische Sicherheit

1.2.1 Hardware

Computergehäuse

Die Computergehäuse sollten abschließbar sein. Es ist nicht schwierig eine Festplatte auszubauen und mitzunehmen, um diese in aller Ruhe und ohne Störung durchzusehen.

Telefon

Daten bzw. Informationen, die per Sprache über Telefon verbreitet und weitergegeben werden, können schlecht verschlüsselt werden. Jedoch handelt es sich um eine Punkt-zu-Punkt-Verbindung, bei der davon ausgegangen werden kann, dass es für Mithörer nicht ganz so einfach ist, sich einzuklinken.

1.2.2 Zutritt zum Gebäude

Die Gebäude sind mit dem elektronischen Zutrittskontrollsystem IKOTRON der Firma IKON ausgestattet. Diese Schließanlage arbeitet mit berührungslosen Codeträgern in Form eines Schlüsselanhängers. Ihnen ist eine Identifikationsnummer zugeordnet. Über einen Eintrag in einer Datenbank können die Zutrittsberechtigungen nach Tür und Zeit festgelegt werden. Der Umgang mit dem System ist sehr angenehm und einfach.

Die Identifikationsnummer eines Codeträgers ist laut Produktbeschreibung einmalig, sie kommt in keinem anderen vor. Auf eine Anfrage bei der Firma IKON, ob spezielle kryptographische Verfahren eingesetzt werden, um diese Nummer vom Codeträger zum Kontrollsystem zu übermitteln, erhielt ich keine Antwort. Ein Zwischenhändler des Systems konnte die Frage auch nicht beantworten und vermutete, dass es sich dabei um schützenswertes Know-How handelt. Das würde auf *Security-by-Obscurity* hindeuten, d.h. die Sicherheit basiert auf der Geheimhaltung des Verfahrens. Dies wurde schon im 19. Jahrhundert von A. Kerckhoff verurteilt. Da aber die Identifikationsnummer wohl teilweise auf dem Codeträger aufgedruckt ist, wäre es schlecht, wenn diese zum Schließsystem übertragen werden würde. Da dies alles ist, was ich in Erfahrung bringen konnte, sehe ich dieses System als unsicher an.

1.3 Angriffe

Das Bedrohungsmodell von [Kra98] teilt sich in unabsichtliche Bedrohungen (Höhere Gewalt, menschliches und technisches Versagen) und in absichtlich herbeigeführte Bedrohungen auf. Letzteres soll hier behandelt werden. Es teilt sich wiederum auf in Spionage und Sabotage.

Unter Spionage, bzw. passivem Abhören gibt es zwei Vorgehensweisen. Es wird entweder von außen oder von innen versucht anzugreifen. Dabei gibt es die Möglichkeiten: eine falsche Identität vorzutäuschen, unberechtigten Netzwerkzugang zu erhalten und das Anzapfen und Abhören von Übertragungsleitungen (Sniffen). Besonders für Angriffe von innerhalb gibt es die Möglichkeiten: Ausspähen von Informationen, Ausnutzung von Schwachstellen, Auswerten von Restdaten und das Zulassen eines unberechtigten Zugriffs.

Unter Sabotage, bzw. aktives Angreifen fallen die Beinträchtigung der Infrastruktur, d.h. stören, zerstören oder Diebstahl von Hardware, Übertragungsleitungen oder Versorgungseinrichtungen. Sabotage ist auch das Einbringen von zusätzlicher Information und Funktionalität (z.B. durch Viren, Trojaner)

und die Beeinträchtigung der Funktionalität von Vermittlungseinrichtungen, d.h. Änderung oder Fehlleitung von Information (z.B. DNS Spoofing), Zugriffsverhinderung (z.B. Denial of Service Angriff) und Missbrauch von Ressourcen.

1.4 Zusammenfassung

Ein Firmennetzwerk sollte Angriffen von innen als auch von außen standhalten. Die zwei wichtigsten Punkte im Zusammenhang mit dem Aufbau einer PKI sind gesicherte E-Mail und sichere Zugriffe über SSL auf Server. Dies sind in der Firma Innovations GmbH die wichtigsten Übertragungswege für vertrauliche Daten.

Kapitel 2

Übersicht und Vergleich von Public Key Systemen

In diesem Kapitel führe ich einen Vergleich von zwei PGP (Pretty Good Privacy) Programmen, von GnuPG (GNU Privacy Guard) und von S/MIME (Secure/Multipurpose Internet Mail Extensions), durch. Nach der Vorstellung der Programme werden die wichtigsten Funktionen gegenübergestellt und die enthaltenen Protokolle in einer Übersicht dargestellt. Dann folgen einige Tests, um zu unterscheiden, inwieweit die Produkte untereinander kompatibel sind.

Verglichen wird PGP-2.6.3i stellvertretend für die „alte“ Version und das aktuelle PGP-6.5.8. Die PGP Versionen ab 5.5 bis 6.5.3 enthalten eine Sicherheitslücke und wurden deshalb im Vergleich weggelassen. Bei GnuPG wurde die aktuelle Version 1.0.4 getestet. Bei S/MIME handelt es sich nicht um ein bestimmtes Programm, es ist ein Verfahren, das ähnlich funktioniert und in viele Programme eingebaut ist.

2.1 Vorstellung der Programme

Das Public Key Verfahren, d.h. die Verschlüsselung mit öffentlichem Schlüssel ist in der Kryptographie ein asymmetrisches Verfahren. Zwei zueinander gehörende Schlüssel bilden dabei ein Schlüsselpaar. Einer der Schlüssel ist der geheime, private Schlüssel, der immer beim Besitzer bleibt. Der andere, öffentliche, Schlüssel wird an die Kommunikationspartner verteilt.

Eine mit dem öffentlichen Schlüssel eines Benutzers verschlüsselte Datei kann nur mit dem dazugehörigen geheimen Schlüssel entschlüsselt werden. Für di-

gitale Unterschriften wird der eigene, geheime Schlüssel zum Unterschreiben verwendet, und jeder, der den dazugehörigen öffentlichen Schlüssel besitzt, kann die Gültigkeit überprüfen. Bei der Verschlüsselung von Daten kommen aus Geschwindigkeitsgründen symmetrische Verfahren zur Anwendung. Der dabei verwendete Schlüssel wird mit einem asymmetrischen Verfahren verschlüsselt. Man spricht dann von einem hybriden Verfahren.

2.1.1 Pretty Good Privacy (PGP)

Pretty Good Privacy ist eines der ersten Programme, welches das Public Key Verfahren realisiert. Es wurde von Phil R. Zimmermann entwickelt und zunächst als freie Software veröffentlicht. Dabei setzten die ersten Versionen das RSA (Rivest, Shamir, Adleman) ¹ Verfahren ein. Aktuelle Versionen besitzen verschiedene Verfahren zur symmetrischen und asymmetrischen Verschlüsselung. Das Programm PGP ist für viele Hardwareplattformen und Betriebssysteme erhältlich. Die kommerzielle Version wird von Network Associates, Inc. vertrieben, wobei eine eingeschränkte Version frei verfügbar ist und der Sourcecode nach einiger Zeit veröffentlicht wird. Der Nachteil von PGP ist, dass bei kommerzieller Nutzung für jeden Benutzer eine Lizenz gekauft werden muss.

ADK Fehler in PGP

Am 24. August 2000 wurde ein Fehler in PGP bekannt [Uni00], der den seit PGP-5.5 existierenden ADK (Additional Decryption Key) betrifft. Alle Versionen zwischen PGP-5.5 und PGP-6.5.3 sind davon betroffen. Der ADK-Mechanismus kann wie ein Hauptschlüssel betrachtet werden. Er wird zum Schlüsselbund der Benutzer hinzugefügt und kann dann benutzt werden, um dessen verschlüsselte Daten zusätzlich zu entschlüsseln. Dies kann z.B. der Chef einer Firma (um bei Krankheit eines Mitarbeiters dessen E-Mail zu lesen) oder die Regierung (um kriminelle Inhalte zu erkennen) fordern. Die Interessen sind unterschiedlich.

Der Fehler besteht darin, dass es möglich ist, einen nicht unterschriebenen ADK zum Schlüsselbund der Benutzer hinzuzufügen. Die betroffenen PGP-Versionen überprüfen nicht, ob sich der ADK im Hash-Block des öffentlichen Zertifikates befindet. Dadurch kann er nicht über den Fingerabdruck erkannt werden. Das Problem besteht ab der Version PGP-6.5.8 nicht mehr.

¹Beschreibung der Protokolle und Verfahren im Anhang C

GnuPG war und ist nicht betroffen, weil es ADKs nicht verwenden kann und deshalb einfach überliest. Ebenso wenig existiert diese Funktionalität in S/MIME.

Fehler bei der Prüfung der Unterschrift

Net-Security schreibt, dass es mit PGP-7.0 möglich ist, falsche Schlüssel zu importieren. Wenn *Decrypt/Verify* ausgeführt wird und die Schlüssel dabei als ASCII (American Standard Code for Information Interchange) beigefügt sind, dann werden sie beim automatischen Importieren nicht geprüft (siehe [Net01]).

2.1.2 Gnu Privacy Guard (GnuPG)

GNU Privacy Guard ist eine freie Realisierung des Public Key Verfahrens, die versucht, mit PGP kompatibel zu sein. Es verfügt wie PGP über verschiedene Verfahren zur symmetrischen und asymmetrischen Verschlüsselung. Ebenfalls ist es für eine Vielzahl von Hardwareplattformen und Betriebssystemen erhältlich. GnuPG wird ständig weiterentwickelt und enthält ab der Version 1.0.3 eine Unterstützung für RSA Schlüssel, dessen Patent am 20. September 2000 abgelaufen ist. Der Quellcode ist frei verfügbar.

2.1.3 Secure Multipurpose Internet Mail Extension (S/MIME)

Secure/Multipurpose Internet Mail Extensions ist kein separates Programm, sondern eine Erweiterung um sicher per E-Mail kommunizieren zu können. Viele E-Mail Programme können mit S/MIME umgehen, die Funktionalität ist dabei immer eingebaut. Es ist nicht kompatibel zu PGP oder GnuPG, obwohl es die gleichen Algorithmen benutzt. S/MIME kann in diesem Vergleich nur für sich allein beurteilt werden, es kann nicht mit den anderen Programmen getestet werden.

2.1.4 Weitere Systeme

PEM (Privacy Enhanced Mail), der Vorgänger von PGP und S/MIME, hat heute keine Bedeutung mehr. Es war zwar der erste Standard für verschlüsselte und digital unterschriebene E-Mail, enthielt aber einige grundsätzliche Designfehler. Somit konnte ausschließlich 7-Bit ASCII Text und Dateianhänge

nur in einem PEM eigenen Format verpackt werden. Die damaligen Verschlüsselungsverfahren DES (Data Encryption Standard) und MD5 (Message Digest 5) sind für heutige Verhältnisse zu schwach (nach [Sch98]).

MOSS (MIME Object Security Services) war ein Versuch, die Nachteile von PEM zu beheben. Es hat sich nicht durchgesetzt. Es unterstützte zwar MIME (Multipurpose Internet Mail Extensions) Formatierung, schrieb aber keinerlei Verfahren vor, wie eine E-Mail verschlüsselt oder unterschrieben werden soll, so dass eine Kompatibilität zwischen verschiedenen Implementierungen fraglich ist (nach [Sch98]).

2.2 Funktionsumfang

In Tabelle 2.1 sind die wichtigsten Kommandozeilenoptionen aufgelistet. Diese sind bei PGP-2.6.3i und PGP-6.5.8 gleich, ab PGP-5.5 sind noch Gruppenfunktionen dazugekommen. GnuPG besitzt eine Fülle von Befehlen und Optionen, die zum Teil weit über die Funktionalität von PGP hinausgehen. Dadurch können sie auch nur bei GnuPG verwendet werden. Der Funktionsumfang von S/MIME ist bei allen Implementierungen gleich und verfügt nicht über eine Kommandozeilenschnittstelle.

Tabelle 2.2 bietet einen guten Überblick der eingebauten Protokolle und Verfahren (durch • markiert). Dabei lässt sich erkennen, welches Programm sich mit welchem verständigen können müsste. Dass dies nicht immer der Fall ist, zeigen die Tests in Abschnitt 2.3. In dieser Tabelle ist auch S/MIME aufgelistet, obgleich es sich mit den anderen nicht verständigen kann.

Bei den Verschlüsselungsverfahren fällt auf, dass sich PGP-2.6.3i mit PGP-6.5.8 über IDEA (International Data Encryption Standard) verständigen kann, aber nicht mit GnuPG zusammen funktioniert. PGP-6.5.8 kann mit GnuPG über 3DES kommunizieren, dafür nicht mit PGP-2.6.3i. Bei den Schlüsseln sorgen Doppelbezeichnungen für Verwirrung: DSS/DH in PGP entspricht DSA/ELG in GnuPG. Außerdem hängt GnuPG noch eine Endung an, z.B. ELG-E oder RSA-S um die Schlüssel genauer zu kennzeichnen. Dabei bedeutet -E, dass man mit diesem Schlüssel nur verschlüsseln kann und -S, dass man damit nur unterschreiben kann.

Funktion	PGP 2.6.3i	PGP 6.5.8	GPG 1.0.4
	Basisfunktionen		
Verschlüsseln	<code>-e</code>	<code>-e</code>	<code>-e, --encrypt</code>
Unterschreiben	<code>-s</code>	<code>-s</code>	<code>-s, --sign</code>
Unterschreiben + Verschlüsseln	<code>-se</code>	<code>-es</code>	<code>-se</code>
symmetrisch Verschlüsseln	<code>-c</code>	<code>-c</code>	<code>-c, --symmetric</code>
Entschlüsseln oder Prüfen	Datei, <code>-d</code>	Datei, <code>-d</code>	Datei, <code>-d, --decrypt</code>
Klartext zerstören	<code>-w</code>	<code>-w</code>	<code>-</code>
	Schlüsselverwaltung		
Schlüsselpaar erzeugen	<code>-kg</code>	<code>-kg</code>	<code>--gen-key</code>
Schlüssel importieren	<code>-ka</code>	<code>-ka</code>	<code>--import</code>
Schlüssel exportieren	<code>-kx</code>	<code>-kx</code>	<code>--export</code>
Schlüssel anzeigen	<code>-kv</code>	<code>-kv</code>	<code>--list-keys</code>
Schlüssel anzeigen (lang)	<code>-kvv</code>	<code>-kvv</code>	<code>--list-sigs</code>
Schlüssel bearbeiten	<code>-ke</code>	<code>-ke</code>	<code>--edit-key</code>
Fingerabdruck zeigen	<code>-kvc</code>	<code>-kvc</code>	<code>--fingerprint</code>
Schlüssel unterschreiben	<code>-ks</code>	<code>-ks</code>	<code>--sign-key</code>
Unterschriften prüfen	<code>-kc</code>	<code>-kc</code>	<code>--check-sigs</code>
Schlüssel zurückziehen	<code>-kd</code>	<code>-kd</code>	<code>--gen-revoke</code>
Schlüssel entfernen	<code>-kr</code>	<code>-kr</code>	<code>--delete-key</code>
Web-of-Trust anzeigen	<code>-km</code>	<code>-km</code>	<code>-</code>
	Zusätzliche Angaben		
Ausgabe als ASCII abgetrennte Unterschrift	<code>-a</code>	<code>-a</code>	<code>-a, --armor</code>
zusätzliche Ausgaben	<code>-b</code>	<code>-b</code>	<code>-b, --detach-sign</code>
Ausgabedatei	<code>-l</code>	<code>-l</code>	<code>-v, --verbose</code>
Textmodus für Ausgabe	<code>-o</code>	<code>-o</code>	<code>-o, --output</code>
Benutzer-ID benutzen	<code>-t</code>	<code>-t</code>	<code>--textmode</code>
Filter/Pipe	<code>-u</code>	<code>-u</code>	<code>-u, --local-user</code>
	<code>-f</code>	<code>-f</code>	<code>--batch</code>
	Gruppenfunktionen		
Schlüssel zu Gruppe hinzufügen	<code>-</code>	<code>-ga</code>	<code>-</code>
Schlüssel von Gruppe entfernen	<code>-</code>	<code>-gr</code>	<code>-</code>
Schlüssel in Gruppe anzeigen	<code>-</code>	<code>-gv</code>	<code>-</code>
	Hilfe		
Allgemeine Hilfe	<code>-h</code>	<code>-h</code>	<code>-h, --help</code>
Hilfe für Schlüsselfunktionen	<code>-k</code>	<code>-k</code>	<code>-</code>
Hilfe für Gruppenfunktionen	<code>-</code>	<code>-g</code>	<code>-</code>

Tabelle 2.1: PGP: Die wichtigsten Funktionen im Überblick.

Funktion	PGP 2.6.3i	PGP 6.5.8	GPG 1.0.4	S/MIME X.509v3
Verschlüsseln (symmetrisch)				
3DES	–	•	•	•
CAST5	–	•	•	–
BLOWFISH	–	–	•	–
TWOFISH	–	–	•	–
Rijndael	–	–	•	–
IDEA	•	•	–	–
RC2	–	–	–	•
RC4	–	–	–	nur SSL
Schlüssel (asymmetrisch)				
RSA	•	•	•	•
DSA $\hat{=}$ DSS	–	•	•	•
ELG $\hat{=}$ DH	–	•	•	•
Hash-Verfahren				
MD5	•	•	•	•
SHA-1	–	•	•	•
RIPMD160	–	•	•	–
Zertifikate				
PGP	•	•	•	–
X.509	–	•	–	•
Zufallszahlen				
Tageszeit	•	•	•	★
Tastaturbetätigung	•	•	•	★
Mausbewegung	–	•	•	★
Dateien	–	–	–	★

★ = je nach Implementierung

Tabelle 2.2: PGP: Protokolle und Verfahren

2.3 Vergleichstests

Um die Unterschiede der drei Programme PGP-2.6.3i, PGP-6.5.8 und GnuPG genau festzustellen und um zu prüfen, ob die Aussagen in den Anleitungen stimmen, führte ich verschiedene Tests durch.

Die Tests erfolgten hauptsächlich mit den Kommandozeilenversionen unter Linux. Die Windows Version verhält sich auf der Kommandozeile identisch. Die grafische Oberfläche ist völlig anders zu bedienen.

Die Tests wurden immer mit der zusätzlichen Option `-a` durchgeführt, weil diese in E-Mail-Programmen verwendet wird. Durch die Ausgabe der verschlüsselten Daten oder der Unterschrift im Radix-64-Format², gibt es keine Probleme bei der Übertragung im Internet, auch wenn alte E-Mail-Server nur 7-Bit ASCII übertragen können. Bei den Unterschriften wurde zusätzlich die Option `-b` angegeben, um die Unterschrift von den eigentlichen Daten abzutrennen. Somit sind auch Unterschriften für Binärdateien möglich. E-Mail-Programme verwenden diese Option, damit der Klartext dennoch lesbar bleibt, falls der Empfänger PGP zur Überprüfung der Unterschrift nicht besitzt.

2.3.1 Verschlüsseln mit RSA Schlüsseln

Dieser Test soll feststellen, ob es möglich ist, zwischen dem alten PGP-2.6.3i und dem neuen PGP-6.5.8 verschlüsselte Daten auszutauschen. Dazu wurde die Kommandozeilenoption `pgp -ea` verwendet, welche die verschlüsselten Daten im ASCII Format erstellt.

Es wurde mit PGP-2.6.3i ein RSA-512-Bit Schlüssel erzeugt. Dieser wurde exportiert und in PGP-6.5.8 importiert. Dann wurde mit PGP-6.5.8 ein Text für den neu erzeugten Schlüssel verschlüsselt. Dieser konnte problemlos in PGP-2.6.3i entschlüsselt werden.

Es wurde mit PGP-6.5.8 ein RSA-1024-Bit und ein RSA-2048-Bit Schlüssel erzeugt. Diese wurden exportiert und in PGP-2.6.3i importiert. Dann wurde mit PGP-2.6.3i jeweils ein Text für die erzeugten Schlüssel verschlüsselt. Diese konnten problemlos in PGP-6.5.8 entschlüsselt werden.

Die größte Schlüssellänge für RSA-Schlüssel beträgt sowohl bei PGP-2.6.3i als auch bei PGP-6.5.8 2048-Bit. Jedoch erreicht man diese Schlüssellänge bei PGP-2.6.3i nur durch direkte Eingabe, 2048-Bit wird nicht im Menü angeboten. In PGP-6.5.8 werden die kleinen Schlüssellängen nicht mehr im Menü angeboten, weil sie heutzutage nicht mehr sicher sind [Bun99].

2.3.2 Unterschreiben mit RSA Schlüsseln

Dieser Test soll feststellen, ob es möglich ist, zwischen dem alten PGP-2.6.3i und dem neuen PGP-6.5.8 digitale Unterschriften zu erstellen, auszutauschen

²Auch Base-64 genannt: 3 Byte werden in 4 Byte eines 7-Bit Zeichensatzes umcodiert

und zu überprüfen. Dazu wurde die Kommandozeilenoption `pgp -sba` verwendet, welche Unterschriften im ASCII Format erstellt und diese in einer separaten Datei ablegt.

Es wurde mit PGP-2.6.3i eine Unterschrift zu einem Testtext erzeugt. Die Unterschrift konnte mit PGP-6.5.8 auf Korrektheit überprüft werden. Ein Verändern des Testtextes führte zu einer ungültigen Unterschrift. Das Verhalten ist korrekt.

Es wurde mit PGP-6.5.8 eine Unterschrift zu einem Testtext erzeugt. Die Unterschrift konnte mit PGP-2.6.3i auf Korrektheit überprüft werden. Ein Verändern des Testtextes führte zu einer ungültigen Unterschrift. Auch dieses Verhalten ist korrekt.

2.3.3 Verwendung von DSS/DH Schlüsseln

Die von PGP ab Version 5 verfügbaren DSS (Digital Signature Standard)/DH (Diffie-Hellmann) Schlüssel können in PGP-2.6.3i nicht zum Schlüsselbund hinzugefügt werden, sondern werden ignoriert. Somit ist weder das Verschlüsseln von Daten noch eine Überprüfung von Unterschriften mit PGP-2.6.3i möglich, wenn der Kommunikationspartner einen DSS/DH Schlüssel verwendet. In die andere Richtung, also von PGP-6.5.8 ausgehend, ist sowohl eine Verschlüsselung als auch eine Überprüfung der Unterschrift möglich, wenn der Kommunikationspartner PGP-2.6.3i verwendet und somit RSA Schlüssel benutzt (siehe auch Abschnitte [2.3.1](#) und [2.3.2](#)). Eine Kompatibilität im Hinblick auf DSS/DH Schlüssel ist mit PGP-2.6.3i nicht gegeben.

