DFN-PCA
Handbuch

Aufbau und Betrieb einer Zertifizierungsinstanz

Ingmar Camphausen
Stefan Kelm
Britta Liedtke
Lars Weber

März 2000

DFN-PCA · Vogt-Kölln-Straße 30 · D-22527 Hamburg

Zusammenfassung

Die oberste Zertifizierungsstelle (CA) des Deutschen Forschungsnetzes (DFN) will mit diesem Handbuch für Zertifizierungstellen die Nutzung von Public-Key-Verfahren erleichtern und fördern. Das Handbuch beschreibt die Abläufe bei der Planung, Einrichtung und dem Betrieb einer CA für eine exemplarische DFN-Mitgliedseinrichtung. Es werden dabei nicht nur technische, sondern ebenso organisatorische und rechtliche Aspekte (Datenschutz, Signaturgesetz) behandelt. Um Anwendern dieses Handbuches den Aufbau und die Inbetriebnahme einer eigenen Zertifizierungsstelle zu vereinfachen, werden entsprechende detaillierte Schritt-für-Schritt-Anleitungen für die meisten Routine-Tätigkeiten einer CA gegeben.

Da sich dieses Handbuch in erster Linie an Anwender aus dem universitären und Forschungsbereich wendet, stehen Open-Source-Programme zur Zertifizierung und zum Betrieb der CA wie PGP, OpenSSL und Apache im Mittelpunkt der Betrachtungen. Zwei der drei Teile dieses Handbuches erklären daher ausführlich die Installation, Konfiguration und Benutzung von OpenSSL und des Apache HTTP-Servers in einer Zertifizierungsstelle.

Das Handbuch wird durch einen umfangreichen Anhang abgerundet, der u.a. eine generische Low-Level-Zertifizierungsrichtlinie, Beschreibungen einiger nützlicher Zertifizierungswerkzeuge und Beispiel-Konfigurationen für PGP, OpenSSL und Apache enthält.

Abstract

The top-level ('root') certification authority (CA) of the German Research Network (DFN) provides this manual in order to simplify the introduction and use of public key based infrastructures. The process of planning, establishing and running a certification authority is described, whereby a fictive member institution of the DFN Association is used to exemplify the discussion. In addition to step-by-step guides for most routine proceedures, a thorough analysis of organizational and legal aspects is provided, whereby the German laws and the EU Directives pertaining to data privacy and digital signatures are considered.

As this handbook is primarily intended for the research and educational communities, open-source software is used by the CA. Detailed instructions are therefore provided, with which the reader is lead through the process of compiling, installing, configuring and using the Apache HTTP server with OpenSSL in a CA environment.

The manual's appendix contains a generic low-level certification policy, descriptions of some useful certification tools, example configurations for PGP, OpenSSL and Apache.

Vorwort

Ziel des Projektes „PCA im DFN – Aufbau einer Policy Certification Authority (PCA) für das Deutsche Forschungsnetz“[1] (DFN-PCA) ist der kontinuierliche Aufbau einer Vertrauensinfrastruktur zur Beglaubigung, Verteilung und Verwaltung von öffentlichen Schlüsseln, wie sie für gängige kryptographische Verfahren gebraucht werden. Dabei wird im Projekt ein “bottom-up”-Ansatz verfolgt, bei dem nach und nach mehr DFN-Mitgliedseinrichtungen eine entsprechende eigene Infrastruktur in ihrem Hause aufbauen, die dann durch Verknüpfung mit der DFN-PCA und untereinander zu einem immer dichter werdenden Netz wächst.

Aus der Praxis des DFN-PCA-Projektes, den Anrufen und E-Mails vieler Nachfrager und dem persönlichen Kontakt auf Tagungen wie dem DFN-CERT-Workshop oder dem DFN-Arbeitskreis „Security“ wissen die PCA-Mitarbeiter, daß es einen großen Bedarf an Informationen zum Aufbau und Betrieb einer Zertifizierungsstelle wie auch zur Einrichtung und dem Betrieb eines sicheren (SSL-)Webservers gibt – und bislang immer noch zu wenig entsprechende Dokumentation.

Die rege Resonanz, auf die die Diplomarbeit des Projekt-Mitarbeiters Herrn CAMPHAUSEN zudem gestoßen war, und die diversen Veränderungen besonders auf dem Gebiet der rechtlichen Rahmenbedingungen, die sich seit der Abgabe der Arbeit Anfang 1999 ergeben hatten, ließen es aus Sicht des PCA-Projektes sinnvoll erscheinen, eine überarbeitete und um das Gebiet Server-Zertifizierung erweiterte Fassung dieser Arbeit als Handbuch herauszugeben.

Zugleich liegen mit diesem Handbuch nun erstmals das bisherige *OpenSSL-Handbuch* und das bisherige *SSL-Apache-Handbuch* des DFN-PCA-Projektes, die bislang nur als Online-Dokumente existierten[2], auch in gedruckter Form vor, wodurch sich hoffentlich der Kreis ihrer Anwenderinnen und Anwender noch vergrößert.

Zielgruppe

Das vorliegende Handbuch wendet sich an Personen, die mit dem Aufbau einer Zertifizierungsstelle oder einer kompletten Infrastruktur für Public-Key-Zertifikate betraut sind oder dies erwägen, ebenso wie an Administratoren, die einen WWW-Server um sichere Zugriffsmöglichkeiten via SSL erweitern wollen. Gleichermaßen kann es aber dem gestandenen CA-Operator als Gelegenheit dienen, seine praktischen Erfahrungen zu hinterfragen oder vielleicht doch noch die eine oder andere

[1]Das Projekt wird unter der Nr. TK 602–SD 111 aus Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) durch den Verein zur Förderung eines Deutschen Forschungsnetzes e.V. – DFN-Verein – finanziert.