2.3.4 Kompatibilität von PGP mit GnuPG

Das Hinzufügen von PGP-2.6.3i RSA Schlüsseln sowie von PGP-6.5.8 DSS/DH Schlüsseln zum Schlüsselbund von GnuPG ist problemlos möglich. Der Typ des Schlüssels wird korrekt erkannt. Ein von GnuPG erstellter DSS/DH Schlüssel kann problemlos in PGP-6.5.8 importiert werden. Da GnuPG keine RSA Schlüssel erzeugen kann, entfällt dieser Schlüsseltausch mit PGP-2.6.3i .

PGP-2.6.3i und GnuPG

Um mit PGP-2.6.3i kommunizieren zu können, muss GnuPG den symmetrischen IDEA Algorithmus unterstützen. Dieser Algorithmus ist aber noch

einige Jahre patentiert. Es widerspricht GNU (GNU's not UNIX) patentierte Algorithmen zu verwenden. Jedoch ist es möglich GnuPG mit IDEA Unterstützung zu kompilieren. Dann darf es nur noch nichtkommerziell verwendet werden oder man kauft eine Lizenz beim Patentinhaber Ascom. Hätte man dann eine GnuPG Version mit IDEA Unterstützung, könnte man laut [Ash00] mit folgendem Aufruf eine Datei verschlüsseln, die mit PGP-2.6.3i entschlüsselt werden könnte: `gpg --rfc1991 --cipher-algo idea --compress-algo 1 --encrypt --recipient alice text`.

Durch das Patent und den nicht gerade benutzerfreundlichen Kommandozeilenaufruf kann die Kommunikation in Bezug auf verschlüsselte Daten mit PGP-2.6.3i als nicht möglich angesehen werden.

Es wurde mit PGP-2.6.3i eine Unterschrift zu einem Testtext erzeugt. Die Unterschrift konnte mit GnuPG auf Korrektheit überprüft werden. Ein Verändern des Testtextes führte zu einer ungültigen Unterschrift. Das Verhalten ist korrekt.

Da GnuPG das Erzeugen von RSA Schlüsseln nicht ermöglicht, müssen um Unterschriften erzeugen zu können, geheime RSA Schlüssel importiert werden. Es wurde mit PGP-2.6.3i ein neues Schlüsselpaar erzeugt und sowohl der geheime als auch der öffentliche Schlüssel exportiert und in GnuPG importiert, alles laut Beschreibung aus [Ash00]. Dies hat funktioniert. Nicht funktioniert hat dagegen das Erzeugen einer Unterschrift mit dem importierten Schlüssel mit der Begründung von GnuPG, dass das Verschlüsselungsverfahren unbekannt sei.

Unterschriften können von GnuPG verifiziert aber nicht erstellt werden. Das Verschlüsseln von Daten für PGP-2.6.3i ist nicht möglich.

PGP-6.5.8 und GnuPG

Es wurde mit GnuPG eine Testdatei verschlüsselt, welche von PGP-6.5.8 problemlos entschlüsselt werden konnte. Eine mit PGP-6.5.8 verschlüsselte Datei konnte auch problemlos mit GnuPG entschlüsselt werden. Der von beiden Produkten verwendete Verschlüsselungsalgorithmus ist standardmäßig CAST5. Ein Versuch, dies bei PGP-6.5.8 auf z.B. 3DES zu ändern, funktionierte nicht. Die Einstellung des symmetrischen Chiffriercodes wurde ignoriert. Mit GnuPG konnten die Daten mit sämtlichen unterstützten Verfahren, z.B.: 3DES, CAST5 oder BLOWFISH, verschlüsselt werden. Wenn allerdings PGP-6.5.8 das Verfahren nicht kennt, bricht es nach der Eingabe der Passphrase kommentarlos ab.

Ein Austausch von verschlüsselten Daten ist möglich, solange symmetrische Chiffrieralgorithmen verwendet werden, die von beiden Programmen verstanden werden. Dies sind im Moment 3DES und CAST5, siehe auch Tabelle 2.2.

Es wurde mit PGP-6.5.8 eine Unterschrift zu einem Testtext erzeugt. Die Unterschrift konnte mit GnuPG auf Korrektheit überprüft werden. Ein Verändern des Testtextes führte zu einer ungültigen Unterschrift. Das Verhalten ist korrekt.

Es wurde mit GnuPG eine Unterschrift zu einem Testtext erzeugt. Die Unterschrift konnte mit PGP-6.5.8 auf Korrektheit überprüft werden. Ein Verändern des Testtextes führte zu einer ungültigen Unterschrift. Auch dieses Verhalten ist korrekt.

Das Unterschreiben und Überprüfen von Unterschriften ist möglich, solange als Message Digest Algorithmus SHA-1 (Secure Hash Algorithm) oder RIPEMD160 verwendet wird. Eine von GnuPG mit MD5 erstellte Unterschrift konnte PGP-6.5.8 nicht verifizieren. Wenn der Text gefälscht wird, erkennen das beide Programme korrekt. Wenn die Unterschrift gefälscht wird, können beide Programme beispielsweise noch die Schlüssel-ID oder den Erstellernamen ausgeben, je nachdem inwieweit die Datei zerstört ist. GnuPG tut dies aber nur dann, wenn die Prüfsumme entfernt wurde. Sonst erhält man lediglich einen Prüfsummenfehler.

2.3.5 Unterschiede zwischen PGP und S/MIME

Der Hauptunterschied zwischen PGP und S/MIME ist das Konzept des Vertrauens.

Web-of-Trust

Bei PGPs Web-of-Trust unterschreiben die Benutzer gegenseitig ihre Schlüssel, nachdem sie sich von der Echtheit überzeugt haben. Dabei können die Schlüssel sicher ausgetauscht oder über einen dritten Kanal, z.B. Telefon, der Fingerabdruck überprüft worden sein. So entsteht mit der Zeit, völlig dezentral, ein Netz gegenseitigen Vertrauens. Jeder Benutzer hat die volle und explizite Kontrolle darüber, wem er wie weit vertraut. Wenn Alice Bob vertraut und Bob Carol, dann vertraut auch Alice Carol (siehe Abbildung 2.1). Dieses Vertrauensschema kann noch feiner eingestellt werden. Man kann einem Schlüssel auch nur halb vertrauen (marginal trust), dann müssen mindestens zwei unabhängige Unterschriften die Echtheit bestätigen.

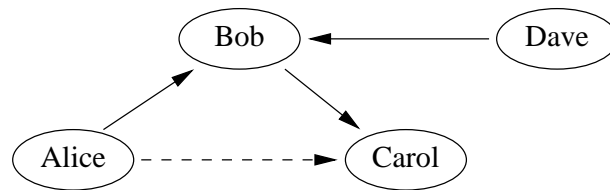


Abbildung 2.1: PGP: Web-of-Trust.

Das Web-of-Trust hat aber nicht nur Vorteile. Mit steigender Anzahl Schlüsseln wird das Navigieren durch das Netz schwierig und unübersichtlich. Fehlende Querverbindungen zwischen gut vernetzten Anwendergruppen erschweren die Kommunikation [Luc99a]. Das Pflegen des Vertrauensstatus wird mühselig, wenn z.B. bestimmte Schlüsseln zurückgezogen wurden und dies noch nicht bei allen Teilnehmern vermerkt wurde.

Certificate Chain

Das Konzept von S/MIME basiert auf einer strengen Hierarchie von X.509 Zertifikaten (siehe Abbildung 2.2). Zentral an oberster Stelle befindet sich die Zertifizierungsinstanz, die Certificate Authority. Diese gibt Benutzerzertifikate aus, die Informationen über den Inhaber, seinen öffentlichen Schlüssel und einen Gültigkeitszeitraum enthalten. Nachdem die Inhalte überprüft sind, unterschreibt die Ausgabestelle das Zertifikat. Erst dann ist es verwendbar.

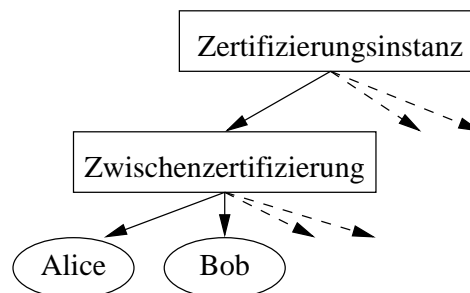


Abbildung 2.2: S/MIME: Certificate Chain

Solange also den übergeordneten Zertifikaten vertraut wird, sind automatisch auch alle davon abhängigen Zertifikate vertrauenswürdig (außer sie sind abgelaufen oder zurückgezogen). D.h. es ist nur nötig dem obersten Zertifikat durch geeignete Mittel z.B. der Prüfung des Fingerabdruckes zu vertrauen.

Jedoch kann auch jedem Zertifikat explizit vertraut werden, wenn man die übergeordneten Zertifikate nicht prüfen kann oder will.

Wenn das Trustcenter bestimmte Anforderungen erfüllt, sind die mit den Zertifikaten erstellten Unterschriften sogar nach deutschem Gesetz rechtsgültig. Derzeit erfüllen nur die TeleSec (Telekom) und Signtrust (Post) diese Anforderungen.

Auch bei der Zertifikatskette (Certificate Chain) ist es schwierig, vorzeitig zurückgezogene Schlüssel durch CRL (Certificate Revocation List), schwarze Listen, bekannt zu machen.

X.509 Zertifikate in PGP

X.509 Zertifikate können von der grafischen Windows PGP Version importiert werden. Sie sind aber nur dann benutzbar, wenn ihnen implizit vertraut wird (muss erst aktiviert werden). Damit erreicht man volles Vertrauen (full trust) des Schlüssels bzw. des Zertifikates. Dies funktioniert nur bei Endbenutzerzertifikaten, nicht aber bei den übergeordneten. Sie bleiben immer im Status nicht vertraut (keine Aktivierung möglich). Importieren kann man sein privates Zertifikat z.B. im PKCS#12 (Personal Information Exchange Syntax Standard) Format. Bei den Tests konnten bestimmte Zertifikate (von Web.de) nicht importiert werden, weil PGP beim Versuch jedesmal abgestürzt ist.

2.4 Zusammenfassung

Mit PGP und RSA Schlüsseln ist eine Ver- und Entschlüsselung in beide Richtungen problemlos möglich. Unterschriften werden auch korrekt gehandhabt, solange es sich um RSA Schlüssel handelt. Dabei ist es unerheblich mit welcher PGP-Version die Schlüssel erzeugt wurden. DSS/DH Schlüssel können nicht zwischen dem alten und neuen PGP ausgetauscht werden. Hier ist keine Kommunikation möglich.

GnuPG kann sich mit PGP-2.6.3i nicht verständigen, lediglich die Überprüfung von Unterschriften ist möglich. Bei PGP-6.5.8 sieht es besser aus. Grundsätzlich ist eine Kommunikation möglich, solange man die Standardeinstellungen nicht verändert. Dann funktioniert das Verschlüsseln und auch das Erstellen und Überprüfen von Unterschriften. Will man andere Algorithmen verwenden, kann die Sache schiefgehen, auch wenn es laut Tabelle 2.2 funktionieren müsste.

Da die PGP-Versionen 5.5 bis 6.5.3 den ADK-Fehler enthalten, kommen diese für die Public Key Infrastruktur nicht in Frage. Da das freie GnuPG bis jetzt von den wenigsten Programmen unterstützt wird, weil es unter anderem auch völlig andere Kommandozeilenoptionen benötigt, kommt dieses auch nicht in Frage. Das Schöne an GnuPG wäre allerdings, dass es sich als einziges Programm an die OpenPGP Konventionen hält (siehe RFC (Request for Comments) 2440). PGP-2.6.3i und PGP-6.5.8 können sich wunderbar verständigen wenn es sich um RSA Schlüssel handelt. Ein Problem ergibt sich aus der Tatsache, dass die meisten Benutzer bzw. Firmen erst einmal die neueste PGP Version wählen. Die alte Version muss man erst suchen. Weiterhin wird dem Benutzer sowohl bei der Kommandozeilenversion als auch bei der Windows Oberfläche der DSS/DH Schlüssel als Voreinstellung angeboten. Bei Windows wird er sogar explizit empfohlen.

Bei den zu PGP kompatiblen Programmen ist PGP-6.5.8 die beste Wahl. S/MIME ist prinzipiell nicht anders. Deshalb fällt die Entscheidung schwer, welches Public Key Verfahren zum Aufbau der Infrastruktur verwendet werden soll. Um sich für eines der beiden Produkte zu entscheiden, wurde eine Umfrage im Betrieb gemacht, die Unterstützung der E-Mail Programme untersucht und versucht herauszufinden, welches Verfahren mehr Firmen benutzen.

Kapitel 3

Aufbau der Public Key Infrastruktur

3.1 Was ist eine Public Key Infrastruktur?

Eine PKI (Public Key Infrastruktur) ist ein Sicherheitssystem, das sich digitaler Zertifikate bedient, um Benutzer zu identifizieren, Kommunikation oder Dateien und Dokumente zu verschlüsseln sowie digital zu unterschreiben. Dazu wird ein Public Key Verfahren, d.h. asymmetrische Verschlüsselung, eingesetzt. Jeder Teilnehmer besitzt einen privaten und einen öffentlichen Schlüssel, mit dem er Dokumente ver- und entschlüsseln kann. Will ein Mitarbeiter eine Nachricht an einen anderen Mitarbeiter schicken, so benutzt er dessen öffentlichen Schlüssel um die Nachricht zu verschlüsseln, und er unterschreibt mit seinem privaten Schlüssel. Nur der Empfänger kann die Nachricht mit seinem eigenen privaten Schlüssel entschlüsseln. Die beigefügte Unterschrift bestätigt eindeutig die Identität des Senders und ist nicht abstreitbar.

Eine PKI schützt die *Privatsphäre*, weil Daten nicht von unbefugten gelesen werden können. Sie garantiert die *Unverletztheit* (Integrität) der Daten weil sie nicht unbemerkt geändert werden können. Unterschriften sind *nicht abstreitbar* (Non-repudiation) und *authentisch*.

Eine PKI besteht aus mehreren Teilen. Es wird die *Schlüsselerzeugung* und *Schlüsselspeicherung* benötigt. Die Speicherung teilt sich auf in die des privaten Schlüssels, der von den Benutzern geheimgehalten werden muss, und die des öffentlichen Schlüssels, der auf einem öffentlichen Server liegen soll. Die *Registrierungsinstanz* (RA) nimmt den Zertifikatsantrag entgegen und prüft ihn. Danach leitet sie ihn weiter an die *Zertifizierungsinstanz* (CA), die

unter Zuhilfenahme eines *Zeitstempeldienstes* das Zertifikat ausstellt. Dieses wird dem Antragsteller zugestellt. Auf dem öffentlichen Server findet die *Zertifikatsverteilung* statt. Kommt der private Schlüssel abhanden oder wird er geklaut, so muss ein vorzeitiger *Zertifikatsrückruf* stattfinden. Siehe auch Abbildung ??.

Oft wird zusätzlich gefordert, die Schlüsselpaare der Anwender bei einem Verlust wiederherstellen zu können. Diese Thematik, die unter dem Begriff *key escrow* etwas in Verruf geraten ist und in *key recovery* umbenannt wurde, wird immer noch heiß diskutiert. Für die Firma Innovations GmbH wird es das nicht geben, weil die geeignete Sicherung der privaten Schlüssel nicht möglich ist. Es werden alle Mitarbeiter angewiesen, selbst Sicherungskopien ihrer Schlüssel anzufertigen.

3.2 Mitarbeiterumfrage

Die Firma überlässt jedem Mitarbeiter selbst die Wahl des E-Mail Programmes. Deshalb verwendet jeder seinen ganz persönlichen Favorit. Um einen Überblick der verwendeten Programme zu bekommen, wurde eine Umfrage durchgeführt.

3.2.1 Auswahl der Fragen

Der Fragebogen wurde als HTML Formular im Intranet bereitgestellt (Siehe Abbildung 3.1). Es wurde darauf Wert gelegt, mit möglichst wenigen Fragen (= kurze Bearbeitungszeit für die Mitarbeiter, höchstens 5 Minuten) möglichst viel an Information herauszubekommen. Desweiteren sollte das komplette Formular nicht größer als eine Bildschirmseite sein.

Es gab drei Kategorien von Fragen:

Allgemeine Fragen: Name des Mitarbeiters, ob er Kontakt zu Kunden hat und inwieweit bei ihm PGP bekannt ist?

Fragen zum E-Mail Programm: Welches E-Mail Programm er benutzt und ob er bereit wäre es zu wechseln?

Fragen zum Web-Browser: Frage nach dem Programm und es sollte dessen SSL Verschlüsselungsstärke mittels einer Testseite von Fortify (<http://www.fortify.net/sslcheck.html>) bestimmt werden.

Die Umfrage wurde am 17. Oktober 2000 gestartet und dauerte 3 Wochen.

Mitarbeiterumfrage PGP

von Daniel Hirscher

Im Rahmen meiner Diplomarbeit zum Thema: "Aufbau einer Public Key-Infrastruktur mit PGP in einem mittelständischen Unternehmen" sollte ich ein paar Sachen von Euch wissen. Die Umfrage erfolgt nicht anonym, da ich wissen muss, wer was benutzt. Außerdem gibt man ja keine Meinung ab.

Allgemeine Fragen

Wie heißt du?

Name:

Musst du mit Kunden kommunizieren, denen du vertrauliche Daten per E-Mail übermittelst?

- Ja.
- Nein.

Hast du schon mal was von Pretty Good Privacy (PGP), einem Programm zum Ver- und Entschlüsseln von E-Mails, gehört?

- Ja, ich habe ein Schlüsselpaar und benutze es.
- Ja, ich habe ein Schlüsselpaar, benutze es aber nicht.
- Habe schon mal davon gehört.
- Nein, nie gehört.

Welches Betriebssystem ist auf deinem Computer installiert?

- Windows NT
- Windows 98
- Linux

Fragen zum E-Mail Programm

Wie heißt das derzeitige Programm, das du zum Lesen von E-Mails benutzt?

Anderes:

Würde es dir was ausmachen, das Programm zum Lesen deiner E-Mails zu wechseln, wenn es damit nicht möglich ist E-Mails zu verschlüsseln und digital zu unterschreiben?

- Ja, ich will meins behalten.
- Nein, ich würde auch wechseln.

Fragen zum Web-Browser

Welchen Webbrowser benutzt du hauptsächlich für das Internet?

Anderes:

Bitte klicke auf [Fortify SSL Check](#) (es öffnet sich ein neues Fenster) und trage das in einem farbigen Balken erscheinende Testergebnis (cipher und key) hier ein.

Cipher, Key:

Vielen Dank.

Abbildung 3.1: Fragebogen im Intranet.

3.2.2 Ergebnisse

Von 51 Mitarbeitern nahmen 46 (90%) teil. Davon gaben 27 (59%) an, Kontakt zu Kunden zu haben. Diese sollten auf jeden Fall sicher (d.h. verschlüsselt) mit den Kunden kommunizieren.

Der Bekanntheitsgrad von PGP ist sehr hoch. 50% der Teilnehmer haben schon davon gehört, 26% kennen sich damit aus und 13% benutzen es sogar manchmal.

Es ist eine Vielfalt von E-Mail Programmen in Benutzung. Eine genaue Aufschlüsselung zeigt Abbildung 3.2. Dabei wird Netscape mit 54% am häufigsten eingesetzt.

Den Schwerpunkt meiner Betrachtung lege ich auf die Mitarbeiter, die Kundenkontakt haben. Dabei fallen nach Abbildung 3.3 die exotischen Programme weg, aber Netscape behält mit 52% Anteil dennoch seinen Spitzenplatz. Auf die Frage, wer bereit wäre sein Programm zum Lesen der E-Mail zu wechseln, sind mit 56% geringfügig mehr dazu bereit als dagegen. Um möglichst wenigen Mitarbeitern ein anderes Programm vorzuschreiben, wurde folgendes ausgezählt:

1. Wieviele Mitarbeiter mit Kundenkontakt müsste man überreden ihr Programm zu wechseln (was sie nicht wollen), weil es nicht mit PGP zusammenarbeitet? Dies wären vier Personen.
2. Wieviele Mitarbeiter mit Kundenkontakt müsste man überreden ihr Programm zu wechseln (was sie nicht wollen), weil es nicht S/MIME fähig ist? Dies würde niemanden betreffen.

Zusätzlich wurde noch nach den Web-Browsern gefragt. Auch hier hat Netscape Communicator mit 56% den größten Anteil, gefolgt vom Microsoft Internet Explorer mit 40% und Opera mit den restlichen 4%. Die Überprüfung der SSL Verschlüsselungsstärke brachte ein gutes Ergebnis, denn fast alle Mitarbeiter (80%) haben die derzeit höchste Stärke von RC4/128-Bit. Die übrigen verfügen über RC4/40-Bit, dies gilt als nicht mehr sicher [BDR⁺96].

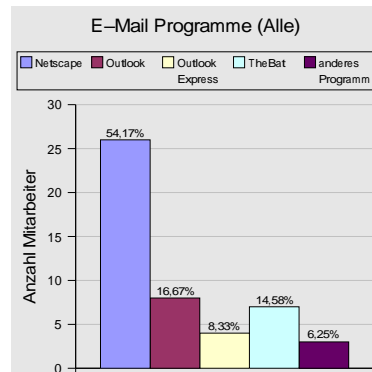


Abbildung 3.2: E-Mail Programme (alle Mitarbeiter).

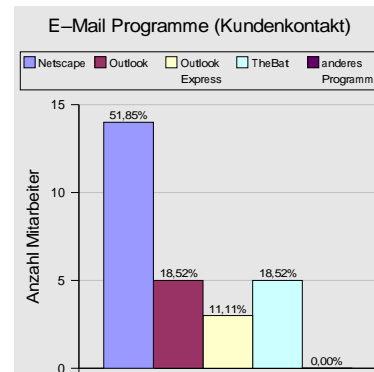


Abbildung 3.3: E-Mail Programme (Mitarbeiter mit Kundenkontakt).

3.3 Einbindung von Public Key Verfahren in vorhandene E-Mail Programme

Anhand der Umfrage wurde bekannt, welche E-Mail Programme verwendet werden. Der nächste Schritt ist, herauszufinden wie die einzelnen Programme mit PGP und S/MIME zurecht kommen.

3.3.1 Programme

Bei allen E-Mail Programmen läuft das Unterschreiben und/oder Verschlüsseln auf ähnliche Weise ab.

1. Die E-Mail wird wie üblich verfasst.
2. Es muss ein Button angeklickt oder ein Menüpunkt ausgewählt werden, wenn man unterschreiben oder verschlüsseln möchte. Meist handelt es sich um mehr oder weniger eindeutige Bildchen.
3. Man klickt auf „E-Mail versenden“.
4. Jetzt wird der Benutzer evtl. nach der Passphrase für seinen privaten Schlüssel gefragt, wenn zuvor „Unterschreiben“ gewählt wurde.
5. Wenn ein passender öffentlicher Schlüssel des Empfängers gefunden wurde, wird die E-Mail verschlüsselt, sofern gewünscht.