[2]`http://www.pca.dfn.de/dfnpca/certify/ssl/handbuch/`

Anregung zu bekommen. Das Handbuch ist sowohl für die konzeptionelle Vorbereitung einer eigenen Zertifizierungsstelle oder -infrastruktur als auch als Nachschlagewerk oder Checkliste bei der praktischen täglichen Arbeit in einer CA oder mit einem eigenen SSL-Server gedacht.

Der erste Teil des Handbuches befaßt sich hauptsächlich mit konzeptionellen und organisatorischen Aspekten des Aufbaus und Betriebs einer Zertifizierungsinstanz, er sollte insofern auch für „Neulinge“ auf diesem Gebiet und für Leserinnen und Leser, die keine Programmier- oder Administrationserfahrung mit Computern haben, verständlich sein, zumal in einem Einführungskapitel die wichtigsten Grundlagen und Begriffe erläutert werden. (Kenntnisse und Erfahrungen mit Verschlüsselung, Public-Key-Verfahren oder -Zertifizierung, mit PGP, S/MIME oder SSL schaden natürlich nicht, sind aber keine Voraussetzung für die Arbeit mit diesem Handbuch.)

Teil II und III sind hingegen stärker technisch ausgerichtet; zu ihrem Verständnis sind Grundlagen-Kenntnisse in UNIX und in der Übersetzung und Installation von Software-Paketen sowie von Public-Key-Zertifizierung erforderlich.

Entstehungsgeschichte

Dieses Handbuch basiert auf mehreren Dokumenten und entstand unter Beteiligung aller Mitarbeiter des DFN-PCA-Projektes: Teil I sowie ein Großteil der Anhänge sind eine Weiterentwicklung und Verallgemeinerung des Konzeptes für eine Zertifizierungsstelle für die FU Berlin, das INGMAR CAMPHAUSEN 1999 als Diplomarbeit geschrieben hat [Cam99]. In die Überarbeitung floß u.a. das Feedback ein, das er von STEFAN KELM als Mitbetreuer der Diplomarbeit bekommen hatte. Bei Teil II und III sowie den dazugehörigen Anhängen handelt es sich um fortgeschriebene Versionen des OpenSSL- und des SSL-Apache-Handbuches der DFN-PCA, die zur Zeit beide von LARS WEBER betreut werden. BRITTA LIEDTKE steuerte schließlich ihre praktische Erfahrung aus der Zertifizierungsarbeit im DFN-PCA-Projekt insbesondere zu den Abschnitten über PGP und den Anhang mit der Aufwandsabschätzung bei.

Dieses Handbuch wurde im März 2000 zum ersten DFN-PCA-Tutorium als Teil der Tutoriumsunterlagen für die Teilnehmer erstmalig aufgelegt.

Struktur der Arbeit

Dieses Handbuch ist in drei Teile nebst einem Anhang gegliedert:

Teil I enthält das Konzept zu Planung, Aufbau und Betrieb einer eigenen Public-Key-Zertifizierungsstelle, die nach den Richtlinien (der *Policy*) der DFN-PCA betrieben und in die DFN-Zertifizierungsstruktur eingebunden werden soll

Teil II ist das Handbuch zur Software *OpenSSL*, deren Installation, Konfiguration und Benutzung zu Zertifizierungszwecken beschrieben wird

Teil III erläutert, wie das Programm OpenSSL aus Teil II benutzt werden kann, um den verbreiteten Web-Server *Apache* auch als HTTPS-Server einsetzen zu können, so daß mit SSL/TLS gesicherte Verbindungen zu ihm aufgebaut werden können

Die Schwerpunkte der einzelnen Kapitel:

TEIL I

Kapitel 1 beschreibt die Motivation, die diese Arbeit entstehen ließ

Kapitel 2 erläutert die theoretischen Grundlagen von Public-Key-Verschlüsselung, digitalen Signaturen, Zertifikaten und Certification Authorities, soweit sie für das Verständnis des Handbuches wichtig sind

Kapitel 3 stellt eine Bestandsaufnahme dessen dar, was in der Praxis an Standards, Verfahren, Software und rechtlichen Regelungen im Zusammenhang mit Zertifizierung existiert

Kapitel 4 enthält das Konzept für die exemplarische Zertifizierungsstelle einer fiktiven Universität („UNI"); wobei die Entwurfsziele erläutert, Lösungsansätze vorgeschlagen und begründet und sowohl rechtliche als auch kryptographische und organisatorische Aspekte des Zertifizierungsbetriebs diskutiert werden

Kapitel 5 gibt detaillierte Hinweise, wie bei der praktischen Realisierung und dem Betrieb einer eigenen Zertifizierungsstelle verfahren werden sollte und worauf dabei besonders zu achten ist

Kapitel 6 zeigt mögliche zukünftige Entwicklungen und Perspektiven für eine Zertifizierungsstelle auf

TEIL II

Kapitel 7 erläutert die Konfiguration, Übersetzung und Installation von OpenSSL

Kapitel 8 beschreibt die von OpenSSL unterstützten X.509-Zertifikat-Erweiterungen sowie deren Bedeutung und Nutzung

Kapitel 9 befaßt sich mit der Konfigurationsdatei von OpenSSL und den darin zur Verfügung stehenden Optionen

Kapitel 10 schildert, wie mit OpenSSL Zertifizierungs-Requests und Zertifikate erzeugt werden können

Kapitel 11 führt in den Umgang mit Widerrufslisten für Public-Key-Zertifikate (CRLs) unter OpenSSL ein

Kapitel 12 gibt Erfahrungen aus Tests und dem praktischen Einsatz von OpenSSL zur Zertifizierung wieder, wobei insbesondere auf die Nutzung von OpenSSL-Zertifikaten mit den beiden verbreiteten WWW-Browsern Netscape Communicator und Internet Explorer eingegangen wird

Kapitel 13 rundet den OpenSSL-Teil mit einigen nützlichen Tools ab, die OpenSSL ideal ergänzen

TEIL III

Kapitel 14 Installation und Einrichtung des Apache-Webservers als HTTPS-Server unter Verwendung von OpenSSL

Kapitel 15 Betrieb des SSL-Apache

ANHANG

Anhang A enthält Empfehlungen von Büchern, Artikeln, Online-Dokumenten, WWW-Hyperlink-Sammlungen und Mailinglisten, die im Zusammenhang mit dem Thema dieses Handbuches erwähnenswert sind

Anhang B ist ein Bezugsquellen-Nachweis für Programme und Geräte, die im Hauptteil genannt werden

Anhang C führt ein Beispiel dafür an, wie die Konfigurationsdatei des Programms PGP 2.6.3in für dessen Einsatz in einer Zertifizierungsstelle aussehen kann

Anhang D listet eine Reihe undokumentierter PGP-Optionen auf

Anhang E Beispiel für eine OpenSSL-Konfigurationsdatei für den OpenSSL-Einsatz in einer Zertifizierungshierarchie

Anhang F dient als Nachschlagewerk für die diversen Kommandos und Parameter, die das OpenSSL-Paket umfaßt

Anhang G gibt ein Beispiel für die Konfigurationsdatei `httpd.conf` des Apache-Webservers

Anhang H stellt einige Software-Werkzeuge für die Analyse von X.509-Zertifikaten und von PGP-Schlüsseln vor

Anhang I liefert Beispiele für Probleme aus der Zertifizierungspraxis sowohl bei Browser-Zertifikaten als auch beim Zertifizieren von PGP-Schlüsseln

Anhang J wirft einen Blick auf den zeitlichen und finanziellen Aufwand, der mit den verschiedenen Auf- und Ausbaustufen einer Zertifizierungsstelle der vorgeschlagenen Form vermutlich verbunden wäre

Anhang K gibt eine exemplarische “Low-Level”-Policy für die DFN-PCA-konforme Arbeit einer Zertifizierungsstelle wieder

Anhang L umfaßt eine kleine Auswahl von „Standard-E-Mails“ als Formulierungsvorschläge für Nachrichten, wie sie bei der Arbeit einer Zertifizierungsstelle immer wieder verschickt werden müssen

Anhang M nennt die Anschriften und sonstigen Kontakt-Informationen einiger Institutionen, die im Zusammenhang mit Public-Key-Zertifizierung und Verschlüsselung von Interesse sein könnten

WICHTIGER HINWEIS

DIESES DOKUMENT (DAS „CA-HANDBUCH") ENTSTAND IN EINEM DFN-PROJEKT AN DER UNIVERSITÄT HAMBURG UND WURDE NACH BESTEM WISSEN UND GEWISSEN VERFASST. DENNOCH WIRD KEINE HAFTUNG FÜR DIE KORREKTHEIT, VOLLSTÄNDIGKEIT ODER ANWENDBARKEIT DER HIER BESCHRIEBENEN INFORMATIONEN UND DER VORGESCHLAGENEN MASSNAHMEN ÜBERNOMMEN. FERNER KANN KEINE HAFTUNG FÜR EVENTUELLE SCHÄDEN, ENTSTANDEN DURCH DIE ANWENDUNG DER IN DIESEM DOKUMENT BESCHRIEBENEN ANWEISUNGEN, ÜBERNOMMEN WERDEN. DIE VERANTWORTUNG FÜR DIE VERWENDUNG DER HIER BESCHRIEBENEN KONZEPTE, VERFAHREN UND PROGRAMME LIEGT ALLEIN BEI DEN JEWEILIGEN ANWENDERN.

Noch eine Bitte…

Wir würden uns sehr freuen, wenn Sie, sehr geehrte Anwenderin, sehr geehrter Anwender, uns mit Ihrem Feedback helfen würden, das vorliegende CA-Handbuch weiterzuentwickeln und noch besser auf Ihre Wünsche und Bedürfnisse abzustimmen! Bitte lassen Sie uns wissen, welche Abschnitte Sie bei Ihrer Arbeit mit diesem Handbuch als besonders hilfreich oder womöglich als entbehrlich empfunden haben, was Sie vielleicht darin vermißt haben oder welche Stellen aus Ihrer Sicht anders formuliert werden sollten.

Auch wenn Sie sonstige Anregungen, Korrekturen oder Hinweise auf eventuelle Fehler in diesem Handbuch haben, zögern Sie bitte nicht, uns diese – gerne auch verschlüsselt – zukommen zu lassen. Am Ende dieses Handbuches finden Sie dazu die Mailadresse, Faxnummer und Postanschrift des DFN-PCA-Projektes sowie unsere Schlüsselinformationen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I

Aufbau und Betrieb einer Zertifizierungsinstanz

Kapitel 1

Motivation

Aufgabe der „Allianz von Technik und Datenschutz“ (Simitis) ist vor allem die Entwicklung und Verbreitung von ‘privacy enhancing technologies’. Sie soll datenschutzfördernde Sicherheitsarchitekturen anbieten ... .

— STEFAN WALZ [Wal97, S. 28]

Ohne Verschlüsselung gibt es bei der Kommunikation per E-Mail keine Vertraulichkeit. Als Beleg müßten die beiden folgenden Zitate eigentlich ausreichen, um jeden Skeptiker davon zu überzeugen, daß abgehört wird:

- 12,6% der 1000 führenden Unternehmen in den USA haben Eingriffe in ihr E-Mail-System entdeckt; mindestens jede zehnte Nachricht werde unbefugt abgefangen oder mitgelesen – laut PGP-Entwicklern ist es sogar jede vierte [CZ98e].
- Der US-Geheimdienst NSA hört europaweit E-Mails ab [Ebe98, Wri98a, S. 19] und betreibt ein weltweites Abhörnetz, das vor allem gegen nicht-militärische Ziele gerichtet ist [Hag97]. Dabei kooperieren EU und USA bei diesem weltweiten Überwachungssystem [StW97].