6. Nun wird wirklich versendet.

In Tabelle 3.1 sind die Public Key Fähigkeiten der in Frage kommenden Programme zusammengefasst. Außer den dort genannten verstehen noch die Programme: Lotus Notes 5, Baltimore MailSecure, OpenSoft ExpressMail, SSE TrustedMIME, WorldTalk und Entegrity Ashure Mail den Umgang mit S/MIME.

Programm	PGP	S/MIME
Netscape	nicht möglich	eingebaut
Outlook	Plugin von PGP	eingebaut
Outlook Express	Plugin von PGP	eingebaut
TheBat!	über eigene DLL	eingebaut

Tabelle 3.1: Public Key Verfahren in E-Mail Programmen

Netscape

Die Versuche, PGP mit Netscape zu benutzen, scheiterten. Es gibt einen veralteten Plugin, der das Verschlüsseln und Unterschreiben erlaubt aber keinerlei Möglichkeiten bietet, eine empfangene E-Mail zu überprüfen bzw. zu entschlüsseln. Ein weiteres Programm (Mollusc), das behauptet PGP in Netscape zu bringen, funktioniert nicht mit der aktuellen Version zusammen und wäre zudem umständlich zu bedienen. Die Alternative, den Text über die Zwischenablage zu Ver- und Entschlüsseln, ist nicht benutzerfreundlich.

Die Unterstützung von S/MIME ist seit langem eingebaut, funktioniert sehr gut und ist einfach zu bedienen. Netscape besitzt eine eigene Zertifikatsverwaltung, die mittels eines Passwortes geschützt ist.

Outlook und Outlook Express

Beide Programme haben den gleichen Funktionsumfang, sind aber unterschiedlich aufgebaut und zu bedienen.

Über einen von PGP mitgelieferten Plugin steht die Benutzung von PGP zur Verfügung. Ankommende E-Mails werden dennoch nicht automatisch überprüft oder entschlüsselt. Dies muss von Hand über die Zwischenablage geschehen.

Die Unterstützung von S/MIME ist eingebaut (ab Outlook 98, Outlook Express 4.0), funktioniert gut und ist einfach zu bedienen. Die Zertifikatsverwaltung übernimmt Windows.

TheBat!

Für dieses Programm liefert PGP keinen Plugin mit. TheBat! bringt diesen selbst mit. Dabei bietet es für PGP den höchsten Komfort im Gegensatz zu den anderen Programmen. TheBat! bietet die Option, dass ausgehende E-Mails immer verschlüsselt werden (sofern der Empfängerschlüssel vorhanden ist).

Die Unterstützung von S/MIME ist eingebaut, hat aber noch einige Mängel. TheBat! hat eine eigene Zertifikatsverwaltung. Der private Schlüssel ist mittels des Export-Passwortes geschützt. S/MIME funktioniert erst ab Version 1.48 und ist noch etwas umständlich zu bedienen.

Allgemeines

Folgende Aussagen treffen auf alle drei Programme zu:

- Keines der Programme kann bei Benutzung von PGP den Dateianhang (Attachment) automatisch mitverschlüsseln. Dies ist besonders nachteilig, da der Text meist nur auf einen vertraulichen Dateianhang verweist, der eigentlich zu sichern wäre.
- Bei S/MIME werden Dateianhänge grundsätzlich mitverschlüsselt. Es wird alles unter einem MIME-Typ zusammengefasst. Bei Unterschriften wird eine abgetrennte Unterschrift erzeugt und als Dateianhang angefügt. Falls das E-Mail Programm des Empfängers die Unterschrift nicht prüfen kann (weil es z.B. S/MIME nicht unterstützt), ist die eigentliche Nachricht unverändert und damit lesbar.
- Alle Programme haben die Möglichkeit, bei Benutzung von S/MIME ausgehende E-Mails immer zu unterschreiben und/oder zu verschlüsseln, sofern sich für den Empfänger ein öffentlicher Schlüssel findet. Dadurch kann es prinzipiell nicht vorkommen, dass man vergisst zu verschlüsseln.

3.3.2 Zusammenfassung

Die Unterstützung von S/MIME ist bei allen in der Firma zum Einsatz kommenden Programmen gegeben. Allein durch die Integration in die Programme ist die Bedienung des S/MIME Verfahrens schon besser gelöst. Nachteilig für PGP ist die fehlende Unterstützung für Netscape.

Allgemein lässt sich sagen, dass bei den E-Mail Programmen S/MIME mehr Pluspunkte sammeln konnte. PGP ist ein eigenes Programm und läuft damit immer außerhalb des E-Mail Programmes. Das führt zu gewissen Anbindungsproblemen, die immer über Plugins gelöst werden. Dabei wird auch keine Rücksicht darauf genommen, ob zukünftige Versionen mit alten zusammenarbeiten.

3.4 Wahl des Public Key Verfahrens

Um sich für eines der beiden Verfahren zu entscheiden, kann ich mich auf drei Punkte stützen:

1. Die Umfrage, die erkennen lässt welche Programme benutzt werden.
2. Die Unterstützung der Verfahren durch die Programme.
3. Anhand dem allgemeinen Trend in anderen Firmen.

Die Punkte eins und zwei wurden in den vorhergehenden Abschnitten behandelt. Punkt drei wird im Folgenden erläutert.

3.4.1 Zahlen und Statistiken

Konkrete Zahlen oder gar Statistiken, ob nun mehr Firmen PGP oder S/MIME benutzen, konnte ich nicht in Erfahrung bringen. Weder im Internet noch im Usenet oder bei Instituten, Organisationen oder Firmen wurden Erhebungen durchgeführt.

Aus aktuellen Artikeln aus Magazinen und Zeitschriften lässt sich ein Trend in Richtung S/MIME erkennen. [NS01, S. 231] schreibt: „... ist X.509 der eindeutig vorherrschende Standard“. Eine Zunahme von X.509 Zertifikaten für S/MIME bestätigt auch das TC TrustCenter auf eine Anfrage per E-Mail. Dagegen steht die Vermutung der Firma *secunet*: „Es sieht aber so aus, als

ob in Deutschland vermehrt PGP eingesetzt wird.“. [Abr00] bemerkt: „Most of them, however, are ‘die-hard Phil fans and encryption gurus’“.

Ein Hinweis auf die Menge der PGP Schlüssel, die im Mai 1999 verwendet wurden, ist aus [Luc99b] zu erfahren. Damals waren über eine halbe Million PGP Schlüssel auf dem internationalen Schlüsselservers abgelegt. S/MIME verwendet X.509 Zertifikate, die von verschiedenen Trustcentern ausgestellt werden. Einer der größten, VeriSign, hat bis zum 3. Quartal 2000 über 400.000 Zertifikate verkauft, darin sind aber auch reine SSL Server Zertifikate enthalten.

3.4.2 Alternativen

Die Vielfalt der E-Mail Programme ist für die Einführung einer Public Key Infrastruktur nicht gerade von Vorteil. Jedes Programm verhält sich ein wenig anders bei der Benutzung. Einige sind mehr, andere weniger komfortabel zu bedienen.

Es gibt nun drei Möglichkeiten wie weiter vorgegangen wird:

1. Es wird in der Firma ein E-Mail Programm vorgeschrieben, das gut mit PGP zusammenarbeitet. Hierbei muss PGP gekauft werden, da die freie Version im kommerziellen Umfeld nicht betrieben werden darf.
2. Es wird in der Firma ein E-Mail Programm vorgeschrieben, das gut mit S/MIME funktioniert. Damit können X.509 Zertifikate verwendet werden. Hier gibt es wiederum zwei Möglichkeiten:
 - (a) Es werden X.509 Zertifikate bei einem Trustcenter gekauft. Diese sind ein Jahr gültig. Die Unterschrift ist aber nur dann nach dem deutschen Signaturgesetz rechtskräftig, wenn die Zertifikate von einem der zwei zertifizierten Trustcenter (TeleSec, SignTrust) stammen (Stand: Dezember 2000).
 - (b) Es werden die benötigten X.509 Zertifikate selbst erstellt. Die Gültigkeitsdauer kann selbst bestimmt werden. Unterschriften sind nicht rechtsgültig. Die Kommunikationspartner müssen das Root-Zertifikat der selbsterstellten CA (Certificate Authority) installieren.

3.4.3 Entscheidung für S/MIME

Ich entscheide mich für S/MIME aus folgenden Gründen:

- Mit S/MIME können Attachments automatisch verschlüsselt werden.
- Die Gültigkeit einer Unterschrift wird sofort angezeigt.
- Allgemein ist S/MIME besser in die Programme integriert als PGP.
- Eine Vertrauenshierarchie ist einfacher aufzubauen als PGPs Web-of-Trust.
- Es ist benutzerfreundlich.
- Kein Mitarbeiter muss sein E-Mail Programm wechseln.
- Netscape unterstützt S/MIME und ist das in der Firma am häufigsten verwendete Programm um E-Mail zu lesen.
- Die Zertifikate können auch z.B. zur Authentifizierung gegenüber einem Server oder zum digitalen Unterschreiben von Programmen eingesetzt werden.

Die Vorgehensweise ist 2b), die Zertifikate werden selbst erstellt. Dies hat folgende Gründe: Der Firma Innovations kommt es hauptsächlich darauf an Daten zu verschlüsseln. Ob der Sender wirklich der ist, der es zu sein scheint, ist von untergeordneter Bedeutung. Dabei ist zu beachten, dass die Zertifikatsinhaber (die Mitarbeiter) der Zertifizierungsstelle persönlich bekannt sind und somit einen von Grund auf hohen Vertrauensstatus besitzen. Wenn ein Zertifikat einem Kommunikationspartner beim Kunden ausgestellt wird, so ist dieser auch mindestens einem Mitarbeiter innerhalb der Firma gut bekannt. Der Unterschied zu einem kommerziellen Trustcenter ist, dass zu Beginn die Root CA Zertifikate verteilt werden müssen.

3.4.4 Weitere Möglichkeiten

Die sicherste Methode zur Aufbewahrung des privaten (geheimen) Schlüssels bzw. des privaten Zertifikates ist z.B. eine Chipkarte. Der Chip sollte einen eigenen Zufallszahlengenerator enthalten, Schlüssel erzeugen und speichern können, Unterschriften erstellen, passwortgeschützt sein und sicherstellen, dass der Schlüssel die Karte unter keinen Umständen verlassen kann. Bei Einsatz einer Chipkarte benötigt man passende Lesegeräte und Programme, die es unterstützen. Einige Trustcenter bieten X.509 Zertifikate auf Chipkarten an. Eine ähnliche Möglichkeit ist das Produkt iKey, das so groß ist wie ein Schlüsselanhänger und das in den USB (Universal Serial Bus) Port

eingesteckt wird (man könnte es mit einem Zündschlüssel vergleichen). Man benötigt dabei keine zusätzliche Hardware und hat die gleiche Funktionalität wie bei Chipkarten.

3.5 Aufbau der Certificate Authority

Für den Aufbau der CA wurde das Skriptpaket `pyCA` gewählt. Es verwendet OpenSSL und stellt ein Web-Interface zur Verfügung, über das Zertifikate angefordert, gesucht und installiert werden können.

Für die Firma Innovations ist eine dreistufige Zertifizierungshierarchie zu empfehlen (siehe Abbildung 3.4). Durch die Zwischenzertifizierungsstellen ist eine Aufteilung in drei logische Bereiche möglich. Damit kann ein Endbenutzerzertifikat in den Möglichkeiten eingeschränkt werden.

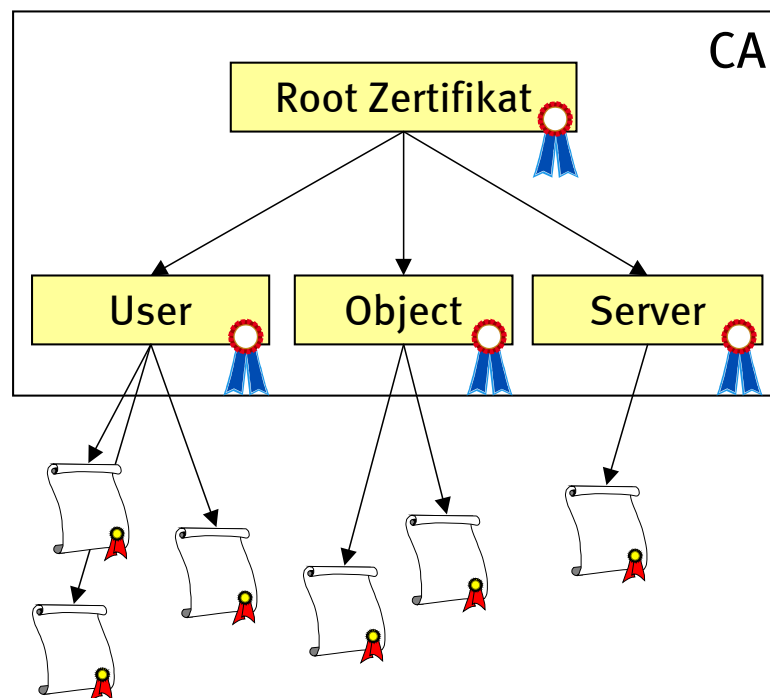


Abbildung 3.4: Zertifizierungshierarchie.

3.5.1 Zertifikattypen

Für die Zertifizierungsstellen wurde folgende Aufteilung vorgenommen:

Root Diese CA gibt keine Benutzerzertifikate aus. Sie dient ausschließlich dazu, die nachfolgenden Zertifikate der Zwischenzertifizierungsstellen auszustellen. Sie kann diese im Falle eines Verlustes oder einer Kompromittierung auch wieder zurückziehen.

Wenn diesem Root CA Zertifikat mittels der Überprüfung des Fingerabdrucks vertraut wird, sind alle davon abhängigen Zertifikate gültig.

User Diese CA stellt Zertifikate aus, die zur Benutzung mit S/MIME zur gesicherten E-Mail Kommunikation genutzt werden können. Gleichzeitig dienen sie zur Authentifizierung gegenüber mit SSL gesicherten Servern, die dies verlangen.

Object Diese CA stellt Zertifikate aus, mit denen z.B. Java-Applets unterschrieben werden können, die im Web-Browser laufen. Es ist wichtig, Code, wie z.B. Java Programme oder beliebige ausführbare Programme, geeignet zu sichern, damit sie als vertrauenswürdig angesehen werden. So kann z.B. erkannt werden, ob die Programme z.B. durch Viren verändert wurden.

Server Diese CA stellt SSL Server Zertifikate aus. Damit können gesicherte Verbindungen über unsichere Kanäle realisiert werden, so dass nur autorisierte Benutzer auf bestimmte Inhalte zugreifen dürfen. Es wird hauptsächlich benutzt um HTTP (Hypertext Transfer Protocol) Übertragungen zu verschlüsseln. SSL kann aber auch benutzt werden, um das Abholen von E-Mail über POP3 (Post Office Protocol) oder IMAP (Interactive Mail Access Protocol) zu sichern.

3.5.2 Schlüsselerzeugung beim Benutzer

Die Schlüsselerzeugung erfolgt nicht bei der CA, sondern beim Benutzer selbst. Dies hat folgende Gründe:

- Der Benutzer soll Vertrauen in die PKI entwickeln. Vor allem soll er erkennen, dass nur er der alleinige Besitzer seines privaten Schlüssels ist.
- Eine sichere, physikalische Löschung von privaten Schlüsseln (bzw. Dateien überhaupt) von Datenträgern ist nur mit hohem Aufwand möglich. Dies wäre nötig, wenn die privaten Schlüssel auf einem anderen Computer erstellt würden.

- Da die Speicherung von Schlüsseln ebenfalls verschlüsselt erfolgt, muss eine Passphrase vergeben werden. Würde der Schlüssel bei der CA erzeugt werden, müsste dem Benutzer diese Passphrase übermittelt werden, was wiederum unsicher ist. Außerdem müsste sie der Benutzer ändern, was er vielleicht nicht tut.

3.6 Schulung der Mitarbeiter

Am Freitag den 01. Dezember 2000 wurde auf der Mitarbeiterversammlung das Vorhaben, d.h. den Aufbau der CA und der damit erforderlichen Zertifikaterstellung, mit einer kleinen Präsentation vorgestellt.

Am Dienstag, den 12.12.2000, und Mittwoch, den 13.12.2000, habe ich im Besprechungsraum eine Einführung in die Public Key Infrastruktur gegeben. Meine Präsentation gliederte sich in zwei Teile: einen allgemeinen Teil und einen technischen für die Interessierten (siehe Anhang ??).

Im allgemeinen Teil erklärte ich, was eine PKI, ein Zertifikat und eine CA ist, wie die CA aufgebaut ist, wie ein Zertifikat zusammengesetzt ist und wie es erstellt wird. Die Zuhörer erfuhren wie es im großen und ganzen funktioniert und was sie selbst dabei tun müssen. Ich verzichtete so weit als möglich auf Fachbegriffe und versuchte den Sachverhalt anhand von eindeutigen Bildern so einfach wie möglich darzustellen.

Im technischen Teil erklärte ich die Funktionsweise von symmetrischer, asymmetrischer und hybrider Verschlüsselung. Die genaue Zusammensetzung eines Zertifikates und wie eine digitale Unterschrift erzeugt und geprüft wird. Zusätzlich zeigte ich noch den RSA Algorithmus. Darauf meinte ein Zuhörer: „So einfach ist das?“.

Zusammenfassend kann man sagen, dass die meisten Zuhörer bis zum Schluss interessiert dabei waren. Selbst die Empfangsdame hat die wichtigsten Aussagen mitnehmen können und verfügt über ein grundsätzliches Verständnis und weiß was sie mit ihrem Zertifikat tun kann.

Bedenken gab es, ob man jetzt einfach so mit dem digitalen Unterschreiben loslegen kann und die Kunden damit konfrontiert oder ob man die Kunden warnt oder einen Hinweis sendet, dass die Firma Innovations demnächst sicher über das Internet kommunizieren will.

3.7 Durchführung

Auf einem Computer ohne Netzwerkanschluss wurden die privaten Schlüssel des Stammzertifikats (Root) und der Zwischenzertifikate erstellt.

Das Root CA Zertifikat wurde selbst unterschrieben. Damit konnten die Zwischenzertifizierungsstellen unterzeichnet werden.

Diese vier Zertifikate bilden die CA.

Danach wurden noch leere CRL erstellt. In diesem Moment wurden ja noch keine Zertifikate zurückgerufen.

Abbildung 3.5 zeigt die Übersicht der CA Zertifikate und deren CRLs. Von hier aus können sie in die Web-Browser installiert oder angesehen werden.

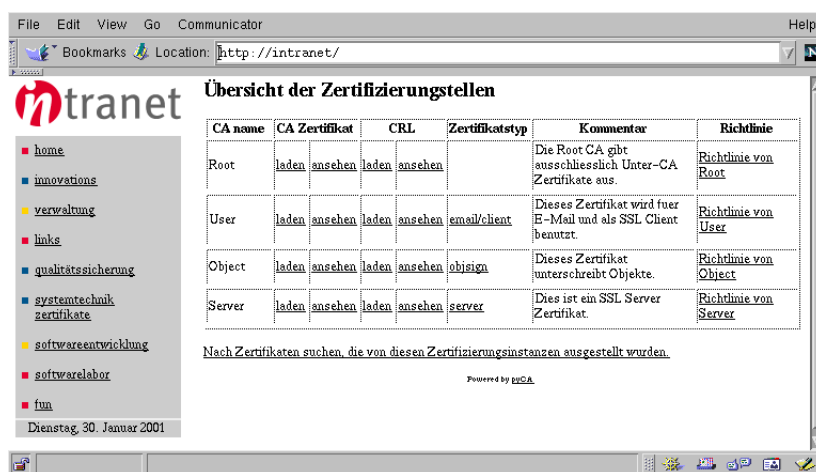


Abbildung 3.5: Übersicht der Zertifizierungsstellen.

Die Zertifikatsdaten, Seriennummer, Gültigkeitszeitraum und, ganz wichtig, der Fingerabdruck der vier Zertifikate wurde auf Papier ausgedruckt. Dieses Blatt Papier wurde traditionell mit einem Kugelschreiber unterschrieben. Es dient der Überprüfung der CA Zertifikate über einen dritten Kanal, z.B. mündlich über Telefon oder durch Mitbringen des Papiers.

Für jeden Mitarbeiter habe ich mir ein Einmalpasswort ausgedacht. Eigentlich wollte ich diese mit der monatlichen Gehaltsabrechnung (dritter Kanal) den Mitarbeitern zukommen lassen. Da ich aber keine weiteren drei Wochen warten wollte, habe ich die Einmalpasswörter persönlich verteilt. Dabei habe ich auch bei der Erstellung der Zertifikate (siehe Abbildung 3.6) geholfen und nochmal auf wichtige Punkte wie die Bedeutung der Schlüsselstärke, dass der

private Schlüssel den eigenen Rechner nicht verlässt und dass es wichtig ist ein Backup vom eigenen Zertifikat zu machen, hingewiesen. Ich habe nochmal die Funktionsweise erklärt und warum man eine E-Mail bekommt, die man zurückschicken muss. Damit wird überprüft ob die E-Mail Adresse existiert.

The screenshot shows a Netscape Communicator browser window displaying a web page titled "Eingabeformular für Zertifikatsantrag". The page is part of an intranet site, as indicated by the URL "http://intranet/". On the left side, there is a navigation menu with links to "home", "innovations", "verwaltung", "links", "qualitätssicherung", "systemtechnik", "zertifikate", "softwareentwicklung", "softwarelabor", and "fun". The main content area contains the following information:

- Zertifizierungsstelle:** User
- Zertifikatstyp:** email/client
- Kommentar:** Dieses Zertifikat wird fuer E-Mail und als SSL Client benutzt.

Below this, it states: "Zertifikate dieses Typs sind 730 Tage gültig, etwa bis 2003-01-30." and "Hier kann ein Zertifikat beantragt werden, indem man die Felder ausfüllt. Durch einen Klick auf die Parameternamen kann man weitere Informationen über die Felder und ihre Bedeutung abrufen." A note indicates that parameters marked with an asterisk (*) are mandatory.

The form fields are as follows:

Browser Software *	Netscape Navigator
Einmalpasswort *	*****
Landescode *	DE
Bundesland	Baden-Wuerttemberg
Stadt	Innenstaad
Organisation *	Innovations gmbH
Name (Vorname Nachname) *	Daniel Hirsche
E-Mail Adresse *	Daniel.Hirsche@innovations.de

An "Abschicken" button is located at the bottom of the form.

Abbildung 3.6: Eingabemaske für den Zertifikatsantrag.

Wenn sich einige Zertifikatsanträge angesammelt hatten, wurden sie auf eine Diskette kopiert. Die Diskette wird dazu benutzt, die Zertifikatsanträge auf den nicht im Netz befindlichen Computer zu bringen und von dort die unterschriebenen, bzw. die fertigen Zertifikate zurück auf den Server zu transferieren.

Auf dem netzlosen Rechner befinden sich die privaten Schlüssel der CA. Damit werden die Zertifikatsanträge nach einer Überprüfung, ob die richtige Person das richtige Einmalpasswort verwendet hatte, unterschrieben. Siehe auch Abbildung ??.

Nach dem Transfer auf den Server, auf dem alle öffentlichen Zertifikate liegen, werden sie automatisch den Antragstellern zugestellt und in der Zertifikatsdatenbank abrufbar gemacht (siehe Abbildungen 3.7 und 3.8).

Abbildung 3.7: Eingabemaske für Suchanfrage.

CA Name	Seriennummer	Gültig bis	Name	E-Mail	Stadt	Organisation	Bundesland	Land	
User	01	laden ansetzen	2002-12-11 12:46	Daniel Hirscher	Daniel.Hirscher@innovations.de	Innoentstade	Innovations GmbH	Baden-Wuerttemberg	DE

Abbildung 3.8: Ergebnis der Suche.

Nachdem der Mitarbeiter sein öffentliches Zertifikat über die Zustellungs-E-Mail abgeholt hat, baut der Web-Browser das öffentliche Zertifikat und den privaten geheimen Schlüssel zum persönlichen Zertifikat zusammen. Jetzt ist es verwendbar.

In den E-Mail Programmen kann jetzt noch eingestellt werden, dass immer versucht werden soll zu unterschreiben und zu verschlüsseln. Alle Mitarbeiter wurden angewiesen, von ihrem Schlüsselpaar, d.h. privater Schlüssel und

öffentliches Zertifikat zusammen, eine Sicherungskopie anzufertigen. Diese sollte auf dem Netzwerklaufwerk abgelegt werden. Die Dateien sind Passwortgeschützt. Sie werden auf dem täglichen Backup gesichert.

Nachdem einige Zertifikate verteilt waren, tauchten eine ganze Menge Probleme auf, die nicht vorhersehbar waren. Sie konnten zum größten Teil gelöst werden. Da die Fehler sehr technisch sind befindet sich die Problemliste im Anhang [B.3](#).

Kapitel 4

Weitere Einsatzmöglichkeiten von Zertifikaten

Der Einsatz von X.509 Zertifikaten um den E-Mail Verkehr zu sichern, wurde in den bisherigen Kapiteln ausführlich beschrieben. Das zweitwichtigste Einsatzgebiet, Internetverbindungen zu Web-Servern mit SSL zu sichern, sowie das digitale Unterschreiben von Programmen, befindet sich hier.

4.1 Server mit SSL sichern

4.1.1 Extranet

Dem Server für das Extranet (`extranet.innovations.de`) wurde ein SSL Server Zertifikat ausgestellt. Damit ist es möglich über HTTPS (Hypertext Transfer Protocol Secured Socket Layer) eine verschlüsselte Verbindung aufzubauen. Der Client benötigt kein Zertifikat. Dessen Web-Browser handelt selbstständig mit dem Server einen geeigneten Schlüssel aus. Somit wird jetzt gewährleistet, dass die Passwörter für den Zugang der Kunden zum Extranet nicht mehr lesbar übertragen werden. Genauso sind die vom Extranet zum Kunde übertragenen Dateien verschlüsselt. Dabei ist es so, dass für jede Verbindung, also für jede Datei, jedes Bild, jede HTML Seite usw. ein eigener zufälliger Sitzungsschlüssel benutzt wird. Damit die Kunden von der Änderung nicht überrascht werden, gibt es dazu eine Einstiegsseite (Abbildung 4.1). Von hier aus können auch die öffentlichen CA Zertifikate in den Web-Browser installiert werden.

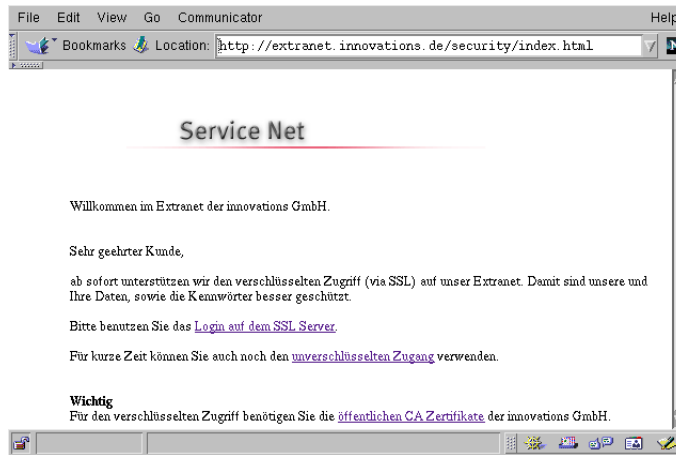


Abbildung 4.1: Einstiegsseite im Extranet.

4.1.2 Intranet

Zur zusätzlichen Sicherung der Zertifikatsanforderungsseiten im Intranet wurde auch dort ein SSL Server Zertifikat eingerichtet. Die Zertifikatsanträge können somit verschlüsselt erfolgen. Das sichert die übertragenen Antragsdaten (Daten + öffentlicher Schlüssel) zusätzlich gegen unbefugtes Abhören. Ein Antrag könnte jedoch auch sonst nicht gefälscht werden, weil die Antragsdaten mit dem privaten Schlüssel schon vor dem Absenden unterschrieben werden. Ein Betrug würde auffliegen.

4.2 Unterschreiben von Programmen

Es gibt die Möglichkeit, Code, d.h. Programme, zu unterschreiben. Damit ist es möglich, die Identität des Herstellers eindeutig festzustellen. Weiterhin läßt es sich erkennen, ob z.B. eine Veränderung durch Viren stattfand. Eine Besonderheit ist, dass man einem vertrauenswürdigen Programm gewisse Rechte gewähren kann. Dabei ist das Unterschreiben von Java Applets, die in einem Browser laufen, die wichtigste Anwendung. Diese Applets dürfen normalerweise die Sandbox¹ des Web-Browsers nicht verlassen. Wollen sie es doch tun, um z.B. eine Datei zu lesen oder sogar zu schreiben, so geht dies nur, wenn sie dazu die Erlaubnis erhalten und so die Sandbox verlassen können.

¹Ein geschützter Bereich innerhalb des Web-Browsers, in dem gefährliche Aktionen (z.B. Dateizugriffe) nicht möglich sind

Zu beachten ist, dass das Zertifikat, welches zum Zeitpunkt des Unterschreibens gültig war, zum Zeitpunkt des Benutzens abgelaufen oder zurückgezogen sein kann.

4.2.1 Netscape's Signtool

Netscape stellt unter dem Namen *Signtool* ein Programm zur Verfügung um JAR (Java Archive) Dateien zu unterschreiben. Der Vorteil ist, dass es auf verschiedenen Betriebssystemen erhältlich ist. Der Nachteil ist, dass damit unterschriebene Dateien nur mit Netscape Browser funktionieren. Das enthaltene Zertifikat durchläuft die üblichen Hierarchieprüfungen im Netscape Browser, um den Vertrauensstatus zu prüfen. Man muss jedoch in jedem Fall beim ersten Aufruf zustimmen, welche Aktionen das Java Programm außerhalb der Sandbox tätigen möchte.

4.2.2 JDK's Keytool

Im JDK (Java Developers Kit) von Sun befindet sich ab der Version 1.2 ein neugestaltetes Sicherheitsmanagement. Dazu gehören die Programme *Keytool*, *Policytool* und *Jarsigner* sowie die Dateien `.keystore` und eine Policy-Datei, z.B. `.policy`.

Das Keytool verwaltet die Schlüssel und greift auf den Keystore zu. Mit dem Policytool werden die Rechte der einzelnen Programme definiert. Hier kann z.B. eingestellt werden, dass das Programm `signedWriteFile.jar`, das von *Duke* unterschrieben wurde, im Verzeichnis `/tmp/foo` schreibend zugreifen darf (Beispiel in [Sec00]). Die Rechte können also sehr fein eingestellt werden. Schließlich kann man mit dem *Jarsigner* `.jar` Dateien unterschreiben.

Mit dem Keytool können keine Zertifikate mit privatem Schlüssel importiert werden. Es können ausschließlich öffentliche Zertifikate in den Keystore aufgenommen werden. Um ein komplettes Schlüsselpaar zu erhalten, muss es mit dem Keytool generiert werden (siehe dazu Abschnitt B.5). Wenn die eingegebenen Daten mit den Richtlinien der eigenen PKI übereinstimmen, kann der Zertifikatsantrag von ihr unterschrieben werden.

Bei der Schlüsselerzeugung innerhalb vom Keytool und dessen Ausstellung über die Zertifikatsanforderung bleibt das Problem des Exportierens des gesamten Zertifikates bestehen. Es kann mit dem Keytool keine Sicherungskopie eines einzelnen Schlüssels erstellt werden oder dieser in einen anderen Keystore übertragen werden.

4.2.3 Microsoft's Signcode

Microsoft bietet unter dem Namen *Authenticode* die Möglichkeit an, beliebige ausführbare Dateien und mit der WSH (Windows Scripting Host) Ergänzung auch Skripte zu unterschreiben. Dies funktioniert allerdings nur wenn alle Zertifikate in der Hierarchie die Erweiterung `extendedKeyUsage` mit dem Wert `codeSigning` (und nur mit diesem!) enthalten. Diese Anforderungen passen nicht in die Struktur der aufgebauten CA. Diese Möglichkeit kann nicht genutzt werden.

4.3 Virtuelles Privates Netz (VPN)

Mit den Benutzerzertifikaten der CA können auch verschlüsselte Netzwerkverbindungen realisiert werden, wenn z.B. Geschäftsreisende sich mit ihrem Notebook ins Firmennetzwerk einwählen wollen, oder eine Außenstelle angeschlossen werden soll. Beides trifft für die Firma Innovations GmbH zu.

Die Anforderungen an ein VPN (Virtual Private Network) sind hoch. *Es soll möglich sein, auf alles so zuzugreifen, als würde man sich in der Firma befinden.* Damit ist eine Realisierung mit SSH (Secure Shell) oder SSL nicht möglich, da sie zu nahe an der Anwendungsschicht arbeiten. Wegen der Firewall, die mit Linux betrieben wird, fallen Windows spezifische Lösungen ebenfalls weg. Es bleibt die Möglichkeit IPsec (Internet Protocol Security) zu verwenden. Dies kann Firewall seitig mit dem freien Produkt FreeS/WAN (Free Secure Wide Area Network) und auf der Clientseite z.B. mit PGPnet (PGP Netzwerk) aufgebaut werden. Läuft auf dem Client Windows 2000, dann kann auch dessen integrierte IPsec Schnittstelle benutzt werden. IPsec sichert die Integrität der Daten, garantiert die Herkunft und stellt einen vertrauenswürdigen Datenkanal zur Verfügung. Technische Einzelheiten dazu sind im Abschnitt [C.7.3](#) beschrieben. Das Thema VPN wurde untersucht aber nicht realisiert.

Kapitel 5

Zusammenfassung und Ausblick

Es wurde in dieser Arbeit untersucht, wie sich die Risiken bei der praktischen Nutzung von Internetdiensten mit Hilfe einer Public Key Infrastruktur effektiv verringern lassen.

Im Vordergrund steht die gesicherte Kommunikation per E-Mail. Denn es ist das am häufigsten genutzte Kommunikationsmittel. Ein zweiter wichtiger Bestandteil dieser Arbeit ist der verschlüsselte Zugang zu Web-Servern. Damit vertrauliche Daten oder Passwörter von Dritten auf der Übertragungsstrecke nicht abgehört werden können. Ein dritter Aspekt ist das Unterschreiben von Programmen, die anhand der Unterschrift bestimmte Aktionen durchführen können.

Zu Beginn der Arbeit musste ein geeignetes Public Key Verfahren ausgewählt werden. Dazu wurden erstens die verschiedenen erhältlichen Public Key Systeme verglichen um festzustellen, was die Programme leisten. Zweitens wurde eine Umfrage durchgeführt um festzustellen, welche E-Mail Programme bei der Firma Innovations GmbH verwendet werden. Drittens versuchte ich zu analysieren, welches Verfahren andere Firmen bevorzugen. Aufgrund dieser Ergebnisse wurden X.509 Zertifikate für den Aufbau der PKI gewählt. Daraufhin wurde nach geeigneten Programmen für den Aufbau der PKI gesucht. Dabei steckt das größte Wissen in der Konfigurationsdatei zu `openssl` und dem Aufbau der X.509 Zertifikate, damit diese auch mit allen verwendeten Programmen funktionieren. Nach einer Einführungsveranstaltung für die Mitarbeiter wurden die Zertifikate erstellt und ausgegeben. Es brach eine Flut von Problemen und Bugs in den Programmen auf mich herein, die aber größtenteils bewältigt wurden.

Die durch den Aufbau der PKI gewonnenen Sicherheitsvorteile sind zum einen die sichere E-Mail Kommunikation mit S/MIME, dies funktioniert sehr gut, viele Mitarbeiter benutzen es, und zum anderen wurden der Extranet Server und Teile des Intranet Servers mit SSL gegen das Abhören der Verbindung durch Verschlüsselung gesichert. Beim digitalen Unterschreiben von Programmen konnte nur das Keytool von Sun einigermaßen überzeugen. Alle drei Gebiete, S/MIME, SSL und Unterschreiben von Code, wurden ausgiebig daraufhin getestet, ob sie auch tatsächlich den versprochenen Schutz bieten. Es wurde absichtlich abgehört, um zu sehen, ob wirklich alles verschlüsselt ist. Es wurde absichtlich gefälscht, um zu sehen, ob die Fälschung erkannt wird oder die Daten unbrauchbar werden.

Für die CA an sich wurden alle notwendigen Verfahrensweisen und Voraussetzungen geschaffen. Es wurde eine geeignete Zertifizierungshierarchie festgelegt. Der Zertifizierungsablauf, die Zertifikatsbereitstellung, die Prüfung und der Widerruf wurden realisiert. Die Handhabung der Zertifikate muss sehr einfach sein, sonst besteht die Gefahr, dass es von den Benutzern nicht akzeptiert wird. Es wurden Schritt-für-Schritt Anleitungen im Intranet bereitgestellt, was bei den E-Mail Programmen eingestellt werden muss, um einen reibungslosen Betrieb zu garantieren.

Für die Zukunft sollte man nach und nach weitere Internet Dienste gegen unbefugtes Mitlesen sichern. Das nächste Ziel ist ein VPN zwischen zwei Firmenstandorten. Dabei sollte stets darauf geachtet werden, dass nur starke kryptographische Techniken mit ausreichender Schlüssellänge zum Einsatz kommen. Die kontinuierliche Schulung vor allem neuer Mitarbeiter darf nicht vernachlässigt werden. Als weiteres Ziel kann man Chipkarten (o.ä. Systeme) oder gar biometrische Systeme zur Authentifizierung ansehen.

Anhang A

Zeitleiste

In den sechs Monaten meiner Diplomarbeit (vom 01.09.2000 – 28.02.2001) traten einige Ereignisse ein, die gut zum Thema passen. Die Veränderungen waren sehr interessant, deshalb entschloss ich mich, diese in zeitlicher Reihenfolge aufzuführen.

Datum	Ereignis
02. September 2000	Die Kommandozeilenversion von PGP-6.5.8 ist erhältlich.
06. September 2000	Der RSA Algorithmus wird vorzeitig freigegeben.
10. September 2000	Die kommerzielle Version von PGP-7 wird herausgegeben.
18. September 2000	Das freie GnuPG 1.0.3 unterstützt RSA.
20. September 2000	Das Patent auf den RSA Algorithmus läuft ab.
24. September 2000	Der Sourcecode von PGP-6.5.8 wird freigegeben.
01. Oktober 2000	In den USA tritt das „Millennium Digital Commerce Act“ in kraft. Er ermöglicht nahezu alle Geschäfte mittels digitalen Unterschriften rechtswirksam über das Internet abzuwickeln.
02. Oktober 2000	Das NIST (National Institute of Standards and Technology) hat den Verschlüsselungsalgorithmus Rijndael als Vorschlag für den AES (Advanced Encryption Standard) gewählt.
17. Oktober 2000	Das freie GnuPG 1.0.4 unterstützt Rijndael.
15. Februar 2001	Der Bundestag beschließt ein Gesetz zur digitalen Signatur, das die entsprechende EU-Richtlinie in deutsches Recht umsetzt.

Anhang B

Technische Anleitung zum Aufbau der CA

B.1 Installation

Es wurden immer die neuesten Versionen installiert, weil durch das Wegfallen der Exportbeschränkung in den USA Schlüssel höherer Stärke möglich sind. Das zog nach sich, dass andere Programme auch auf die neuen Versionen bestehen.

B.1.1 OpenSSL

OpenSSL stellt alle Funktionen zur Erzeugung und zum Rückzug von X.509 Zertifikaten zur Verfügung.

Zu finden bei <http://www.openssl.org>.

Benutzte Version: OpenSSL 0.9.6 vom 24 Sep. 2000.

Kompilieren und installieren:

```
> ./config --prefix=/usr
> make
> make test
> make install
```

B.1.2 pyCA

PyCA stellt ein Web-Interface zur Verfügung, über das X.509 Zertifikate beantragt, betrachtet und in die Web-Browser installiert werden können.

Zu finden bei <http://www.pyca.de>.

Benutzte Version: pyca 0.6.5.

PyCA benötigt OpenSSL, OpenLDAP ist optional.

Die Skripte sind in Python geschrieben. Sie müssen nur an die richtige Stelle kopiert werden.

Übersicht über die Skripte

cgi-bin/ ca-index.py Zeigt die CA Zertifikate. Von hier können die Zertifikate der Root CA und der Zwischenzertifizierungsstellen, sowie die jeweilige CRL in den Browser installiert oder angesehen werden.

client-enroll.py Generiert eine Zertifikatsanforderung in drei Schritten.

cert-query.py Damit kann nach Zertifikaten gesucht und das Ergebnis angezeigt werden.

view-cert.py Skript zur Anzeige eines Zertifikates.

get-cert.py Lädt ein Zertifikat in den Browser.

ns-check-rev.py Verifiziert ein Zertifikat.

ns-revoke.py Ruft ein Zertifikat zurück.

browser-check.py Testet die Fähigkeiten des Web-Browsers von SSL und der Schlüsselerzeugung. In `/etc/httpd/httpd.conf` muss für das `cgi-bin` Verzeichnis folgendes angegeben werden: `SSLOptions +StdEnvVars +CompatEnvVars`.

sbin/ ca-make.py Generiert die CA Hierarchie, alle benötigten Dateien und Verzeichnisse und die CA Zertifikate. Es wird nur einmal von `root` aufgerufen.

ca-certreq-mail.py Dem Skript wird die E-Mail über einen Eintrag in der Datei `.forward` zugeschickt. Es verschiebt dann die bestätigte Zertifikatsanforderung zu den zu unterschreibenden Anforderungen.

ca-cycle-pub.py Wird als cron-Job aufgerufen.

- Publiziert neue Zertifikate und informiert die Benutzer über E-Mail, wo sie ihre Zertifikate herunterladen können.
- Entfernt nach 24 Stunden unbearbeitete Zertifikate aus dem Verzeichnis `pendreqs`.

ca-cycle-priv.py Dieses Skript läuft auf dem System, das die privaten Schlüssel enthält. Es erzeugt die CRLs. Es kann auch als `cron`-Job aufgerufen werden.

- Markiert abgelaufene Zertifikate in der Datenbank.

ca-revoke.py Zum Rückruf von verlorenen, gelöschten oder kompromittierten Schlüsseln. Die neue CRL wird auf Wunsch gleich mit-erstellt. Sonst sollte man `ca-cycle-priv.py` aufrufen.

bin/ ca2ldif.py Erzeugt aus CA Zertifikaten und CRLs eine LDIF (LDAP Data Interchange Format) Datei. Sollte nur einmalig ausgeführt werden.

certs2ldap.py Sendet alle Zertifikate und CRLs an den LDAP (Lightweight Directory Access Protocol) Server.

copy-cacerts.py Kopiert alle CA Zertifikate zusammen in eine PEM Datei oder ein Verzeichnis mit symbolischen Links der Hash-Werte der Zertifikate. Dies ist nötig für SSL-Verbindungen mit Apache.

ns-jsconfig.py Erzeugt Javascript Code zur Benutzung mit dem Netscape Admin Tool (`netscape.cfg`).

print-ca-certs.py Zeigt alle Zertifikate in der Konsole an, um z.B. authentische Papierausdrucke machen zu können. Es kann mit der Option `--html` formatierte HTML Dateien erzeugen.

Beschreibung der Verzeichnisse und Dateien

Im Homeverzeichnis des Users `caadmin` befinden sich unterhalb des Verzeichnisses `ca` die Root CA und die Zwischenzertifizierungsstellen: `User`, `Object` und `Server`. Unterhalb dieser ist der Aufbau immer der gleiche (außer `Root`):

pendreqs Hier liegen die unbestätigten Zertifikatsanträge. Es ist der erste Schritt zum Zertifikat. Wird der Antrag nicht per E-Mail bestätigt, wird er nach 24 Stunden gelöscht. Bei einer Bestätigung wird er in das Verzeichnis `newreqs` verschoben.

newreqs Hier liegen die bestätigten Zertifikatsanträge. Diese müssen auf den netzwerklosen Computer übertragen werden. Nach dem Prüfen und Unterschreiben befinden sich die zuzustellenden Zertifikate im Verzeichnis `newcerts`.

oldreqs Hier können die verarbeiteten Zertifikatsanträge gelagert werden.

newcerts Hier befinden sich die noch nicht zugestellten Zertifikate. Nach der Zustellung, dem Losschicken der E-Mail, werden sie ins Verzeichnis **certs** verschoben.

certs Hier liegen alle öffentlichen Zertifikate.

crl Hier liegen alte CRL Dateien. Die aktuelle CRL liegt im übergeordneten Verzeichnis und heißt **crl.pem**.

private Hier befinden sich auf dem netzwerklosen Computer die privaten Schlüssel der jeweiligen Zertifizierungsstelle. Auf dem Server ist das Verzeichnis leer.

cacert.pem Das öffentliche Zertifikat der Zertifizierungsstelle.

crl.pem Die Rückrufliste (CRL) die diese Zertifizierungsstelle ausgestellt hat.

index.txt Die Datenbank aller von der Zertifizierungsstelle ausgegebenen Zertifikate (auch die zurückgezogenen) mit Status (V=Valid, R=Revoked), Datum, Seriennummer und dem vollständigen DN (Distinguished Name).

serial Diese Datei enthält die nächste zu vergebende Seriennummer.

B.2 Struktur der CA

B.2.1 PKI Definitionen

LDAP Feldbezeichnungen

Die Attribute und ihre Werte für X.509 Zertifikate sollten einheitlich sein, damit sich die Zertifikate nahtlos in einen späteren Verzeichnisdienst einfügen. Die für die Firma Innovations festgelegten Attribute für X.509 sind in Tabelle [B.1](#) definiert.

Erweiterungen als kritisch markieren

Jede X.509v3 Erweiterung kann als *kritisch* markiert werden. Ein Programm welches eine als kritisch markierte Erweiterung nicht kennt, sollte das Zertifikat nicht benutzen. Ein nicht als kritisch markiertes Feld kann beachtet werden, muss aber nicht.