Und während sich der Zwischenbericht des EU-Parlamentes noch über das NSA-Abhörnetz empört

> “If even half of these allegations are true then the European Parliament must act to ensure that such powerful surveillance systems operate to a more democratic consensus now that the Cold War has ended. [...] No proper Authority in the USA would allow a similar EU spy network to operate from American soil without strict limitations, if at all. [...] [The] European Parliament is advised to set up appropriate independent audit and oversight procedures and that any effort to outlaw encryption by EU citizens should be denied until and unless such democratic and accountable systems are in place, if at all.” [Wri98b]

plant die EU ziemlich im Verborgenen und bislang kaum beachtet mit „ENFOPOL“ schon etwas Ähnliches [SH98b].

Ohne Verschlüsselungsverfahren (Kryptographie) ist nicht nur die Vertraulichkeit der elektronischen Kommunikation gefährdet, es ist, wie z.B. DAMKER et al. demonstrieren [DFS96] – und wie es vermutlich viele Internet-Nutzer bereits selber erlebt haben (etwa in Form einer E-Mail von `schroeder@kanzleramt.de` o.ä.) – auch keine sichere Feststellung des Urhebers (Authentizität) oder der Unverfälschtheit einer Nachricht (Integrität) möglich.

Und dennoch fehlt es immer noch an der unterstützenden Infrastruktur für den breiten Einsatz von geeigneten kryptographischen Verfahren. Was RÜDIGER GRIMM 1997 beklagte, ist, wenn auch vielleicht in etwas abgeschwächter Form, immer noch aktuell:

> „Es werden global kompatible Zertifizierungsinfrastrukturen gebraucht, in denen jeder Bürger jeden öffentlichen Signaturschlüssel glaubwürdig verifizieren kann, ohne vorher mit seinem Besitzer einen bilateralen Prüfvorgang durchlaufen zu müssen. [...] Warum aber gibt es diese Zertifizierungsinfrastruktur nicht längst, wenn sie doch so dringend gebraucht wird? Die Verfahren sind seit fast zehn Jahren bekannt. Hier gibt es ein Henne-Ei-Problem, indem die Zertifizierungsinfrastruktur einerseits und die Sicherheitsanwendungen andererseits aufeinander warten. [...] Unternehmer sind nicht bereit, in den Aufbau von Zertifizierungsinstanzen zu investieren, wenn man mit den neuen Signaturschlüsseln nichts anfangen kann: Sie warten darauf, daß es ... interessante Anwendungen gibt, die eine Nachfrage nach einer Schlüsselverwaltung erzeugen. Umgekehrt investiert aber kein Anwender in teure Zusatzfunktionen in die schon bestehenden Internetanwendungen für Email und Word Wide Web [sic], wenn dafür Schlüssel gebraucht werden, die es nirgends gibt ...“ [Gri97, S. 217]

Das Projekt „Policy Certification Authority für den DFN“ (DFN-PCA) zielt darauf ab, innerhalb des Deutschen Forschungsnetzes eine Zertifizierungsinfrastruktur aufzubauen. Als ein Teil davon sind Zertifizierungsinstanzen in den DFN-Mitgliedseinrichtungen vorgesehen, die mithelfen (werden), den von GRIMM genannten „Teufelskreis“ zu durchbrechen. Mit dem hier vorgelegten CA-Handbuch will die DFN-PCA interessierten Einrichtungen und Personen im DFN, aber auch außerhalb des Forschungsnetzes, eine Handreichung bieten, die ihnen dabei helfen soll, eine eigene Zertifizierungsinstanz aufzubauen und zu betreiben. Auf diese Weise kann und soll eine breite Zertifizierungsinfrastruktur entstehen, die in möglichst vielen der DFN-Mitgliedseinrichtungen verankert ist.

Gerade *Forschungs*einrichtungen müßten ein originäres Interesse daran haben, Know-how im Umgang mit Public-Key-Verschlüsselung und elektronischen Signaturen zu sammeln, denn sie können bereits heute vom Einsatz dieser Techniken ganz praktisch profitieren: 1998 hat die EU den Auftrag für die Einrichtung und den Betrieb einer Zertifizierungsstelle ausgeschrieben, die die *elektronische* Einreichung von Geboten und Anträgen zu Forschungs- und Entwicklungs-Programmen der EU ermöglichen soll [EUK98a]. Diese Stelle ist inzwischen eingerichtet und kann von jeder Einrichtung in Anspruch genommen werden, die EU-Projekt-Proposals auf elektronischem Weg einreichen möchte.[1]

Auch bei der Verwaltung einer eigenen Internet-Domain lassen sich Public-Key-Verfahren sinnvoll einsetzen und tragen zu einer größeren Sicherheit bei: Änderungswünsche an den Einträgen

[1] `http://fp5-csp.org`

in der RIPE[2]-Datenbank, über die die europäischen Internetnummern vergeben werden, können – und sollten – inzwischen mit digitalen PGP-Signaturen authentifiziert werden, um gefälschten Änderungsaufträgen einen Riegel vorzuschieben. [Zsa99]

Auch die neue Bundesregierung verschließt sich diesen Vorzügen nicht, nachdem der ehemalige Innenminister Kanther noch ein vehementer Verfechter staatlicher Abhörbefugnisse war. In ihrem „Eckwertepapier zur Kryptopolitik“[3] vom 2. Juni 1999 beschließt die Bundesregierung u.a.:

> „Die Bundesregierung beabsichtigt nicht, die freie Verfügbarkeit von Verschlüsselungsprodukten in Deutschland einzuschränken. Sie sieht in der Anwendung sicherer Verschlüsselung eine entscheidende Voraussetzung für den Datenschutz der Bürger, für die Entwicklung des elektronischen Geschäftsverkehrs sowie den Schutz von Unternehmensgeheimnissen.“

Mit der „Initiative des Bundesministeriums für Wirtschaft und Technologie für mehr Sicherheit in der Informationsgesellschaft“[4] will sie darüberhinaus „... Projekte durch[zu]führen, u.a. um die Transparenz technischer Vorgänge zu verbessern und dadurch die Nutzer zu mehr Eigenverantwortlichkeit anzuregen. [...] Gerade ... bei den privaten Internet-Nutzern besteht ... noch erheblicher Informationsbedarf [...]“ [BMW99b]. Im Rahmen dieser Initiative fördert das Ministerium auch die Integration von starker Open-Source-Kryptosoftware („Gnu Privacy Guard“[5]) in verbreitete Mailprogramme.[6]

Erfreulicherweise kann jede Sicherungsinfrastruktur für digitale Signatursysteme, die technisch zuverlässige Signierschlüssel bereitstellt, auch dazu benutzt werden, Verschlüsselungsschlüssel (Konzelationsschlüssel) sicher auszutauschen [HP96]. Insofern hilft eine Zertifizierungsinfrastruktur, den Nutzern entsprechen der Forderungen der Datenschutzbeauftragten im Sinne ihres informationellen Selbstbestimmungsrechtes Hilfsmittel und Verfahren an die Hand zu geben, mittels derer sie die Authentizität und Integrität, aber auch die Vertraulichkeit ihrer Kommunikation selber sicherstellen können.

Ein weiterer Grund, sich für den verstärkten Gebrauch von Verschlüsselung einzusetzen und die Voraussetzungen für einen solchen auf breiter Basis zu schaffen, liegt in der Langlebigkeit einmal verschickter E-Mails begründet. Festplattenplatz ist billig geworden, und so werden elektronische Mitteilungen (auch unwichtige) länger aufgehoben als früher vergleichbare Notizen oder Briefe, die irgendwann aus Platzgründen aussortiert wurden. Heute können solche Mitteilungen nicht nur in großer Zahl archiviert, sondern bei Bedarf auch blitzschnell nach Suchkriterien durchsucht und die Ergebnisse dann ohne größere Umstände elektronisch weitergeleitet oder auf Knopfdruck einem großen Empfängerkreis zugänglich gemacht werden.

Hinzu kommt noch (zumindest bei der Kommunikation mit öffentlichen Einrichtungen) die Gefahr, daß Dritte über Akteneinsichtsregelungen Einblick in die persönlichen Schreiben bekommen könnten [Mee99] – ein Umstand, dessen sich sicher nur die wenigsten von uns bewußt sind, wenn sie mit einer Behörde Schrift- bzw. E-Mailverkehr führen. Dieser Punkt betrifft uns in der Bundesrepublik

[2] `http://www.ripe.net`
[3] dokumentiert unter `http://www.dud.de/dud/doks/kreg990602.htm`
[4] `http://www.sicherheit-im-internet.de`
[5] `http://www.gnupg.org`
[6] `http://www.sicherheit-im-internet.de/download/pm-gnupg.pdf`

(noch) nicht in gleichem Maße wie beispielsweise die US-Amerikaner, die schon lange ein Akteneinsichtsrecht ("Freedom of Information", FOI) haben. 1998 hat jedoch mit Brandenburg das erste Land der Bundesrepublik ein entsprechendes Gesetz [AIG98] erlassen, inzwischen sind auf Länderebene Berlin[7] und Schleswig-Holstein[8] dem Brandenburger Vorbild gefolgt, und auf Bundesebene hat die neue Regierungskoalition die Einführung eines Akteneinsichtsrechts in Deutschland als Ziel in ihrem Koalitionsvertrag[9] festgeschrieben.

Es gibt also auch in Deutschland immer mehr Gründe, verschlüsselt zu mailen – „Nicht immer, aber immer öfter!" ; –)

[7] `http://www.datenschutz-berlin.de/recht/bln/ifg/ifg.htm`

[8] `http://www.rewi.hu-berlin.de/Datenschutz/DSB/SH/material/themen/presse/ldsgifg.htm`

[9] online dokumentiert auf dem Webserver der SPD unter `http://www.spd.de/politik/koalition/`, Abschnitt IX „Sicherheit für alle – Bürgerrechte stärken", Punkt 13 „Beteiligungsrechte"

Kapitel 2

Theoretische Grundlagen

Die Schlüssel werden golden und silbern gemalt.
Der goldene ... bedeutet die Gewalt, zu lösen und zu öffnen.
Der silberne ... versinnbildet die Gewalt, zu binden und zu schließen.
— BRUNO B. HEIM*, Wappenbrauch und Wappenrecht*
in der Kirche, zitiert nach [Rab70, S. 214]

Nachfolgend werden die theoretischen Grundlagen der Public-Key-Verschlüsselung sehr kurz umrissen. Eine gewisse Vertrautheit mit Public-Key-Verfahren ist daher von Vorteil beim Arbeiten mit diesem Handbuch.[1]

2.1 Public-Key-Verschlüsselung

DIFFIE und HELLMAN stellten 1976 ein für die Kryptographie revolutionäres Verfahren vor [DH76], das sie zurecht mit “New Directions in Cryptography” überschrieben.[2] Sie brachen darin mit den bis dahin gängigen Verschlüsselungsverfahren, bei denen Sender und Empfänger beide denselben geheimzuhaltenden Schlüssel sowohl zum Ver- als auch zum Entschlüsseln benutzen (sogenannte *single key* oder auch ‘symmetrische’ Verschlüsselungsverfahren). Sie schlugen vielmehr vor, daß Sender und Empfänger jeweils ein Schlüsselpaar erzeugen, das aus zwei Komponenten besteht, die in einem bestimmten mathematischen Bezug zueinander stehen.