Attribut	Abk.	Wert
CountryName	C	DE
StateOrProvinceName	ST	Baden-Wuerttemberg
LocalityName	L	Immenstaad
OrganizationName	O	Innovations GmbH

Tabelle B.1: LDAP: Feste Attribute für Innovations

Weil es durch diese Markierung oft zu Problemen kommt (Hauptsächlich zwischen Netscape und Microsoft) wurde es bei allen Erweiterungen weggelassen, d.h. alles ist als `non critical` markiert.

Der Vorteil ist, dass die Zertifikate von allen Programmen anerkannt werden. Der Nachteil ist, dass z.B. bei Windows mehr Wert auf die `extendedKeyUsage` gelegt wird, als auf die ursprüngliche `keyUsage`.

Erweiterung: Basiseinschränkung und `keyUsage`

Die Basiseinschränkungen und die `keyUsage` für die einzelnen Zertifikate sind in Tabelle B.2 aufgelistet. Diese zwei Erweiterungen sollten mindestens gesetzt werden, um die Zertifikatsbenutzung einzuschränken. Dadurch macht die Aufteilung in Zwischenzertifizierungsstellen erst einen Sinn.

Zertifikat	basicConstraints	keyUsage
Root CA	CA:true	cRLSign, keyCertSign
User CA	CA:true	cRLSign, keyCertSign
Object CA	CA:true	cRLSign, keyCertSign
Server CA	CA:true	cRLSign, keyCertSign
User	CA:false	nonRepudiation, digitalSignature, keyEncipherment
Object	CA:false	nonRepudiation, digitalSignature
Server	CA:false	

Tabelle B.2: Definition Zertifikate

Erweiterung: `extendedKeyUsage`

Mit der Erweiterung `extendedKeyUsage` können die Einschränkungen eines Zertifikates feiner bestimmt werden. Es kann anstelle von `keyUsage` oder

als Ergänzung verwendet werden. Auf die `extendedKeyUsage` wurde wegen Microsofts Forderung, dass, wenn die Erweiterung gesetzt wird, sie in allen Zertifikaten der Kette enthalten sein muss, verzichtet (auch in [CKLW00, S. 145] empfohlen). Der Sinn der Forderung ist fraglich. Warum soll man in einem CA Zertifikat z.B. `emailProtection` setzen, wofür es gar nicht gedacht ist, nur damit die davon abhängigen Zertifikate gültig sind?

Netscape Erweiterungen

Zusätzlich enthalten die Zertifikate noch einige Erweiterungen, die von Netscape eingeführt wurden. Darunter ist ein Kommentar zur Erklärung für den Benutzer, wofür das Zertifikat ist. Verschiedene URL (Uniform Resource Locator) Einträge zu Skripten, um z.B. die CRL online zu überprüfen, das Zertifikat zurückzurufen oder die Policy anzusehen, sowie Netscapes eigene Definition zur Schlüsselbenutzung sind dort eingetragen.

Schlüssellänge

Die Schlüssellänge für den asymmetrischen RSA Schlüssel beträgt immer 1024-Bit. Schlüssellängen kleiner als 1024-Bit gelten als unsicher [Bun99]. Sie werden von der CA nicht zugelassen. Längere Schlüssel können in den Programmen Probleme bereiten. In der RFC 2633 sind diese mit SHOULD aufgeführt [WL].

B.2.2 Zertifikatspeicherung

Die Zertifikate werden in den verschiedenen E-Mail Programmen unterschiedlich gespeichert und verwaltet.

Netscape

Die öffentlichen Zertifikate befinden sich in der Datei `cert7.db`. Die privaten Schlüssel befinden sich in `key3.db`. Sie werden von Netscape mit dem Passwort der Zertifikatsdatenbank verschlüsselt.

Windows

Outlook und Outlook Express benutzen die Zertifikatsverwaltung von Windows. Die Zertifikate sind in der Registry (Datei: `NTUSER.DAT`) gespeichert. Sie befinden sich dort im Pfad:

```
CURRENT_USER\Software\Microsoft\SystemCertificates\my\Certificates
```

Dabei sind die öffentlichen Zertifikate im DER-Format gespeichert. Es ist aber ein unbekannter 72-Byte langer Header vorangestellt. Den Speicherort des privaten Schlüssels konnte ich in Windows nicht finden.

TheBat!

Das private Zertifikat im PFX (Personal Information Exchange) Format befindet sich im Pfad:

C:\Programme\The Bat!\MAIL\accountname\ACCOUNT.PFX. Die Datei kann mit `openssl` nur gelesen werden wenn der MAC (Message Authentication Code) nicht überprüft wird.

```
openssl pkcs12 -nomacver -in ACCOUNT.PFX -out account.pem
```

Die Schlüssel sind mit dem Export-Passwort verschlüsselt.

Zertifikate im Netzwerk

Es ist zu beachten, dass private Schlüssel unter Umständen im regelmäßigen Firmenbackup gespeichert sind. Dies ist dann der Fall, wenn Outlook oder Outlook Express verwendet wird (Profil), oder wenn die Installation des Programmes auf einem Netzwerklaufwerk statt lokal erfolgte. Die Backupmedien müssen so gesichert sein, dass niemand an sie herankommt. Bei der Firma Innovations GmbH befinden sich diese im Serverraum, der immer abgeschlossen ist und zusätzlich in einem Bankschließfach.

B.2.3 Zertifizierungsablauf

Der Zertifizierungsablauf ist in Abbildung [B.1](#) dargestellt. Die einzelnen Schritte sind hier erklärt.

1. Der Antragsteller wählt im Intranet über eine Einstiegsseite die URL des gewünschten Antragsformulars aus (User, Object oder Server Zertifikat). Eventuell wird er gleich zu einem bestimmten Formular geleitet.
2. Die Seite enthält noch keine Parameter. Somit wird ein leeres Formular zurückgeschickt.

3. Der Antragsteller füllt das Formular aus. Hierbei sind die laut Policy erforderlichen (match) Parameter schon fest vorgegeben. Standardwerte sind in die Textfelder schon eingetragen.
4. Die Parameter werden geprüft. Sollte noch einer fehlen, so wird die Antragsseite erneut geschickt. War alles ok, dann wird das Formular zur Erzeugung der Schlüssel dargestellt. Die Schlüsselerzeugung erfolgt in Netscape über das <KEYGEN> Tag, im Internet Explorer mit VB (Visual Basic) Script.
5. Der Web-Browser des Antragstellers erzeugt das Schlüsselpaar. Daraufhin behält der Web-Browser den privaten Schlüssel (und den öffentlichen natürlich auch) bei sich und schickt den öffentlichen Schlüssel zusammen mit den digital unterschriebenen Antragsdaten zum Server. Unterschrieben wird mit dem eben erstellten Schlüsselpaar zum Schutz gegen einen Man-in-the-Middle Angriff. Netscape verwendet dabei das SPKAC (Signed Public Key and Challenge) Format. Der Internet Explorer verwendet PKCS#10 (Certification Request Syntax Standard).
6. Die übermittelten Daten und Schlüssel werden nochmals geprüft und dann gespeichert. Es wird eine E-Mail an den Antragsteller verschickt um zu prüfen, ob die angegebene E-Mail Adresse gültig ist. Dies verhindert auch, dass Fremde ein Zertifikat unter einem anderen Namen erstellen können.
7. Der E-Mail Empfänger muss nämlich entscheiden, ob er es wirklich war, der gerade eben ein Zertifikat beantragt hat. Er schickt die E-Mail deshalb zur Bestätigung zurück.
8. Anhand des Betreffs, der einen Zufallscode enthält, wird die E-Mail dem Zertifikatsantrag zugeordnet. Der Antrag wird dann zur Weiterbearbeitung durch die CA bereitgestellt.
9. Die CA befördert den Antrag auf eine Diskette und zum netzwerklosen Computer, auf dem die Schlüssel der Zwischenzertifizierungsstellen liegen. Hier werden noch einmal die Daten und speziell das Einmalpasswort überprüft. Danach wird der Antrag von der entsprechenden Zwischenzertifizierungsstelle unterschrieben.
10. Das fertige Zertifikat wird wieder per Diskette auf den Server gebracht.
11. Ein auf dem Server laufender Cron-Job stellt dem Antragsteller das neue Zertifikat per E-Mail zu.

12. Diese E-Mail enthält einen URL, mit dem das Zertifikat abgeholt werden kann. Der Antragsteller klickt diesen an.
13. Der Server bereitet das Zertifikat für die Web-Browser Zustellung auf und schickt es dem Antragsteller; bei Netscape mit dem MIME-Typ `application/x-x509-user-cert`, beim Internet Explorer mit einer Seite, die VB Script enthält.

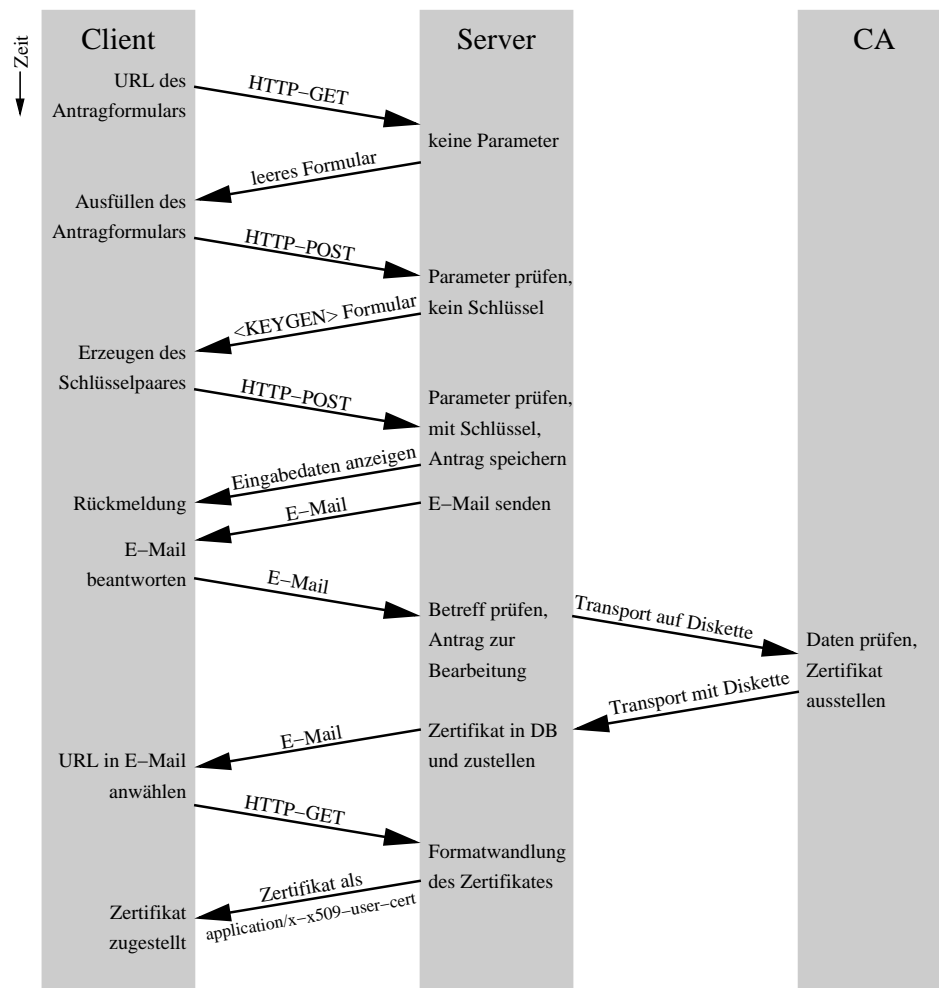


Abbildung B.1: Zeitlicher Ablauf der Zertifizierung.

B.2.4 Benutzerzertifikate unterschreiben

Beim Einloggen auf den netzwerklosen Computer erscheint zusammen mit der Uhrzeit die Meldung, dass die Uhrzeit und das Datum überprüft werden soll. Da beim Unterschreiben kein offizieller Zeitstempeldienst herangezogen werden kann, sollte zumindest die eingebaute Uhr die richtige Zeit anzeigen. Es kommt dabei nicht auf Sekunden, wohl aber auf Minuten, an.

Die Zertifikatsanträge können jetzt von der Diskette in das Verzeichnis `newreqs` der entsprechenden Zertifizierungsstelle kopiert werden.

Unterschreiben der Zertifikatsanträge, die mit Netscape erstellt wurden (SPKAC Format):

```
> openssl ca -name CA_User -spkac <filename>
```

Unterschreiben der Zertifikatsanträge, die mit Microsofts Internet Explorer erstellt wurden (PKCS#10 Format, Endung `.pem`):

```
> openssl ca -name CA_User -in <filename>
```

Exportieren der unterschriebenen Zertifikate ohne die privaten Schlüssel der Zertifizierungsstelle unter Erhaltung der Dateirechte mit:

```
> tar cvzps --exclude cakey.pem -f ca-export.tar.gz ca/
```

zur Vereinfachung existiert ein Alias:

```
> caexport
```

Hinweis: Es sollte darauf geachtet werden, dass auf dem netzwerklosen Computer

vor dem Exportieren alle Zertifikatsanträge im Verzeichnis `newreqs` gelöscht oder zumindest ins Verzeichnis `oldreqs` verschoben werden.

nach dem Exportieren die Zertifikate im Verzeichnis `newcerts` gelöscht oder zumindest ins Verzeichnis `certs` verschoben werden.

Ansonsten bekommt man Warmeldungen vom Script `ca-cycle-pub.py`, wenn zugestellte Zertifikate wiederholt zugestellt werden oder Zertifikatsanträge zweimal bearbeitet werden.

Importieren der unterschriebenen Zertifikate auf den Server `proxy2` mit:

```
> tar xvzpsf ca-export.tar.gz .
```

zur Vereinfachung existiert ein Alias:

```
> caimport
```

Hinweis: Die Zertifikatsanträge (Requests) sollte man nicht löschen, sondern zusammen mit den `.ldif` Dateien in das dafür vorgesehene Verzeichnis `oldreqs` verschieben. Mit diesen Anträgen kann man, sofern man dies will, die Zertifikate nach dem Ablauf erneuern. Sie enthalten dann nämlich den gleichen Schlüssel. Aus Sicherheitsgründen wird man jedoch das Schlüssel-paar erneuern wollen. Entweder weil der private Schlüssel vielleicht doch heimlich geklaut wurde oder einfach um die Schlüssellänge auf z.B. 2048-Bit zu erhöhen, damit wieder zeitgemäßer Schutz gegeben ist. Die `.ldif` Dateien können in ein Adressbuch eines LDAP Servers importiert werden.

B.2.5 Zertifikat zurückrufen

Wenn ein Zertifikat verloren, das Passwort zum Entschlüsseln vergessen oder geklaut wurde, dann muss es zurückgerufen werden. Für einen Rückruf benötigt man die Seriennummer (hexadezimal) des betroffenen Zertifikates. Dies ist über die Datenbank, bzw. die Suche nach Zertifikaten, in Erfahrung zu bringen.

Auf dem netzwerklosen Rechner, auf dem die Schlüssel der Zwischenzertifizierungsstellen liegen, ruft man folgenden Befehl auf:

```
ca-revoke.py --pycalib /home/caadmin/pylib --name User --serial 42
```

Bei `--name` muss die Zertifizierungsstelle stehen. Die Seriennummer muss hexadezimal angegeben werden. In der Regel genau so, wie es im Zertifikat und in der Datenbank erscheint.

Daraufhin muss die Passphrase der entsprechenden Zwischenzertifizierungsstelle eingegeben werden. Es erscheinen die Daten des Zertifikates. Wenn es sich um das richtige handelt, muss man dies bestätigen und kann auch gleich eine neue CRL ausstellen lassen, was sehr sinnvoll ist. Danach sollte man mit `caexport` alles packen und per Diskette auf den Server bringen. Nach dem Entpacken mit `caimport` stehen die CRLs sofort zur Verfügung.

Der Hauptgrund für ein Verfallsdatum in den Zertifikaten ist, dass die CRL klein gehalten werden kann. Ein zurückgerufenes Zertifikat kann aus der CRL entfernt werden, wenn es abgelaufen ist.

B.2.6 Konfigurationsdateien

Die OpenSSL Konfigurationsdatei heißt `openssl.cnf`. Die Scriptsammlung pyCA führt in dieser Datei eigene Einträge auf (ab Zeile 286). Da die Zertifikatserweiterungen für die CA Zertifikate anders lauten müssen als die Endbenutzerzertifikate, gibt es für die Root CA und die Zwischenzertifizierungsstellen eigene Konfigurationsdateien. Diese beginnen mit „`ext_`“.

In diesen Konfigurationsdateien steckt sehr viel Arbeit. Sie musste so gestaltet werden, dass die Zertifikate von Netscape und Microsoft akzeptiert werden. Dazu musste auch jeder Parameter verstanden werden. Das hat viel Zeit gekostet.

openssl.cnf

```
# OpenSSL configuration file: Dreistufige CA-Hierarchie
# Geändert: 27.12.2000 - falsche URLs verbessert

5  RANDFILE           = "$ENV::HOME/.rnd"
   oid_file           = /etc/openssl/.oid
   oid_section        = new_oids

[ new_oids ]
10  dnQualifier       = 2.5.4.46
   surName            = 2.5.4.4
   givenName          = 2.5.4.42
   initials           = 2.5.4.43
   generationQualifier = 2.5.4.44

15  #####
   [ ca ]
   Root               = CA_Root       # oberste CA
   User                = CA_User       # fuer E-Mail und SSL Client
20  EmailCerts        = CA_User       # Alias wegen Zerts mit falscher URL
   Object             = CA_Object      # Code unterschreiben
   Server              = CA_Server     # SSL Server

#####
25  [ CA_Root ]
   dir                 = /home/caadmin/ca/Root
   certs               = $dir/certs    # ausgestellte Zerts
   crl_dir             = $dir/crl      # ausgestellte CRLs
   database            = $dir/index.txt # Datenbank
30  new_certs_dir     = $dir/newcerts  # neue Zerts
   pend_reqs_dir      = ""            # nicht Zutreffend
   new_reqs_dir       = ""            # nicht Zutreffend
   certificate        = $dir/cacert.pem # CA Zert
   serial              = $dir/serial    # aktuelle Seriennummer
35  crl                = $dir/crl.pem   # aktuelle CRL
   private_key        = $dir/private/cakey.pem # privater CA Schluessel
   RANDFILE           = $dir/private/.rand # Zufallszahl
   default_days       = 730            # Gueltigkeitsdauer (*)
   default_crl_days  = 365            # ?
40  default_md        = md5            # welcher Message Digest
   preserve           = no            #
   policy             = policy_Root   # Richtlinien
   ca_x509_extfile    = /etc/openssl/ext_rootCA.cnf # Erweiterungen
50  x509_extensions  = x509v3_ext_Root # Erweiterungen

45  [ CA_User ]
```

```

dir           = /home/caadmin/ca/User
certs        = $dir/certs           # ausgestellte Zerts
crl_dir      = $dir/crl             # ausgestellte CRLs
50 database   = $dir/index.txt      # Datenbank
new_certs_dir = $dir/newcerts      # neue Zerts
pend_reqs_dir = $dir/pendreqs      # Antraege, nicht bestaetigt
new_reqs_dir = $dir/newreqs        # Antraege
certificate   = $dir/cacert.pem     # CA Zert
55 serial     = $dir/serial          # aktuelle Seriennummer
crl          = $dir/crl.pem         # aktuelle CRL
private_key   = $dir/private/akey.pem # privater CA Schluessel
RANDFILE     = $dir/private/.rand   # Zufallszahl
default_days  = 730                 # Gueltigkeitsdauer
60 default_crl_days = 67            #
default_md    = md5                 # welcher Message Digest
preserve     = no                   #
policy       = policy_User          # Richtlinien
x509_extensions = x509v3_ext_User   # Erweiterungen
65 signedby   = Root                # Aussteller
ca_x509_extfile = /etc/openssl/ext_userCA.cnf # Erweiterungen
req          = req_User             # Anforderungstemplate
min_key_size  = 1024                # minimale Schluessellaenge

70 [ CA_Object ]
dir           = /home/caadmin/ca/Object
certs        = $dir/certs           # ausgestellte Zerts
crl_dir      = $dir/crl             # ausgestellte CRLs
75 database   = $dir/index.txt      # Datenbank
new_certs_dir = $dir/newcerts      # neue Zerts
pend_reqs_dir = $dir/pendreqs      # Antraege, nicht bestaetigt
new_reqs_dir = $dir/newreqs        # Antraege
certificate   = $dir/cacert.pem     # CA Zert
80 serial     = $dir/serial          # aktuelle Seriennummer
crl          = $dir/crl.pem         # aktuelle CRL
private_key   = $dir/private/akey.pem # privater Schluessel
RANDFILE     = $dir/private/.rand   # Zufallszahl
default_days  = 730                 # Gueltigkeitsdauer
85 default_crl_days = 67            #
default_md    = md5                 # welcher Message Digest
preserve     = no                   #
policy       = policy_Object        # Richtlinien
x509_extensions = x509v3_ext_Object  # Erweiterungen
90 signedby   = Root                # Aussteller
ca_x509_extfile = /etc/openssl/ext_objectCA.cnf # Erweiterungen
req          = req_User             # Anforderungstemplate

[ CA_Server ]
95 dir           = /home/caadmin/ca/Server
certs        = $dir/certs           # ausgestellte Zerts
crl_dir      = $dir/crl             # ausgestellte CRLs
database     = $dir/index.txt      # Datenbank
new_certs_dir = $dir/newcerts      # neue Zerts
pend_reqs_dir = $dir/pendreqs      # Antraege, nicht bestaetigt
100 new_reqs_dir = $dir/newreqs      # Antraege
certificate   = $dir/cacert.pem     # CA Zert
serial       = $dir/serial          # aktuelle Seriennummer
crl          = $dir/crl.pem         # aktuelle CRL
private_key   = $dir/private/akey.pem # privater Schluessel
105 RANDFILE     = $dir/private/.rand # Zufallszahl
default_days  = 730                 # Gueltigkeitsdauer
default_crl_days = 67              #
default_md    = md5                 # welcher Message Digest
preserve     = no                   #
110 policy       = policy_Server     # Richtlinien
x509_extensions = x509v3_ext_Server  # Erweiterungen
signedby     = Root                # Aussteller
ca_x509_extfile = /etc/openssl/ext_serverCA.cnf # Erweiterungen

115 ##### Policies #####

```

```

[ policy_User ]
countryName           = supplied
stateOrProvinceName  = optional
localityName         = optional
120 organizationName    = supplied
organizationalUnitName = optional
commonName           = supplied
initials             = optional
emailAddress         = supplied

125 [ policy_Object ]
countryName           = match
stateOrProvinceName  = match
localityName         = match
130 organizationName    = match
organizationalUnitName = optional
commonName           = supplied
initials             = optional
emailAddress         = supplied

135 [ policy_Server ]
countryName           = match
stateOrProvinceName  = match
localityName         = match
140 organizationName    = match
organizationalUnitName = optional
commonName           = supplied
emailAddress         = supplied

145 [ policy_Root ]
countryName           = match
stateOrProvinceName  = match
localityName         = match
organizationName     = match
150 organizationalUnitName = optional
commonName           = supplied
emailAddress         = supplied

#####
155 [ req ]
default_bits          = 1024
default_keyfile       = privkey.pem
distinguished_name    = req_distinguished_name

160 [ req_distinguished_name ]
countryName           = Landescode
countryName_default  = DE
countryName_min       = 2
countryName_max       = 2
165 countryName_regex  = "[a-zA-Z][a-zA-Z]"

stateOrProvinceName  = Bundesland
stateOrProvinceName_default = "Baden-Wuerttemberg"

170 localityName        = Stadt
localityName_default  = Immenstaad

organizationName     = Organisation
organizationName_default = "Innovations GmbH"

175 #organizationalUnitName = Organizational Unit Name
#organizationalUnitName_default = Department One,Department Two

commonName           = "Name (Vorname Nachname)"
180 commonName_max     = 64

emailAddress         = "E-Mail Adresse"
emailAddress_default = @innovations.de
emailAddress_max     = 64

```

```

185 emailAddress_regex      = "^( [\w@.=/_ - ]+ )@( [\w- ]+ )(\. [ \w- ]+ )*$"

[ req_User ]
distinguished_name        = req_distinguished_name_User

190 [ req_distinguished_name_User ]
countryName               = Landescode
countryName_default      = DE
countryName_min          = 2
countryName_max          = 2
195 countryName_regex      = "[a-zA-Z][a-zA-Z]"

stateOrProvinceName      = Bundesland
stateOrProvinceName_default = "Baden-Wuerttemberg"

200 localityName           = Stadt
localityName_default     = Immenstaad

organizationName          = Organisation
organizationName_default = "Innovations GmbH"
205 #organizationalUnitName = Organizational Unit Name
#organizationalUnitName_default = Department One,Department Two

commonName                = "Name (Vorname Nachname)"
210 commonName_max        = 64

emailAddress              = "E-Mail Adresse"
emailAddress_default      = @innovations.de
emailAddress_max          = 64
215 emailAddress_regex    = "^( [\w@.=/_ - ]+ )@( [\w- ]+ )(\. [ \w- ]+ )*$"

[ req_short_and_empty ]
distinguished_name        = req_distinguished_name_short_and_empty

220 [ req_distinguished_name_short_and_empty ]
countryName               = Landescode
countryName_min          = 2
countryName_max          = 2
countryName_regex        = "[a-zA-Z][a-zA-Z]"
225 stateOrProvinceName   = Bundesland

localityName              = Stadt

230 organizationName      = Organisation

organizationalUnitName    = Organizational Unit Name

commonName                = "Name (Vorname Nachname)"
235 commonName_max        = 64

emailAddress              = "E-Mail Adresse"
emailAddress_max          = 64
240 emailAddress_regex    = "^( [\w@.=/_ - ]+ )@( [\w- ]+ )(\. [ \w- ]+ )*$"

#####
[ x509v3_ext_Root ]
basicConstraints          = CA:true
nsComment = "Die Root CA gibt ausschliesslich Unter-CA Zertifikate aus."
245 nsBaseUrl              = "http://intranet.innovations.de/"
nsCaRevocationUrl        = cgi-bin/pyca/get-cert.py/Root/crl
nsRevocationUrl          = cgi-bin/pyca/ns-check-rev.py/Root?
nsRenewalUrl             = cgi-bin/pyca/ns-renewal.py/Root?
nsCaPolicyUrl            = security/policy/ca.html
250

# hier folgen die End-User-Extensions, die CA Extenstions sind in
# den extra Dateien
# wie Verisign: basicConstraints: non critical, CA false

```

```

255 [ x509v3_ext_User ]
    basicConstraints           = CA:false
    keyUsage = nonRepudiation, digitalSignature, keyEncipherment
    nsComment = "Dieses Zertifikat wird fuer E-Mail und als SSL Client benutzt."
    nsBaseUrl                = "http://intranet.innovations.de/"
260 nsCaRevocationUrl         = cgi-bin/pyca/get-cert.py/User/crl
    nsRevocationUrl          = cgi-bin/pyca/ns-check-rev.py/User?
    nsRenewalUrl             = cgi-bin/pyca/ns-renewal.py/User?
    nsCaPolicyUrl            = security/policy/user.html
    nsCertType               = email, client
265
    [ x509v3_ext_Object ]
    basicConstraints         = CA:false
    keyUsage                 = nonRepudiation, digitalSignature
    nsComment                = "Dieses Zertifikat unterschreibt Objekte."
270 nsBaseUrl                = "http://intranet.innovations.de/"
    nsCaRevocationUrl        = cgi-bin/pyca/get-cert.py/Object/crl
    nsRevocationUrl          = cgi-bin/pyca/ns-check-rev.py/Object?
    nsRenewalUrl             = cgi-bin/pyca/ns-renewal.py/Object?
    nsCaPolicyUrl            = security/policy/object.html
275 nsCertType               = objsign

    [ x509v3_ext_Server ]
    basicConstraints         = CA:false
    nsComment                = "Dies ist ein SSL Server Zertifikat."
280 nsBaseUrl                = "http://intranet.innovations.de/"
    nsCaRevocationUrl        = cgi-bin/pyca/get-cert.py/Server/crl
    nsRevocationUrl          = cgi-bin/pyca/ns-check-rev.py/Server?
    nsRenewalUrl             = cgi-bin/pyca/ns-renewal.py/Server?
    nsCaPolicyUrl            = security/policy/server.html
285 nsCertType               = server

#####
[ pyca ]

290 caCertFormat = DER

    nsBaseUrl = "http://intranet.innovations.de/" # Basis URL
    nsCAIndexUrl = cgi-bin/pyca/ca-index.py # ca-index.py
    nsEnrollUrl = cgi-bin/pyca/client-enroll.py # client-enroll.py
295 nsGetCertUrl = cgi-bin/pyca/get-cert.py # get-cert.py
    mailGetCertUrl = cgi-bin/pyca/get-cert-mail.py # get-cert-mail.py
    # wegen IE probleme
    nsViewCertUrl = cgi-bin/pyca/view-cert.py # view-cert.py
    HelpUrl = security/help/ # Hilfeseiten
300
    OpenSSEExec = /usr/bin/openssl # absoluter Pfad zum Programm

    userCAAdmin = caadmin # caadmin User
    userWWWRun = wwwrun # WWW User
305 userMailDaemon = caadmin # SMTP User
    ScriptMethod = POST # GET oder POST

    MailRelay = smtp # Mail Host
    TmpDir = /var/tmp # temporaere Dateien
310 # ErrorLog = /var/log/pyca-errors # Error Logfile
    caCertReqMailAdr = caadmin@innovations.de # E-Mail Check Adresse
    caPendCertReqValid = 24 # Verfallszeit (h) unbestaetigter Antraege

    # Unterscheidung Intern/Extern, wird nicht verwendet
315 caInternalCertTypes = keine
    caInternalIPAdr = 127.0.0.0/255.0.0.0,10.0.0.0/255.0.0.0
    caInternalDomains = innovations.de

    # Einfache HTML body tag steuerung (? statt ")
320 htmlBodyParam` =TEXT="#000000" LINK="#000000" VLINK="#000000"
    ALINK="#000000" BGCOLOR="#FFFFFF`"

```

ext_rootCA.cnf

```
# X.509v3 Erweiterungen fuer Root CA Zertifikat
basicConstraints = CA:true
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
5 keyUsage = cRLSign, keyCertSign
subjectAltName = email:copy
nsComment = "Die Root CA gibt ausschliesslich Unter-CA Zertifikate aus."
```

ext_userCA.cnf

```
# X.509v3 Erweiterungen fuer User CA Zertifikat
basicConstraints = CA:true
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
5 keyUsage = cRLSign, keyCertSign
subjectAltName = email:copy
nsCertType = sslCA,emailCA
nsCaRevocationUrl = cgi-bin/pyca/get-cert.py/CA/crl
nsRevocationUrl = cgi-bin/pyca/ns-check-rev.py/CA?
10 nsRenewalUrl = cgi-bin/pyca/ns-renewal.py/CA?
nsCaPolicyUrl = security/policy/ca.html
nsComment = "Diese CA gibt E-Mail und SSL Client Zertifikate aus."
```

ext_objectCA.cnf

```
# X.509v3 Erweiterungen fuer Object CA Zertifikat
basicConstraints = CA:true
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
5 keyUsage = cRLSign, keyCertSign
subjectAltName = email:copy
nsCertType = objCA
nsCaRevocationUrl = cgi-bin/pyca/get-cert.py/CA/crl
nsRevocationUrl = cgi-bin/pyca/ns-check-rev.py/CA?
10 nsRenewalUrl = cgi-bin/pyca/ns-renewal.py/CA?
nsCaPolicyUrl = security/policy/ca.html
nsComment = "Diese CA gibt Zertifikate aus um Objekte zu unterschreiben."
```

ext_serverCA.cnf

```
# X.509v3 Erweiterungen fuer Object CA Zertifikat
basicConstraints = CA:true
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
5 keyUsage = cRLSign, keyCertSign
subjectAltName = email:copy
nsCertType = sslCA
nsCaRevocationUrl = cgi-bin/pyca/get-cert.py/CA/crl
nsRevocationUrl = cgi-bin/pyca/ns-check-rev.py/CA?
10 nsRenewalUrl = cgi-bin/pyca/ns-renewal.py/CA?
nsCaPolicyUrl = security/policy/ca.html
nsComment = "Diese CA gibt SSL Server Zertifikate aus."
```

Weiterleitung von E-Mail mit .forward

Die Datei `.forward` enthält die automatische Weiterleitung einer eingehenden E-Mail an das Skript `ca-certreq-mail.py`. Wenn es sich um eine Bestätigung eines Zertifikatsantrages handelt, wird die entsprechende Datei vom Verzeichnis `pendreqs` ins Verzeichnis `newreqs` verschoben.

```
\caadmin, "|/home/caadmin/sbin/ca-certreq-mail.py \
--pycalib=/home/caadmin/pylib"
```

Zeitsteuerung mit der crontab

Die Zertifikate sollten in regelmäßigen Abständen an die Empfänger verteilt werden. Dies übernimmt die Zeitsteuerung `cron`. Die `crontab` sieht folgendermaßen aus:

```
SHELL=/bin/sh
4,14,24,34,44,54 * * * * $HOME/sbin/ca-cycle-pub.py \
--pycalib=/home/caadmin/pylib >> $HOME/cron/log 2>&1
```

B.3 Probleme

Viele Probleme tauchten auf, nachdem die ersten Zertifikate ausgegeben waren. Jedoch kamen auch später immer wieder neue hinzu. Weil ich jedem Problem sofort nachgegangen bin, konnte ich dadurch die meisten lösen. Die Problemlösung hat teilweise viel Zeit und manchmal auch Nerven gekostet. Sie war aber in jedem Fall erforderlich.

- Ein Mitarbeiter möchte das Initial seines Mittelnamens unbedingt im Zertifikat haben. Die Funktion `minify.mime_encode_header` im `client-enroll.py` Skript macht dabei den Namen kaputt. Damit kann keine E-Mail losgeschickt werden und das Skript schlägt fehl. Lösung: Zeile auskommentieren. Die E-Mail wird jetzt nur noch anhand der E-Mail Adresse und nicht mit der Kombination Name + E-Mail Adresse verschickt. Hinweis: Bei der Skriptsprache Python hängt die Einrückung durch Leerzeichen mit der Verschachtelungstiefe bei der Ausführung zusammen. Falsche Einrücktiefen führen zu Laufzeitfehlern.
- Personen, die ihren Namen nicht richtig schreiben, können keine E-Mail zugestellt bekommen und somit ihr Zertifikat nicht beantragen. Gefunden wurde dies durch die Sichtung der Log-Dateien.
- Manche Mitarbeiter haben alte oder uralte Versionen von E-Mail Programmen. Diese beinhalten die S/MIME Funktionalität nicht. Sie zeigen eine digitale Unterschrift als `.p7s` (S/MIME Cryptographic Signature) Attachment an. Lösung: neue Version installieren.
- Bis alle Mitarbeiter die CA Zertifikate haben, sind bei einigen Mitarbeitern die digitalen Unterschriften ungültig. Das erledigt sich, wenn alle die öffentlichen CA Zertifikate installiert haben.

- Netscape 6 stürzt ab, wenn ein Zertifikatsdatenbankpasswort vergeben werden soll. Lösung: Warten auf das nächste Update oder Version 4.76 benutzen.
- Unter (noch nicht näher bekannten) Umständen macht Netscape seine eigene, in der Datei `cert7.db` gespeicherte, Zertifikatsdatenbank kaputt. Das äußert sich so, dass kaputte Zertifikate nicht mehr aus der Datenbank gelöscht, aber auch nicht überschrieben werden können.
- Netscape auf Linux sendet im Mail-Header das Feld *Sender* mit dem Inhalt des Usernamens, unter dem die Person angemeldet ist, mit. Outlook beachtet dieses Feld und schreibt folgendes in der Absenderzeile (Beispiel): von `hirscher@innovations.de` im Auftrag von `Daniel.Hirscher@innovations.de`. Aus unerfindlichen Gründen erzeugt Netscape das Feld im Header. Es lässt sich unterdrücken, wenn man in der Datei `.netscape/preferences.js` im Homeverzeichnis folgende Zeile einfügt:

```
user_pref("mail.suppress_sender_header", true);
```

Die Einstellungen können mit `about:config` (in der Location Zeile eingeben) angesehen werden.

- Wenn das Benutzerprofil für Netscape auf einem Netzlaufwerk liegt und die Verbindung zu diesem Laufwerk unterbrochen war, dann stellt Netscape die Verbindung nicht mehr automatisch her. Das Problem ist dann, dass alle Zertifikate (auch die Stammzertifizierungsstellen) nicht mehr verfügbar sind. Beim Versenden einer E-Mail bekommt man eine unverständliche Meldung, dass ein Server nicht mehr erreichbar sei. Lösung: Netscape beenden und neu starten.
- Outlook Benutzer können zwar digitale Unterschriften erkennen, aber das darin enthaltene Zertifikat wird nicht automatisch in die Zertifikatsdatenbank übernommen. Lösung: Zertifikat von Hand importieren, oder die Option *Bei Versenden ins Adressbuch übernehmen* aktivieren.
- Outlook kann eine (nicht Klartext-) unterschriebene oder verschlüsselte E-Mail nicht im Schnellvorschauenfenster anzeigen. Dies ist wohl Absicht. Es gibt keine Option, die das Verhalten ändert.
- Falls ein Empfänger, dessen E-Mail Programm keine S/MIME Funktionalität beherrscht, eine digital unterschriebene E-Mail nicht lesen kann, so ist vom Sender keine Klartextunterschrift erzeugt worden. Dies kann

leicht bei der Benutzung von Outlook passieren. Hier sollte jeder im Menü unter *Optionen* → *Sicherheit*, die Einstellung *Unterschrift im Klartext senden* aktivieren.

- Um in Outlook eine E-Mail verschlüsselt an sich selbst zu schicken, muss man sein eigenes öffentliches Zertifikat explizit in seinen eigenen Adressbucheintrag importieren. Es reicht nicht, dass das öffentliche Zertifikat zusammen mit dem privaten Schlüssel ohnehin schon in den Voreinstellungen eingetragen wurde. Lösung: eigenes öffentliches Zertifikat importieren.
- Outlook benutzt zur Verschlüsselung RC2 (Rivest Cipher 2) 40-Bit, obwohl in den Voreinstellungen 3DES eingestellt ist! Der High-Encryption-Pack ist installiert. Zertifikate beinhalten keinerlei Informationen über Schlüsselstärken. Wenn jedoch Outlook das Zertifikat über eine digital unterschriebene E-Mail bekommen hat, stehen dort Informationen über symmetrische Verschlüsselungsalgorithmen des Senders dabei. Aus unerfindlichen Gründen entscheidet sich Outlook für das schwächste Verfahren, wenn man dieser Person schreiben will. Lösung: der Zertifikatsinhaber schickt nochmals eine digital unterschriebene E-Mail, nachdem er schwache Schlüssel abgeschaltet hat. Dadurch wird Outlook gezwungen, eine starke Verschlüsselung für diesen Empfänger zu verwenden.
- Bei Outlook 2000 gibt es plötzlich zwei verschiedene DLLs, die für die Verschlüsselung zuständig sind. Zu finden sind sie im Hilfemenü unter dem Punkt Info. Eines ist für Outlook 2000 und eines für den Internet Explorer. Für beide benötigt man Patches, um sie von 40-Bit auf 128-Bit zu verstärken. Dabei braucht man laut Microsoft den Patch für Outlook 2000 nur, wenn man nicht Windows 2000 benutzt. Die Schlüsselstärke für den Internet Explorer (und Outlook Express) deckt der High Encryption Patch für das entsprechende Windows Betriebssystem ab.
- Outlook Express 5.5 mit High-Encryption Pack zeigt eine Warnung bei unbekannter Unterschrift an. Beim Anzeigen des Zertifikates dauert es sehr lange bis sich das Fenster öffnet. Es gab einen Absturz beim versuchten Anzeigen der E-Mail. Auf die E-Mail mit der ungültigen Unterschrift kann nicht geantwortet werden. Der Grund hierfür ist unbekannt. Es besteht die Möglichkeit, dass es an den fehlenden Root Zertifikaten liegt. Dieses Problem trat bei einer externen Person auf und konnte selbst nicht reproduziert werden.

- Mit Outlook 2000 kann das private Zertifikat nicht exportiert werden. Das kann auch fehlschlagen, wenn beide Sicherheitsinformationen 128-Bit Stärke zeigen. Nachdem man alle Daten eingegeben hat, erscheint die Fehlermeldung: *Digitale IDs und Schlüsselangaben können nicht exportiert werden*. Lösung: über den Internet Explorer das Zertifikat exportieren.
- Wenn mit Outlook 2000 eine E-Mail digital unterschrieben (und nicht verschlüsselt) wird, so ist die Unterschrift ungültig, wenn Umlaute enthalten sind. Dieses Problem wird durch Einspielen des Office 2000 SR-1a (Service Release) gelöst (Outlook 2000 Produkt ID: 9.0.0.3821).
- Im Internet Explorer kann keine Verschlüsselungsstärke größer als 512-Bit ausgewählt werden. Lösung: High-Encryption-Patch installieren.
- Im Internet Explorer wird bei der Schlüssellänge ein leeres Auswahlfeld angezeigt. Zur Erzeugung der Schlüssel im IE (Internet Explorer) muss VB-Script installiert und aktiviert sein. Lösung: VB-Script installieren, aktivieren oder Schlüssel mit Netscape erzeugen und exportieren und in Windows importieren.
- Bei Windows kommt nach dem Login manchmal die Meldung, dass das lokale Profil aktueller sei und ob man das Netzwerkprofil überschreiben will. Wenn zuvor (also beim letzten Login) der Schlüssel erzeugt und gespeichert wurde, wird beim Überschreiben des Profils der Schlüssel zerstört.
- Windows lässt Zertifikate ungültig werden, wenn der Zertifikatsaussteller eine kürzere Gültigkeit besitzt als das Zertifikat selbst. Die Fehlermeldung ist: *Die Gültigkeitsdauer des Zertifikates übersteigt die seiner Zertifizierungsstelle*. Beispiel: Das Zertifikat der Zwischenzertifizierungsstelle User ist gültig vom 11.12.2000 bis zum 11.12.2002. Ein Benutzerzertifikat ist gültig vom 13.12.2000 bis zum 13.12.2002. In Windows ist es damit ungültig. Eigentlich wäre es bis zum 13.12.2002 gültig und bis zum 11.12.2002 vertrauenswürdig. Dieses Problem taucht bis jetzt nur bei einer Person auf und konnte noch nicht gelöst werden.
- The Bat! Benutzer haben noch Probleme, weil dieser E-Mail Client S/MIME erst ab der Version 1.47 unterstützt. Im Dezember 2000 erschien die Version 1.48, die Besserung verspricht. Dies ist auch ein Beispiel dafür, dass das Thema der sicheren Kommunikation zur Zeit immer populärer wird.

- Wenn eine unterschriebene und verschlüsselte Nachricht an TheBat! (Version 1.48) geschickt wird, so erscheint diese in TheBat!, als ob sie zweimal verschlüsselt ist. Die Unterschrift wird nicht angezeigt. Laut RITlabs, dem Hersteller, wird dieses Problem behoben.
- Bei TheBat! werden die S/MIME Einstellungen nicht gespeichert. Laut RITlabs wird dieses Problem behoben.
- Wenn ein bestimmter TheBat! Benutzer eine digital unterschriebene E-Mail mit Anhang verschickt, dann ist der Anhang kaputt. Das liegt daran, dass die Anhänge nicht BASE64 kodiert sind, obwohl dies in den Voreinstellungen eingestellt ist, sondern im binären Rohformat verschickt werden. Dieses Problem kann beim betroffenen Benutzer reproduziert werden, aber von niemand anderem, auch nicht bei RITlabs, dem Hersteller.
- Ein Fehler in meiner `openssl.cnf` führte dazu, dass in den Netscape Erweiterungen bei den URL Angaben falsche Werte standen. Dadurch konnte z.B. die Online Gültigkeitsprüfung der Zertifikate nicht durchgeführt werden. Dies lässt sich leicht durch einen Alias in der Konfigurationsdatei beheben (`EmailCerts` ist jetzt gleich `User`). Die danach erstellten Zertifikate enthalten die korrekten Angaben.

B.4 SSL Server Zertifikat auf Extranet Server einrichten

Anleitung für das Erstellen von SSL Server Zertifikaten und das Konfigurieren des Servers. Die Beispiele sind für einen *Apache* Web-Server.

B.4.1 SSL Server Zertifikat generieren

1. Über das Web-Interface Server Zertifikat beantragen. Der commonName (Feld: Name Vorname) muss der FQDN (Fully Qualified Domain Name) sein, z.B. `extranet.innovations.de`.
2. Als E-Mail Adresse z.B. `caadmin@innovations.de` angeben.
3. Der angegebene E-Mail Empfänger muss die Mail beantworten.
4. Warten bis das Zertifikat unterschrieben ist und zugestellt wird.

5. Zertifikat aus dem Web-Browser exportieren.
6. Das PKCS#12 Format in PEM umwandeln mit:

```
> openssl pkcs12 -in <name>.p12 -out <name>.pem
```
7. Mit einem Editor den Key und das öffentliche Zertifikat herauslösen und in die zwei Dateien `<name>.key` und `<name>.crt` speichern.
8. Das (Export) Passwort aus dem privaten Schlüssel entfernen mit:

```
> mv <name>.key <name>.pw
> openssl rsa -in <name>.pw -out <name>.key
> chmod 400 <name>.key
```
9. Die zwei Dateien nach `/etc/httpd/ssl.<key|crt>` kopieren.