Soll nun eine Nachricht vertraulich übermittelt werden, so verschlüsselt der Absender sie mit der einen Komponente des Empfängerschlüssels, die dafür öffentlich bekannt sein muß – dem öffentlichen Schlüssel oder auch *public key*. Entschlüsseln kann diese Nachricht nun nur derjenige, der über die andere Komponente des Schlüssels verfügt. Diese sollte also geheimgehalten werden, damit kein Dritter unbefugt die verschlüsselte Nachricht lesen kann. Diese geheime Komponente wird als *secret key* oder auch als *private key* bezeichnet. Zur Veranschaulichung wird oft das Bild eines

[1] Einen guten, verständlichen Überblick über die grundlegenden Mechanismen bietet z.B. GEHRING [Geh98].

[2] Der britische Geheimdienst CESG hat kürzlich behauptet, bereits einige Jahre zuvor das Konzept der Public-Key-Verschlüsselung entdeckt, es aber nicht öffentlich gemacht zu haben [Ell97]. In der nicht-militärischen Kryptologie gelten daher nach wie vor Diffie und Hellman als die „Erfinder“ der Public-Key-Verschlüsselung.

Briefkastens herangezogen. Der Absender kann eine Nachricht hineinwerfen (= verschlüsseln), sie aber nicht mehr wieder herausholen (= entschlüsseln). Das kann nur der Inhaber des passenden „privaten“ Schlüssels zu diesem Briefkasten – eben der Inhaber des Secret Keys. Aufgrund der unterschiedlichen Schlüssel, die zum Ver- und zum Entschlüsseln verwendet werden, spricht man statt von Public-Key-Verschlüsselung gelegentlich auch von „asymmetrischer“ Verschlüsselung.

Auf diese Weise muß nur jeder seinen öffentlichen Schlüssel bekannt geben, und schon können ihm alle Leute, auch wildfremde, die ihm nie zuvor persönlich begegnet sind, verschlüsselt und damit vertraulich Nachrichten zukommen lassen. Das ist ein wesentlicher Fortschritt gegenüber allen Verfahren, die man bis dahin kannte, bei denen jedes Sender-Empfänger-Paar einen separaten Schlüssel ausmachen mußte – bei einer größeren Zahl von Kommunikationsteilnehmern ein fast unlösbares Unterfangen. Bei den Public-Key-Verfahren hingegen ist es egal, ob 10 oder 10 000 Menschen miteinander kommunizieren wollen, jeder muß nur (s)einen öffentlichen Schlüssel bekannt geben, quasi seinen „Briefkasten aufhängen“, um in diesem Bild zu bleiben.

Es gibt dabei nun nur ein kleines, anderes Problem: Woran erkennt man, ob ein Briefkasten wirklich zu demjenigen gehört, dessen Name darauf prangt? Oder, auf die Verschlüsselung bezogen: Wie kann ein Absender einer Nachricht sicher sein, daß er wirklich den authentischen öffentlichen Schlüssel des beabsichtigten Empfängers vorliegen hat und nicht irgendeinen anderen Schlüssel, den ihm jemand untergeschoben hat? Bei einer großen Zahl von (potentiellen) Kommunikationsteilnehmern kann man sich nicht mehr mit jeder oder jedem persönlich treffen, um den öffentlichen Schlüssel des jeweils anderen sozusagen „aus erster Hand“ zu erhalten. Man spricht in diesem Zusammenhang auch vom „Schlüsselverteilungsproblem“, das sich glücklicherweise durch organisatorische Vorkehrungen stark reduzieren läßt (s. 2.3).

Ungünstigerweise sind die Rechenoperationen, auf denen Public-Key-Verfahren üblicherweise aufbauen, sehr aufwendig und dieses Verfahren entsprechend langsam, besonders, wenn längere Nachrichten damit verschlüsselt werden sollen. Auch lassen sich die Grundoperationen für Public-Key-Verfahren, das Rechnen mit sehr großen Zahlen (mehr als 100 Stellen), nicht so gut auf einem Mikroprozessor ausführen wie die Basisoperationen, aus denen einige der besten herkömmlichen, symmetrischen Verschlüsselungsverfahren aufgebaut sind. (Beispiele für bekannte Vertreter dieser Verfahren sind der *Data Encryption Standard*, DES, und der *International Data Encryption Algorithm*, IDEA [DES77, LM91].) Um nun den Vorteil der Public-Key-Verschlüsselung – einfachere Schlüsselverteilung – mit den Vorzügen der symmetrischen Verschlüsselung – sehr schnell und besonders effizient in Hardware realisierbar – zu kombinieren, bedient man sich einer Kombination aus beiden: Es wird ein zufälliger Sitzungsschlüssel (*session key*, quasi ein „Einmal-Schlüssel“) erzeugt, der nur für die symmetrische Verschlüsselung einer einzigen Nachricht verwendet wird. Die vertraulich zu übermittelnde Nachricht wird z.B. mit DES unter diesem Sitzungsschlüssel chiffriert. Der Sitzungsschlüssel selber ist typischerweise nur wenige Bytes groß und wird nun wiederum selber mittels asymmetrischer Verschlüsselung unter Verwendung des Public-Keys der beabsichtigten Empfängerin der Nachricht verschlüsselt. (Das geht trotz Public-Key-Verschlüsselung schnell, weil der Session-Key so kurz ist.) Beide verschlüsselten Informationen, der Chiffretext der ursprünglichen Nachricht und der des zu ihrer Verschlüsselung verwendeten Einmalschlüssels, werden nun zusammen an die Empfängerin übermittelt. Diese entschlüsselt zuerst den Public-Key-verschlüsselten Session-Key, wozu sie ihren geheimen Schlüssel benutzt. Keine andere Person, auch kein „Lauscher“, könnte auf diese Weise den Sitzungsschlüssel ermitteln, da ja – sorgfältigen Umgang mit dem Private-Key vorausgesetzt – niemand sonst über den zum öffentlichen Schlüssel der Empfängerin korrespondierenden geheimen Schlüssel verfügt. Die Empfängerin erhält auf diese Weise den Sitzungsschlüssel und kann dann mit dessen Hilfe die herkömmlich verschlüsselte Nachricht wieder lesbar machen. Eine solche Kombination aus herkömmlicher und Public-Key-Verschlüsselung bezeichnet man auch als „Hybrid“-Verfahren.