Hinweis: Laut Policy muss auch hier eine E-Mail Adresse angegeben werden. Es dürfen keine Umlaute und Sonderzeichen wie `&`, `#`, `!`, usw. in den Textfeldern verwendet werden. Als CN (Common Name) muss der FQDN angegeben werden, z.B. `extranet.innovations.de`. Es reicht nicht `extranet` alleine anzugeben. Jeder Browser wird eine Warnmeldung ausgeben. Eine andere schlechte Lösung wäre `*.innovations.de` als CN anzugeben.

B.4.2 Serverkonfiguration

Die Konfiguration für den *Apache* Web-Server wird unter Linux in der Datei `/etc/httpd/httpd.conf` vorgenommen. Die SSL Optionen sind ganz zum Schluss zu finden. Wichtige Einstellungen sind (siehe auch Anleitung zu `mod_ssl` in [Eng99]):

- Einschalten des SSL Mechanismus. Port 443, HTTPS läuft als Virtueller Host.

```
SSLEngine on
```

- Angabe des Server Zertifikates im PEM Format.

```
SSLCertificateFile /etc/httpd/ssl.crt/extranet.crt
```

- Angabe des privaten Schlüssels, der zum Zertifikat gehört, auch im PEM Format. Wenn man vom Schlüssel das Export-Passwort entfernt hat, sollte man darauf achten, dass die Datei nur von root lesbar ist (`chmod 400 extranet.key`).

```
SSLCertificateKeyFile /etc/httpd/ssl.key/extranet.key
```

Damit ist eine sichere Kommunikation mit dem Server möglich wenn die URL mit `https://` beginnt. Es ist jedoch möglich, die selbe Seite mit `http://` aufzurufen. Diese ist dann unverschlüsselt.

SSL mit Client Authentication

- Um Client Authentication einzuschalten, d.h. der Client muss sich gegenüber dem Server mit einem Zertifikat ausweisen, welches der Server zulässt, muss `require` angegeben werden.

```
SSLVerifyClient require
```

- Die zur Überprüfung herangezogenen CA Zertifikate müssen in diesem Verzeichnis liegen. Hier befindet sich auch ein Makefile, das symbolische Links auf die PEM Dateien erzeugt. Diese werden vom Server benötigt, um das richtige Zertifikat zu finden.

```
SSLCACertificatePath /etc/httpd/ssl.crt
```

- Hier wird angegeben bis zu welcher Tiefe die Client Zertifikate überprüfbar sein müssen, bis ihm der Server vertraut. Die Angabe von 0 vertraut immer und ist deshalb nicht zu empfehlen.

```
SSLVerifyDepth 2
```

Weitere Konfigurationsmöglichkeiten

- Um die Verschlüsselungsverfahren einzuschränken, z.B. um nur starke Verschlüsselung zuzulassen, kann hier angegeben werden:

```
SSLCipherSuite \  
ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:!LOW:!SSLv2:!EXP:!eNULL
```

- Um bei bestimmten Verzeichnissen die SSL Verschlüsselung zu erzwingen, kann `SSLRequireSSL` innerhalb eines `<Location>` Abschnittes angegeben werden. Genauere Einschränkungen sind mit `SSLRequire` möglich. Zu beachten ist, dass der `<Location>` Abschnitt außerhalb des `<VirtualHost>` Abschnittes liegt. Der Benutzer bekommt eine 403 `Forbidden` Antwort wenn er versucht, ohne SSL auf die Seite zuzugreifen.

```
<Location />
SSLRequire %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
    and %{SSL_CLIENT_S_DN_O} eq "Innovations GmbH" \
    and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
    and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20
</Location>
```

B.5 Object Zertifikat zum Unterschreiben von Code

Die Sicherheitswerkzeuge aus Suns JDK bieten eine funktionierende Lösung zum digitalen Unterschreiben von Programmen. Jedoch beschränkt sich das Unterschreiben auf Java Programme. Dafür funktioniert es auf allen Betriebssystemen und in allen Web-Browsern, sofern das Java Plugin installiert ist. Man hat einige Mühe, seine Zertifikate in den Keystore zu bringen, aber dann funktioniert es.

Da es nicht möglich ist, mit dem Keytool private Schlüssel in den Keystore zu importieren, muss der Schlüssel mit dem Keytool selbst erstellt werden. Standardmäßig werden allerdings DSA (Digital Signature Algorithm) Schlüssel erzeugt. Um RSA Schlüssel zu erhalten muss dies angegeben werden. Um einen Request zu erzeugen muss man zuerst ein Schlüsselpaar erzeugen. Danach kann der Request ausgegeben werden.

```
> keytool -genkey -alias aliasname -keyalg RSA
> keytool -certreq -alias aliasname > reqdateiname
```

Der Request vom Keytool liegt im PKCS#10 Format vor (PEM codiert) und kann von der PKI unterschrieben werden mit:

```
> openssl ca -name CA_Object -in request
```

Der unterschriebene Request ist DER (Distinguished Encoding Rules) codiert. Wenn er PEM codiert ist, muss man allen Text vor dem Trenner -----BEGIN CERTIFICATE----- entfernen. Er kann in das Keytool importiert werden mit:

```
> keytool -import -alias aliasname -file dateiname
```

Wenn die Root CA Zertifikate auch in den Keystore importiert sind, wird die Zertifikatskette (Certificate Chain) aufgelistet.

```
> keytool -v -list
```

Wenn man dann ein Schlüsselpaar im Keystore gespeichert hat, geht das Unterschreiben sehr einfach. Dazu benötigt man noch die fertige JAR Datei, egal ob Applet oder Applikation.

```
> jarsigner datei.jar aliasname
```

Hinweis: Applets die mit dem Jarsigner von Sun unterschrieben sind benötigen das Java Browser Plugin: <http://java.sun.com/products/plugin/>. Er ist für viele Betriebssysteme erhältlich.

Wenn man nun eine gültig unterschriebene JAR Datei entpackt, einige Bytes verändert und wieder packt dann funktioniert das Programm nicht mehr. Beim Packen mit `jar` darf die Datei `MANIFEST.MF` nicht neu geschrieben werden, sonst wird nicht mehr erkannt, dass unterschrieben wurde (Option `-M`). Jarsigner stellt bei der Überprüfung der Unterschrift eine Verletzung des Fingerabdrucks fest. Ein Applet, das im Web-Browser läuft und außerhalb seiner Sandbox auf das Dateisystem zugreifen will und dem dies nicht erlaubt ist, erzeugt einen `SecurityException` Fehler.

Aus der Mailingliste *Java-Security* von Sun erfährt man, wie man den privaten Schlüssel bzw. ein komplettes Zertifikat in den Keystore importieren kann. Dazu existiert die Methode `KeyStore.setKeyEntry()` aus dem Paket `java.security`. Möglicherweise wird das Keytool in einer späteren Version das direkte Importieren erlauben.

Anhang C

Protokolle und Verfahren

In diesem Kapitel werden die Protokolle und die Verfahren vorgestellt, die von PGP, S/MIME, SSL und IPsec verwendet werden. Zur Verdeutlichung kommen dabei traditionell die zwei Personen *Alice* und *Bob* zum Einsatz.

C.1 Symmetrische Verschlüsselungsverfahren

Symmetrische Verschlüsselungsverfahren verwenden den gleichen Schlüssel zum Verschlüsseln wie auch zum Entschlüsseln. Der Schlüssel ist dabei eine (oft zufällige) Bitfolge. Die Verfahren sind sehr schnell. Die Operationen beschränken sich hauptsächlich auf Konfusion (= Vermischen der Bits) und Diffusion (= Verteilen der Bits), die in mehreren Runden wiederholt angewendet werden.

Es wird unterschieden zwischen Blockchiffren, die die zu verschlüsselnde Nachricht in Blöcke gleicher Länge aufteilen, und Stromchiffren, die die Nachricht fortlaufend ver- bzw. entschlüsseln. Da Stromchiffren weniger getestet sind, werden sie seltener eingesetzt [Sch98, S. 137], obwohl sie gleich stark sind.

3DES Alter Datenverschlüsselungsstandard. Es ist ein 64-Bit-Blockchiffrierer. Die effektive Schlüssellänge von DES ist 56-Bit, von 3DES 112-Bit oder 168-Bit. Er ist seit über 20 Jahren weit verbreitet, 1976 übernommen als FIPS 46.

Die ursprüngliche Version kann heutzutage mit vertretbarem Aufwand geknackt werden. Deshalb verwendet man DES dreimal hintereinander mit zwei oder drei verschiedenen Schlüsseln.

CAST5 Algorithmus von Carlisle Adams und Stafford Tavares, daher der Name. Blockchiffrierung mit Blockgröße und Schlüssel der Länge 64-Bit.

IDEA Der IDEA Block Chiffrier Algorithmus wurde an der ETH (Eidgenössische Technische Hochschule) Zürich von James L. Massey und Xuejia Lai entwickelt und 1990 veröffentlicht. IDEA hat vielen kryptoanalytischen Attacken widerstanden.

IDEA ist eine Blockchiffrierung mit 64-Bit Blockgröße und 128-Bit Schlüssellänge. Er funktioniert nach der „Mischung von Operationen unterschiedlicher algebraischer Gruppen“ und verwendet dabei XOR, Addition modulo 2^{16} und Multiplikation modulo $2^{16} + 1$. Das Verfahren verschlüsselt in 8 Runden und kann im CBC (Cipher Block Chaining) und CFB (Cipher Feedback) Modus betrieben werden.

PGP-2.6.3i benutzt IDEA im 64-Bit CFB Modus.

Da auf den Algorithmus noch einige Jahre ein Patent besteht, ist er nicht in GnuPG implementiert.

RC2 Von Ron Rivest entwickelter nichtiterativer 64-Bit-Blockchiffrieralgorithmus, dessen Schlüssellänge variabel ist. Er ist etwas schneller als DES und enthält keine S-Boxen. In den Runden werden die Operationen *mix* oder *mash* ausgeführt. Siehe auch [Sch97, S. 368].

S/MIME benutzt RC2 mit 40-Bit, 64-Bit und 128-Bit langen Schlüsseln immer im CBC Modus.

RC4 Von Ron Rivest entwickelter Stromchiffriercode mit variabler Schlüsselgröße. Er war früher Eigentum von RSA Data Security. Er wird bei SSL Verbindungen benutzt.

Rijndael Der „Siegerkandidat“ für den neuen AES Standard vom NIST. Er soll DES ablösen. Es ist ein von J. Daemen und V. Rijmen entwickelter Blockchiffrieralgorithmus mit variabler Block- und Schlüssellänge. Er arbeitet mit mehreren Runden. Er ist ab der Version 1.0.4 von GnuPG implementiert.

C.2 Asymmetrische Verschlüsselungsverfahren

Bei asymmetrischen Verschlüsselungsverfahren werden zwei Schlüssel benutzt. Einer zum Verschlüsseln (öffentlich), ein dazugehöriger zweiter (pri-

vat) zum Entschlüsseln. Die Verfahren beruhen auf mathematischen Formeln. Das Chiffre und der Klartext werden jeweils berechnet. Die Schlüssel sind als Zahlen anzusehen.

C.2.1 RSA

Beim Verschlüsselungsverfahren nach RSA [Sch97, Ert98, Har00, Wob00] muss zuerst ein Schlüsselpaar erzeugt werden. Danach kann ver- und entschlüsselt werden.

Schlüsselerzeugung

Jeder Beteiligte bei einer Public Key Verschlüsselung muss selbst zwei Schlüssel erzeugen. Einen geheimen, den er für sich behält, und einen öffentlichen, den er an alle seine Kommunikationspartner verteilt. Alice wie auch Bob führen folgendes, jeder für sich, durch:

- (1) Man wählt zwei große Primzahlen p und q und berechnet $n = p \cdot q$.
- (2) Man wählt eine zufällige Zahl $e > 1$ so, dass e und $(p - 1)(q - 1)$ teilerfremd zueinander sind.
- (3) Man berechnet $d = e^{-1} \bmod ((p - 1)(q - 1))$ mit dem erweiterten Euklidischen Algorithmus, siehe auch [Sch97, S. 288].
- (4) Man vernichtet p und q .
- (5) Der geheime Schlüssel ist: $K_S = (d, n)$.
- (6) Der öffentliche Schlüssel ist: $K_P = (e, n)$.

Verschlüsseln

Um eine Nachricht für einen bestimmten Empfänger zu verschlüsseln, benötigt man dessen öffentlichen Schlüssel. Wenn Alice für Bob eine Nachricht verschlüsseln will, geht sie so vor:

- (1) Zerlege den Klartext M in Blöcke der Größe der Schlüssellänge.
- (2) Alice verschlüsselt die Nachrichtenblöcke von M mit Bobs öffentlichem Schlüssel $K_{P_{Bob}}$ indem sie $C = M^e \bmod n$ berechnet.

- (3) Alice sendet C an Bob.
- (4) Bob benutzt seinen geheimen Schlüssel $K_{S\text{Bob}}$ um die Nachricht zu entschlüsseln und berechnet $M = C^d \bmod n$ um M zu erhalten.

Sicherheit von RSA

Das RSA Verfahren basiert auf der Schwierigkeit der Primfaktorzerlegung.

C.2.2 El Gamal

Beim Verschlüsselungsverfahren nach ELG (El Gamal) [Sch97, Har00] muss zuerst wie bei RSA ein Schlüsselpaar erzeugt werden. Erst danach kann ver- und entschlüsselt werden.

PGP und S/MIME können ELG Schlüssel benutzen. Ein Nachteil ist aber, dass der Chiffretext doppelt so lang ist wie der Klartext. Da das Patent auf RSA abgelaufen ist, besteht keine Notwendigkeit mehr, ELG deshalb als Alternative einzusetzen.

Sicherheit von El Gamal

Das ELG Verfahren basiert auf der Schwierigkeit, den diskreten Logarithmus über einem endlichen Körper zu berechnen.

C.3 Einwegfunktionen

Zur kryptographischen Integritätssicherung dienen Einweg-Hashfunktionen, auch Falltürfunktion genannt. Sie erzeugen einen Fingerabdruck (Fingerprint), auch Message Digest genannt. Diese werden zum Prüfen eines Zertifikates und zum Verkleinern der Nachricht beim Unterschreiben verwendet.

Die Einwegfunktion H verarbeitet eine beliebig lange Nachricht M und erzeugt einen Hashwert h fester Länge m : $h = H(M)$. Es gelten folgende Eigenschaften:

- Zu gegebenem M ist es leicht, h zu berechnen.
- Zu gegebenem h ist es schwer, ein M zu berechnen mit $H(M) = h$.

- Zu gegebenem M ist es schwer, eine andere Nachricht M' zu berechnen mit $H(M) = H(M')$.
- Es ist schwer, zwei beliebige Nachrichten M und M' zu finden mit $H(M) = H(M')$ (Kollisionsresistenz).

Einwegfunktionen erzeugen also zu der gleichen Eingabe immer die gleiche Ausgabe. Es ist aber schwer möglich, zu gegebener Ausgabe eine passende Eingabe zu finden. Sie werden bei digitalen Unterschriften eingesetzt, da jede Änderung des Textes auch den Fingerabdruck ändert und damit die Unterschrift ungültig werden lässt.

MD5 Der MD5 wurde von Ronald Rivest entwickelt. Er ist eine Modifikation des MD4 (Message Digest 4). Leider hat der Algorithmus eine Schwäche, welche die Berechnung von bestimmten Kollisionen ermöglicht [Sch98]. Er produziert einen 128-Bit Hashwert. Der MD5 wird langsam aber sicher von SHA-1 abgelöst.

SHA-1 Der SHA-1 wurde von der NSA (National Security Agency) zur Verwendung im DSS entwickelt. Er ist eine Weiterentwicklung von SHA (Secure Hash Algorithm), was ihn sicherer macht. Er produziert einen 160-Bit Hashwert.

C.4 Schlüsseltausch

C.4.1 Diffie-Hellmann

Der Schlüsseltausch nach Diffie-Hellmann [Sch97, Ert98, Har00] kommt bei SSL unter dem Namen KEA (Key Exchange Algorithm) zum Einsatz. Es funktioniert folgendermaßen (siehe auch Abbildung C.1):

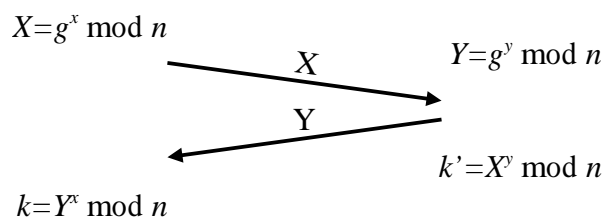


Abbildung C.1: Schlüsseltausch nach Diffie-Hellman.

- (1) Alice und Bob einigen sich auf eine große Primzahl n und eine Zahl g .
- (2) Alice wählt eine große zufällige Zahl x , berechnet $X = g^x \bmod n$ und sendet X an Bob.
- (3) Bob wählt eine große zufällige Zahl y , berechnet $Y = g^y \bmod n$ und sendet Y an Alice.
- (4) Alice berechnet $k = Y^x \bmod n$ und erhält mit k den geheimen Schlüssel.
- (5) Bob berechnet $k' = X^y \bmod n$ und erhält ebenfalls mit k' den geheimen Schlüssel.

Es gilt $k = k' = g^{xy} \bmod n$. Damit haben sich Alice und Bob auf einen gemeinsamen geheimen Schlüssel geeinigt. Auf dem Übertragungskanal wurde nur n , g , X und Y übermittelt. Damit ist es für einen Lauscher schwierig, durch das Problem des diskreten Logarithmus x und y auszurechnen.

Das Diffie-Hellmann Protokoll ist anfällig gegen einen Man-in-the-Middle Angriff. Dies kann z.B. durch Unterschreiben der übertragenen Zahlen verhindert werden.

C.5 Zufallszahlen

Alle oben genannten Verfahren sind abhängig von Zufallszahlen. Sind diese nicht wirklich „zufällig“, wird das Verfahren verwundbar [ECS94]. Echte Zufallszahlen sind nicht vorhersehbar. Solche sind auf deterministischen Maschinen, wie Computern, nur mit spezieller Hardware zu erzeugen. Pseudozufallszahlen, wie sie gewöhnlich verwendet werden, sind hierbei völlig ungeeignet, da sie die Anforderungen, nicht vorhersehbar zu sein, nicht erfüllen. Sie sind allenfalls für Spiele ausreichend.

Die meisten kryptographischen Programme benutzen daher eine Mischform zur Erzeugung der benötigten Zufallszahlen (siehe auch Tabelle 2.2). Auf die dadurch gewonnene „Zufallszahl“ wird oft noch eine Hashfunktion angewendet, um Rückschlüsse auf die Eingangsdaten zu erschweren.

C.6 X.509 Zertifikat

C.6.1 Versionen

Der X.509 Standard für Zertifikate wurde von der ITU (International Telecommunication Union) definiert und im Laufe der Zeit erweitert und geändert.

X.509v1 Ursprüngliche Definition von 1988, sehr allgemein gehalten.

X.509v2 Eindeutige Identifikation für Besitzer (Subject) und Aussteller (Issuer) wurden eingeführt. Damit können prinzipiell diese Namen wiederverwendet werden. Viele Zertifizierungsrichtlinien empfehlen aber, genau dies nicht zu tun, und so sind X.509v2 Zertifikate kaum in Gebrauch.

X.509v3 In der aktuellen Version von 1994 wurden Erweiterungen (Extensions) eingeführt. Prinzipiell kann jeder seine eigenen Erweiterungen definieren und in das Zertifikat einfügen. Es gibt jedoch einige gebräuchliche Erweiterungen wie z.B. `keyUsage` wodurch die Nutzung des Schlüssels für bestimmte Zwecke eingeschränkt werden kann. Zusätzlich kann jede Erweiterung als kritisch (critical) markiert werden.

C.6.2 Beispielzertifikat

Ein X.509 Zertifikat in der üblichen Textdarstellung (kryptographische Angaben sind gekürzt).

```

1 Certificate:
2   Data:
3     Version: 3 (0x2)
4     Serial Number: 1 (0x1)
5     Signature Algorithm: md5WithRSAEncryption
6     Issuer: C=DE, ST=Baden-Wuerttemberg, L=Immenstaad, O=Innovations GmbH,
7           CN=Innovations GmbH User CA/Email=caadmin@innovations.de
8     Validity
9       Not Before: Dec 11 12:46:11 2000 GMT
10      Not After : Dec 11 12:46:11 2002 GMT
11     Subject: C=DE, ST=Baden-Wuerttemberg, L=Immenstaad, O=Innovations GmbH,
12            CN=Daniel Hirscher/Email=Daniel.Hirscher@innovations.de
13     Subject Public Key Info:
14       Public Key Algorithm: rsaEncryption
15       RSA Public Key: (1024 bit)
16         Modulus (1024 bit):
17           00:dc:59:8b:ae:af:da:92:f3:43:f4:f9:6e:ba:f5:
18           e9:0c:72:a6:f8:e0:ac:d5:1a:2f:1a:f8:86:80:6e:
19           [...]
20           2b:23:56:fb:80:37:c7:98:47:22:f1:b8:3c:c2:65:
21           12:c5:24:22:99:4b:67:f2:f5
22       Exponent: 65537 (0x10001)

```

```

23 X509v3 extensions:
24 X509v3 Basic Constraints:
25   CA:FALSE
26 X509v3 Key Usage:
27   Digital Signature, Non Repudiation, Key Encipherment
28 Netscape Comment:
29   Dieses Zertifikat wird fuer E-Mail und als SSL Client benutzt.
30 Netscape Base Url:
31   http://intranet.innovations.de/
32 Netscape CA Revocation Url:
33   cgi-bin/pyca/get-cert.py/EmailCerts/crl
34 Netscape Revocation Url:
35   cgi-bin/pyca/ns-check-rev.py/EmailCerts?
36 Netscape Renewal Url:
37   cgi-bin/pyca/ns-renewal.py/EmailCerts?
38 Netscape CA Policy Url:
39   security/policy/user.html
40 Netscape Cert Type:
41   SSL Client, S/MIME
42 Signature Algorithm: md5WithRSAEncryption
43 ac:64:e0:20:8f:49:a6:27:93:66:bd:35:74:1e:1d:62:a2:06:
44 09:59:8a:77:9e:2c:08:15:bc:2c:af:ee:42:1d:16:ba:64:be:
45 [...]
46 3a:1e:16:89:78:bb:07:2b:15:cf:25:98:38:d7:7a:8d:12:e5:
47 11:c2

```

Beschreibung der Felder (Zeilen) des Zertifikates:

- 3** Version, sollte immer X.509v3 sein (Zählend von 0)
- 4** Seriennummer (Zählend von 1)
- 5** Unterschriftsalgorithmus, hier `md5WithRSAEncryption`.
- 6–7** Aussteller (Zertifizierungsinstanz, CA), hier werden LDAP Feldbezeichnungen verwendet.
- 8–10** Gültigkeitszeitraum.
- 11–12** Subject, hierfür ist das Zertifikat ausgestellt.
- 14** Asymmetrischer Verschlüsselungsalgorithmus, hier `rsaEncryption`.
- 15** Länge des öffentlichen Schlüssels in Bit, für die Innovations CA immer 1024-Bit.
- 16–21** Modulus n ($= p \cdot q$) des öffentlichen Schlüssels.
- 22** Exponent, enthält wenige Einsen in der Binärdarstellung für schnelle Rechnung. Angriffe hierüber sind möglich, siehe [Sch98, S. 95]!
- 23–41** X.509v3 Erweiterungen, meistens Netscape-eigene Erweiterungen.
- 24–25** Basiseinschränkungen, entweder `CA:true` oder `CA:false`, je nachdem, ob das Zertifikat zur CA gehört oder ein Endbenutzerzertifikat ist.
- 26–27** Vorgesehene Schlüsselbenutzung; Windows beachtet diese Angaben nicht.

28–41 Netscape-eigene Erweiterungen, wie Kommentar, Schlüsselbenutzung, und URLs zur Online-Überprüfung, Wartung und Erneuerung von Zertifikaten.

42 Algorithmus der Unterschrift, hier `md5WidthRSAEncryption`.

43–47 Unterschrift des Ausstellers.

Netscape kann die Zertifikathierarchie grafisch nicht abbilden. Es kann jedes Zertifikat explizit als vertrauenswürdig markiert werden. Das gilt auch für Zwischenzertifizierungsstellen.

Wie die einzelnen Programme die privaten Schlüssel verschlüsseln bleibt ihnen überlassen. Meist wird 3DES benutzt.

Zur Speicherung der Zertifikate bzw. auch bei der Übertragung innerhalb einer E-Mail kommen unterschiedliche Formate zum Einsatz. Einen Überblick liefert Tabelle C.1. Siehe auch Tabelle C.2 für weitere PKCS#7 (Cryptographic Message Syntax Standard) Dateierweiterungen.

Suffix	Bedeutung
<code>.pem</code>	PEM Format, Base64 kodiert (ASCII Armored)
<code>.der</code>	DER Format, binär
<code>.cer</code>	unbestimmt, kann PEM oder DER sein
<code>.crt</code>	unbestimmt, kann PEM oder DER sein
<code>.p12</code>	PKCS#12 Format, binär, enthält privaten Schlüssel
<code>.pfx</code>	Privater Informationsaustausch, veraltet, entspricht PKCS#12
<code>.p7c</code>	PKCS#7, Dateianhang in E-Mail

Tabelle C.1: Kodierungen von Zertifikaten

C.6.3 PKCS

PKCS (Public-Key Cryptography System) definiert mehrere Standards für unterschiedliche Aufgaben bei der Nutzung einer PKI. Vier davon haben z.Z. eine praktische Bedeutung:

PKCS#7 - Cryptographic Message Syntax Standard beschreibt eine allgemeine Syntax für Daten, auf welche kryptographische Verfahren angewandt wurden, z.B. Unterschriften (digital signatures) oder verschlüsselte Daten (digital envelopes). Sie können rekursiv geschachtelt werden, d.h. eine Unterschrift kann verschlüsselt werden. Es können

Attribute (wie z.B. Zeitstempel) und weitere Zertifikate oder CRLs enthalten sein. Eine mit S/MIME unterschriebene oder verschlüsselte E-Mail wird im PKCS#7 Format versendet.

PKCS#10 - Certification Request Syntax Standard beschreibt eine Syntax für Zertifikatsanforderungen. Der Datensatz besteht aus eindeutigen Namen, zusätzlichen Attributen und dem öffentlichen Schlüssel. Er wird an eine Zertifizierungsstelle übermittelt. Der Internet Explorer benutzt PKCS#10.

PKCS#11 - Cryptographic Token Interface Standard definiert eine einheitliche Zugriffsmethode auf kryptographische Einheiten. Dabei kann es sich um Software oder um Hardware, wie z.B. einem externen Kartenleser handeln. PKCS#11 (Cryptographic Token Interface Standard) stellt eine Schnittstelle zwischen der Anwendung und den einzelnen kryptographischen Einheiten dar.

PKCS#12 - Personal Information Exchange Syntax Standard beschreibt eine Syntax für die Speicherung von Zertifikaten mit dem privaten Schlüssel, z.B. als Sicherungskopie. Zur Sicherung wird vom Zertifikat- und Schlüsselinhaber ein Kennwort vergeben. Es dient zur Verschlüsselung des PKCS#12 Datensatzes.

Die Syntax wird in ASN.1 (Abstract Syntax Notation) definiert, die Daten werden BER (Basic Encoding Rules) kodiert.

C.7 Verfahren, die X.509 Zertifikate benutzen

C.7.1 SSL

Secure Socket Layer können verwendet werden um sichere Verbindungen zu Web-Servern, sowie zu POP3 oder IMAP E-Mail Accounts aufzubauen.

SSL versteht, je nach Version, folgende Algorithmen (aus [Net98]) zur Verschlüsselung: RSA, DES, 3DES, RC2, RC4 (Rivest Cipher 4) und SKIPJACK, zur Authentifizierung: RSA und DSA. Für den Schlüsseltausch: RSA-key-exchange und KEA. Als Einwegfunktionen: MD5 und SHA-1.

Verbindungsaufbau

Der Ablauf des SSL Handshakes ist in Abbildung C.2 zu sehen.

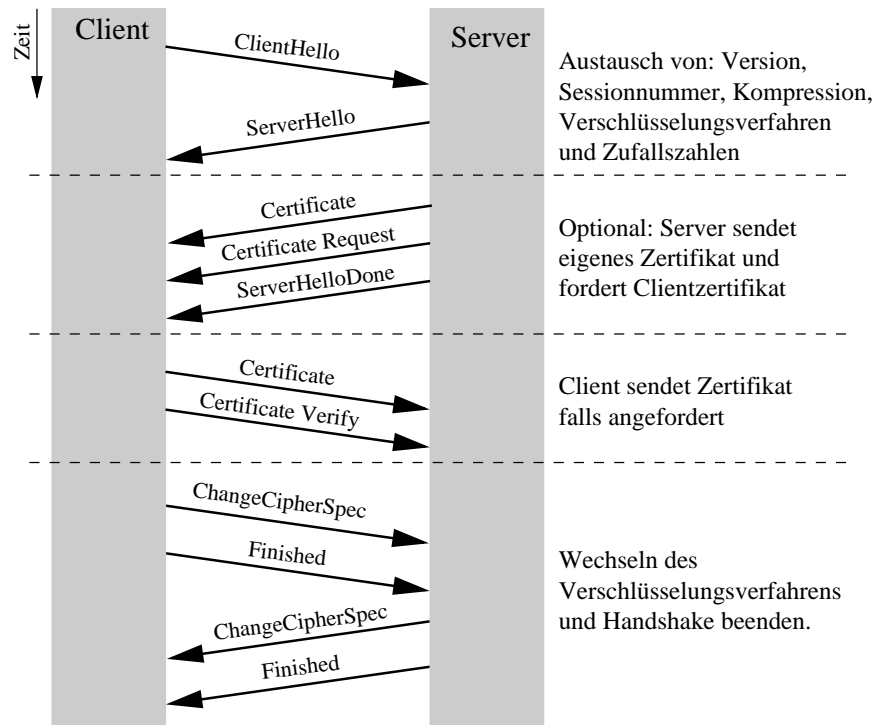


Abbildung C.2: Ablauf des SSL Handshake

C.7.2 S/MIME

Unterstützt folgende symmetrische Verschlüsselungsverfahren (alle im CBC Modus): 3DES 168-Bit, RC2 128-Bit, DES 56-Bit, RC2 64-Bit und RC2 40-Bit. Als Einwegfunktionen kommen MD5 und SHA-1 zum Einsatz. Als asymmetrisches Verfahren wird hauptsächlich RSA mit Schlüssellängen zwischen 512 und 1024-Bit benutzt. Es ist aber möglich, ELG Schlüssel zu benutzen.

C.7.3 IPsec

Der IPsec Standard definiert keine Verfahren sondern lässt alles offen. Deshalb stütze ich mich hier auf eine spezielle Implementierung, nämlich FreeS/WAN.

MIME Typ	Endung	Bedeutung
application/x-pkcs7-certificates	.p7b	mehrere Zertifikate
application/x-pkcs7-mime	.p7m	Unterschrieben + Verschlüsselt
application/x-pkcs7-mime	.p7c	nur Zertifikat
application/x-pkcs7-certreqresp	.p7r	Zertifikatsanforderungsantwort
application/x-pkcs7-signature	.p7s	Unterschrieben

Tabelle C.2: S/MIME Typen

FreeS/WAN unstützt MD5 und SHA-1 als Hashverfahren. Es versucht immer, 3DES für die symmetrische Verschlüsselung einzusetzen. Als asymmetrisches Verfahren kommt RSA oder Diffie Hellman (Gruppe 2 oder 5, siehe [HC98, A. 6]) zum Einsatz.

Es gibt zwei verschiedene Betriebsmodi (siehe auch Abbildung C.3):

Tunnel Modus Das gesamte IP (Internet Protocol) Paket wird verschlüsselt und im Datenfeld eines neuen IP Paketes verschickt (tunneln). Es wird mit einem neuen IP sowie IPsec Header versehen.

Transport Modus Die Daten des IP Paketes werden verschlüsselt. Der IP Header bleibt erhalten. Es wird noch ein IPsec Header dazwischengeschaltet. Der Transport Modus kommt bei einer Ende-zu-Ende Verbindung zum Einsatz.

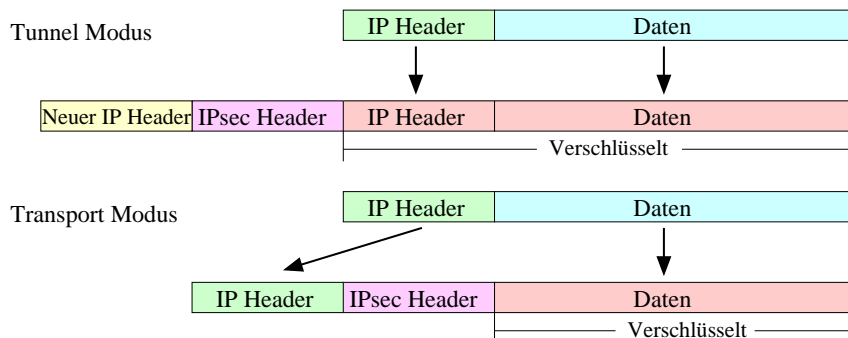


Abbildung C.3: IPsec Betriebsmodi

Abkürzungsverzeichnis

ADK.....	Additional Decryption Key
AES	Advanced Encryption Standard Löst in Zukunft den DES ab.
ANSI	American National Standards Institute http://www.ansi.org/
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation
BER	Basic Encoding Rules
CA.....	Certificate Authority Ein vertrauenswürdiger Dritter, der Zertifikate mit Beurteilungen über verschiedene Attribute erstellt und diese einem Benutzer und/oder dem zugehörigen öffentlichen Schlüssel zuordnet. Stellt das Zertifikat aus.
CAST	C.M. Adams und S.E. Tavares Blockchiffrieralgorithmus.
CBC	Cipher Block Chaining Rückkopplungsmechanismus bei Blockchiffrieralgorithmen.
CFB	Cipher Feedback Siehe CFM.
CFM.....	Cipher Feedback Mode Ein Blockchiffrierer, der als selbstsynchronisierender Stromchiffriercode implementiert wurde.
CN.....	Common Name

- CRL Certificate Revocation List Eine aktualisierte schwarze Liste mit nicht mehr gültigen Zertifikaten.
- CRM Customer Relationship Management One-to-One Marketing
- DER Distinguished Encoding Rules von ASN.1
- DES Data Encryption Standard
- DH..... Diffie-Hellmann Der erste Verschlüsselungsalgorithmus für öffentliche Schlüssel, der diskrete Logarithmen in einem endlichen Feld verwendete. Er wurde 1976 erfunden.
- DLL Dynamic Link Library
- DMZ..... Demilitarized Zone Die demilitarisierte Zone ist ein Bereich zwischen unsicherem Netz und sicherem Netz. Normalerweise befinden sich hier die Webserver für den Internetauftritt einer Firma nach außen hin.
- DN..... Distinguished Name
- DSA Digital Signature Algorithm Ein vom NIST entworfener digitaler Unterschriftenalgorithmus für öffentliche Schlüssel zur Verwendung in DSS.
- DSS..... Digital Signature Standard Ein vom NIST vorgeschlagener Standard (FIPS) für digitale Unterschriften unter Verwendung des DSA.
- ELG El Gamal Asymmetrisches Verschlüsselungsverfahren.
- E-MAIL Elektronische Post Fast schon eingedeutschter Begriff für Textkommunikation im Internet.
- ETH Eidgenössische Technische Hochschule www.ethz.ch
- FIPS..... Federal Information Processing Standard Eine vom NIST veröffentlichte Norm der Regierung der USA.
- FQDN Fully Qualified Domain Name Der komplette Name einer Domäne, also Rechnername und Domäne zusammen. Beispiel: `extranet.innovations.de` statt `extranet`.

- FREES/WAN . Free Secure Wide Area Network Freie Implementierung des IPsec Protokolls um Weitverkehrsverbindungen sicher zu machen.
- FYI For Your Information „Zu Ihrer Information“. Manchmal auch als „For Your Interest“ bezeichnet.
- GNU GNU's not UNIX
- GNUPG GNU Privacy Guard Freie Implementierung der PGP Funktionalität unter der GNU General Public License. Siehe auch Abschnitt 2.1.2.
- HTML Hypertext Markup Language
- HTTP Hypertext Transfer Protocol Port 80.
- HTTPS Hypertext Transfer Protocol Secured Socket Layer Port 443.
- IDEA International Data Encryption Standard
- IE Internet Explorer
- IETF Internet Engineering Task Force <http://www.ietf.cnri.reston.va.us/>
- IMAP Interactive Mail Access Protocol
- IP Internet Protocol
- IPSEC Internet Protocol Security Ein von der IETF in Erwägung gezeigtes Verschlüsselungssystem auf TCP/IP-Ebene.
- ITU International Telecommunication Union <http://www.itu.int/>
- JAR Java Archive
- JDK Java Developers Kit
- KEA Key Exchange Algorithm

- LDAP Lightweight Directory Access Protocol Ein Protokoll zur Unterstützung von Zugriff und schnellen Suchvorgängen in Verzeichnissen mit Informationen, wie beispielsweise Namen, Telefonnummern und Adressen in sonst inkompatiblen Systemen über das Internet.
- LDIF LDAP Data Interchange Format
- MAC Message Authentication Code
- MD4 Message Digest 4 Eine Einweg-Hash-Funktion mit 128-Bit, in der ein einfacher Satz von Bit-Manipulationen mit 32-Bit-Operanden verwendet wird. Sie wurde von Ron Rivest entwickelt.
- MD5 Message Digest 5 Siehe Abschnitt [C.3](#).
- MIME Multipurpose Internet Mail Extensions Eine frei verfügbare Menge von Spezifikationen, mit denen Text in Sprachen mit verschiedenen Zeichensätzen sowie Multimedia-E-Mails zwischen vielen verschiedenen Computer-Systemen mit Internet-E-Mail-Standards ausgetauscht werden können.
- MOSS MIME Object Security Services
- NIST National Institute of Standards and Technology Eine Abteilung des U.S. Department of Commerce (Wirtschaftsministerium der USA). Veröffentlicht Normen bezüglich der Kompatibilität (FIPS). <http://csrc.ncsl.nist.gov/>
- NSA National Security Agency
- NTFS NT File System Das Dateisystem von Windows NT. Es unterstützt die Vergabe von Dateirechten.
- PEM Privacy Enhanced Mail Ein Protokoll für sichere Internet-E-Mail-Nachrichten (RFC 1421-1424). Es enthält Dienste zur Verschlüsselung, Authentifizierung, Nachrichtenintegrität und Schlüsselverwaltung. PEM verwendet ANSI X.509-Zertifikate.
- PFX Personal Information Exchange
- PGP Pretty Good Privacy Siehe auch Abschnitt [2.1.1](#).

- PGPNET..... PGP Netzwerk Eine Erweiterung zu PGP, um Netzwerkverbindungen zu verschlüsseln
- PKCS..... Public-Key Cryptography System
- PKCS#10.... Certification Request Syntax Standard
- PKCS#11.... Cryptographic Token Interface Standard
- PKCS#12.... Personal Information Exchange Syntax Standard
- PKCS#7..... Cryptographic Message Syntax Standard Digital unter-schriebene oder eingebettete Nachricht.
- PKI..... Public Key Infrastruktur Beschreibung in Abschnitt 3.1.
- POP3..... Post Office Protocol Version 3. Port 110.
- RC2..... Rivest Cipher 2
- RC4..... Rivest Cipher 4
- RFC..... Request for Comments Ein IETF-Dokument, aus der Untergruppe FYI RFC (geben Überblicke und Einführungen) oder aus der Untergruppe STD RFC (geben Internet-Normen an). Jeder RFC hat zur Indizierung eine RFC-Nummer, anhand deren er abgerufen werden kann (<http://www.ietf.org>).
- RSA..... Rivest, Shamir, Adleman Algorithmus von Ron Rivest, Adi Shamir und Leonard Adleman, daher der Name. Public Key Verfahren zum Verschlüsseln und Unterschreiben von Daten.
- SHA..... Secure Hash Algorithm Veraltet, siehe SHA-1.
- SHA-1..... Secure Hash Algorithm
- S/MIME..... Secure/Multipurpose Internet Mail Extensions Eine verschlüsselte Variante von MIME.
- SPKAC..... Signed Public Key and Challenge
- SSH..... Secure Shell Sicheres Telnet durch Verschlüsselung und Authentifizierung gegenüber dem Host.

SSL	Secure Socket Layer	Wurde von Netscape zur Gewährleistung von Sicherheit und zur Geheimhaltung im Internet entwickelt.
TCP	Transmission Control Protocol	
URL	Uniform Resource Locator	
USB	Universal Serial Bus	
VB.....	Visual Basic	
VPN	Virtual Private Network	Ermöglicht die Ausdehnung von privaten Netzwerken vom Endbenutzer über ein öffentliches Netzwerk (Internet) direkt bis zum Home-Gateway, wie beispielsweise zum Intranet der Firma.
WSH.....	Windows Scripting Host	

Literaturverzeichnis

- [Abr00] ABREU, ELINOR: *Encoding E-Mail - It's Not for Everyone*. <http://www.thestandard.com/article/display/0,1151,15814,00.html>, Juni 2000.
- [Ash00] ASHLEY, MIKE: *Replacing PGP 2.x with GnuPG*. The Free Software Foundation, 2000.
- [BDR⁺96] BLAZE, MATT, WHITFIELD DIFFIE, RONALD L. RIVEST, BRUCE SCHNEIER, TSUTOMU SHIMOMURA, ERIC THOMPSON und MICHAEL WIENER: *Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security*, 1996.
- [Bun99] BUNDESANZEIGER NR. 213: *Geeignete Kryptoalgorithmen gemäß §17 (2) SigV*, November 1999.
- [CKLW00] CAMPHAUSEN, INGMAR, STEFAN KELM, BRITTA LIEDTKE und LARS WEBER: *DFN-PCA Handbuch: Aufbau und Betrieb einer Zertifizierungsinstanz*. DFN-PCA, März 2000.
- [ECS94] EASTLAKE, D., S. CROCKER und J. SCHILLER: *Randomness Recommendations for Security*. <http://www.faqs.org/rfcs/rfc1750.html>, Dezember 1994.
- [Eng99] ENGELSCHALL, RALF S.: *mod_SSL*. <http://www.modssl.org>, 1999.
- [Ert98] ERTEL, PROF. DR. W.: *Datensicherheit*. Vorlesungsskript, Fachhochschule Ravensburg-Weingarten, 1998.
- [Har00] HARTMANN, JOSEF: *Evaluierung von Sicherungs- und Verschlüsselungsalgorithmen für das EH&SWeb-Interface*. Diplomarbeit, Fachhochschule Ravensburg-Weingarten, 2000.

- [HC98] HARKINS, D. und D. CARREL: *The Internet Key Exchange (IKE)*. <http://www.faqs.org/rfcs/rfc2409.html>, November 1998.
- [Kon00] KONIETZKA, MICHAEL: *Entwurf und Realisierung eines Konzepts zur Sicherung des E-Mail-Verkehrs für Kunden von Internet-Dienstleistern unter Verwendung von X.509-Zertifikaten*. Diplomarbeit, Universität Karlsruhe (TH), <http://www.konietzka.de/diplom>, 2000.
- [Kra98] KRAUSE, R.: *Anforderung an die Datensicherheit bei Kommunikationsnetzen*. Daimler-Benz Aerospace, Dornier, 1998.
- [Luc99a] LUCKHARDT, NORBERT: *Pretty Good Privacy: Teil 1: Einstieg in das Web of Trust*. c't, 12:212–214, Juni 1999.
- [Luc99b] LUCKHARDT, NORBERT: *Pretty Good Privacy: Teil 2: Schlüsselfragen und -antworten*. c't, 13:208–210, Juni 1999.
- [Net98] NETSCAPE COMMUNICATIONS CORPORATION, <http://developer.netscape.com/docs/manuals/security/sslin/index.htm>: *Introduction to SSL*, 1998.
- [Net01] NET SECURITY, <http://www.net-security.org/text/bugs/979055938,89332,.shtml>: *PGP 7.0 signature verification vulnerability*, Januar 2001.
- [NS01] NIESING, MATTHIAS und KLAUS SCHMEH: *Schlüssel des Vertrauens: Digitale Ausweise im Internet*. c't, 4:224–231, Februar 2001.
- [Ryd00] RYDER, JOSHUA: *Establishing Email Validity*. <http://securityportal.com/cover/coverstory20001225.printerfriendly.html>, 2000.
- [Sch97] SCHNEIER, BRUCE: *Angewandte Kryptographie: Protokolle, Algorithmen und Sourcecode in C*. Addison-Wesley, 1997.
- [Sch98] SCHMEH, KLAUS: *Safer Net: Kryptographie im Internet und Intranet*. dpunkt, 1998.
- [Sec00] SECURITY, JAVA: *JDK 1.2 - Signed Applet Example*. <http://java.sun.com/security/signExample12/>, Januar 2000.

- [Uni00] UNIVERSITY, CARNEGIE MELLON: *CERT® Advisory CA-2000-18 PGP May Encrypt Data With Unauthorized ADKs*. <http://www.cert.org/advisories/CA-2000-18.html>, 2000.
- [WL] WEIS, RÜDIGER und STEFAN LUCKS: *Bigger is better! Anmerkungen zur Schlüssellängendiskussion*. <http://www.informatik.uni-mannheim.de/~rweis/cc1999/WeisLucksDatenschl\%euderFaktor.html>.
- [Wob00] WOBST, REINHARD: *Schlüsselbefreiung: Das Ende des RSA-Patents und die Folgen*. iX, Oktober 2000.