Die bekannteste konkrete Umsetzung des Public-Key-Ansatzes dürfte das RSA-Verfahren sein [RSA78], das nach seinen drei Erfindern RONALD RIVEST, FIAT SHAMIR und LEONARD ADLEMAN benannt ist und das die Grundlage für die Arbeit der gleichnamigen Firma (RSA Data Security, heute eine Tochter von Security Dynamics) darstellt. Beispiele für Programme, in denen Public-Key-Verschlüsselung vorkommt, und zwar meist als Teil einer Hybrid-Verschlüsselung, also in Kombination mit symmetrischen Verschlüsselungsverfahren, sind das Verschlüsselungsprogramm *Pretty Good Privacy* (PGP) [Zim95] und alle WWW-Browser, die die verschlüsselte Übertragung via *Secure Socket Layer* (SSL) [FKK96] unterstützen.

2.2 Digitale Signaturen

Manche der Public-Key-Verschlüsselungsverfahren verfügen über eine zusätzliche Eigenschaft, die sie noch für einen anderen Zweck einsetzbar macht: Ver- und Entschlüsselungsoperation sind kommutativ, d.h. das Resultat ist dasselbe, egal ob erst ver- und dann entschlüsselt wird oder umgekehrt. Man kann also auch eine Nachricht zuerst mit dem geheimen Schlüssel „entschlüsseln“ und sie danach mit dem öffentlichen Schlüssel „verschlüsseln“ und erhält am Ende auch bei dieser Ausführungsreihenfolge wieder den ursprünglichen Klartext. Dies kann man sich zu Nutze machen, um die Urheberschaft eines Dokumentes zu kennzeichnen, es quasi zu „unterschreiben“: Der Absender, der eine Nachricht digital unterzeichnen möchte, verschlüsselt diese mit seinem Private-Key, auf den niemand außer ihm Zugriff hat. Das Ergebnis ist eine Nachricht, die aussieht, als wäre sie verschlüsselt. Sie erweckt aber nur den Anschein als ob, denn jeder kann mit dem Public-Key des Absenders diese „Verschlüsselung“ wieder rückgängig machen und erhält dann den Klartext. Durch die Entschlüsselungsoperation mit dem öffentlichen Schlüssel erfolgt quasi eine Prüfung der Unterschrift bzw. des vermuteten Absenders. Ergibt diese Operation sinnvollen Klartext, so kann die vorhergehende Verschlüsselungsoperation nur mit dem korrespondierenden Private-Key durchgeführt worden sein – und auf den sollte, normalen Umgang mit geheimen Schlüsseln vorausgesetzt, nur der Inhaber zugreifen können. Also muß diese Nachricht von ihm stammen. Man könnte sagen, er hat sie mit seinem geheimen Schlüssel *signiert*.

Da, wie bereits in 2.1 erwähnt, Public-Key-Verfahren ziemlich rechenaufwendig sind, zu unterschreibende Nachrichten aber mitunter auch sehr groß werden können, greift man auch hier wieder zu einem Trick, um nicht nur möglichst kurze Datenfolgen mit dem Public-Key-Verfahren bearbeiten zu müssen: Statt eine ganze, möglicherweise sehr große, Nachricht mit dem geheimen Schlüssel zu „verschlüsseln“, um sie zu signieren, wird eine Art Prüfsumme, eine mathematische „Kurzfassung“ der Nachricht verschlüsselt, die dann zusammen mit der eigentlichen Nachricht übermittelt wird. Dazu bedient man sich sogenannter *Hash*-Funktionen – man spricht auch von *message digests* –, die große Nachrichten mit beliebiger Länge auf eine sehr kurze Bitfolge abbilden können. Der Empfänger einer auf diese Weise signierten Nachricht muß nun mehrere Schritte ausführen, um die digitale Signatur zu der Nachricht prüfen zu können: Der Text der eigentlichen Nachricht wird von der angefügten Signatur getrennt. Dann wird über diesem Text nach demselben Verfahren wie beim Absender die Prüfsumme berechnet. Das Ergebnis dieser Berechnung wird mit der Zahl verglichen, die man erhält, indem man die signierte Prüfsumme, die ebenfalls Bestandteil der übermittelten Nachricht war, mit dem Public-Key des Absenders „entschlüsselt". Sind beide Zahlen identisch, heißt das, daß der Absender die Hash-Funktion auf genau den Text angewendet haben muß, der auch beim Empfänger (zusammen mit der Signatur) angekommen ist. Weicht hingegen die vom Empfänger ermittelte Prüfsumme von der ab, die der Absender als Signatur mit der Nachricht mitgeschickt hat, so ist entweder die Nachricht unterwegs verändert worden oder zum Prüfen der Signatur wurde der falsche Public-Key verwendet, und die Nachricht hat einen anderen Urheber.

Um außerdem noch erkennen zu können, *wann* ein Dokument auf diese Weise digital unterschrieben wurde, wird außer der Prüfsumme meistens noch ein Zeitstempel mit „verschlüsselt". Dadurch läßt sich dann beim Empfänger auch erkennen, ob eventuell eine alte Nachricht von einem Angreifer erneut eingespielt wurde (eine sog. *replay attack*).

Auch für die Prüfung einer digitalen Signatur ist es unerläßlich, daß der Empfänger den *authentischen* Schlüssel des vermuteten Absenders vorliegen hat, denn nur dann kann er mit Gewißheit prüfen, ob eine Nachricht von dieser Person stammt. Damit sind wir wieder beim Problem der verläßlichen Verteilung der authentischen öffentlichen Schlüssel angelangt.

2.3 Zertifizierungsstellen

Die direkte Übergabe eines Public-Keys bei einem persönlichen Kontakt (z.B. auf einer Diskette) oder die Übermittlung durch einen vertrauenswürdigen Kurier oder über eine gesicherte Standleitungsverbindung ist die naheliegendste Art und Weise, den authentischen öffentlichen Schlüssel eines Kommunikationspartners zu erhalten. Sie entspricht dem in [ISO] beschriebenen Verfahren *Public-Key Distribution without a Trusted Third Party – Public-Key Transport Mechanism 1: using an Authenticated Channel*.

Die Übermittlung des Public-Keys auf unsicherem Wege, z.B. per E-Mail, oder der Abruf von einem Keyserver oder einer Homepage mit anschließender Überprüfung beispielsweise durch telefonischen Vergleich des Schlüssels oder seiner kryptographischen Prüfsumme (bei Schlüsseln sagt man auch "Fingerprint" dazu; das ist letztlich nichts anderes als der Hash-Wert oder der *message digest* des betreffenden Schlüssels) analog zu *Public-Key Transport Mechanism 2: Authentication using a Second Channel* [a.a.O.] ist eine weitere – einfachere und daher häufigere – Art und Weise des authentischen Schlüsselaustausches. Darunter fällt auch der Austausch des Fingerprints auf Visitenkarten bei einem direkten persönlichen Kontakt (= der zweite, authentische Kommunikationskanal) o.ä. und das anschließende Vergleichen dieses Fingerprints mit der errechneten Prüfsumme des via unsicherem Kanal erhaltenen Schlüssels.

Aber auch diese Vorgehensweise ist nicht immer praktikabel; oft wird man mit neuen, fremden Kommunikationspartnern Public-Key-gesichert kommunizieren wollen, mit denen ein direkter Kontakt (vorerst) u.U. nicht möglich ist und deren Stimme man nicht – oder nicht gut genug – für eine telefonische Überprüfung des Fingerprints kennt.

In diesem Fall kommen die Zertifizierungsstellen (engl. *certification authorities* – CAs) ins Spiel:

> “The introduction of a CA reduces the problem of authentic user public key distribution to the problem of authentic distribution of the CA’s public key, at the expense of a trusted center (the CA).” [ISO]

Eine Zertifizierungsstelle – synonym wird auch der Begriff ‘Zertifizierungsinstanz’ verwendet – stellt digitale Bestätigungen (Zertifikate, siehe 2.4) aus. In ihnen bestätigt sie mit ihrer digitalen Signatur die Bindung eines bestimmten Public-Keys (und damit implizit auch des zugehörigen Private-Keys) an eine Person, Institution oder Instanz (die beispielsweise auch ein Rechner sein kann), die in dem Zertifikat namentlich benannt wird. Die Zertifizierungsstelle bietet damit also einen Dienst ganz ähnlich einer notariellen Beglaubigung dafür, daß ein bestimmter Public-Key und eine bestimmte Person zusammengehören.

Es wird dadurch eine Komplexitätsreduktion möglich: statt vieler individueller Public-Keys muß nur noch der öffentliche Schlüssel der Zertifizierungsstelle authentisch verteilt werden. Liegt er einer Person vor, kann diese mit seiner Hilfe die digitalen Unterschriften der Zertifizierungsstelle unter jedem der von ihr ausgestellten Zertifikate überprüfen und sich damit der Echtheit und Unverfälschtheit aller von ihr ausgestellten Zertifikate vergewissern. Stimmt die Unterschrift, dann kann aus dem betreffenden Zertifikat der öffentliche Schlüssel der darin genannten Person abgelesen werden. Allerdings muß man dafür dem Aussteller des Zertifikates, also der betreffenden CA, dahingehend vertrauen, daß sie immer korrekt zertifiziert und keine falschen Bestätigungen ausstellt. Hat man dieses Vertrauen in die Arbeit der CA nicht, so bringt die Verschiebung des Schlüsselverteilproblems auf den CA-Schlüssel keinen wirklichen Vorteil.

Sehr wichtig, ja geradezu essentiell für die Arbeit einer Zertifizierungsstelle ist also ihre Glaubwürdigkeit bei möglichst vielen Nutzern. Nur wenn die Anwender bereit sind, sich auf die Bestätigung eines fremden öffentlichen Schlüssels durch die betreffende CA zu verlassen, hat diese eine Arbeitsgrundlage. Also muß es im Interesse der Zertifizierungsstelle sein, alles zu tun, um sich ihre entsprechende Reputation zu erhalten – oder eine solche gegebenenfalls auch erst zu erarbeiten (siehe dazu auch 4.16.2).

Im einzelnen läßt sich der Vorgang der Public-Key-Zertifizierung durch eine Zertifizierungsstelle in die folgenden Teilschritte gliedern:

- Erzeugung eines Schlüsselpaares (beim Zertifikatnehmer selbst oder in der Zertifizierungsstelle)
- Registrierung und Identifizierung des Zertifikatnehmers direkt bei der CA oder in einer Außenstelle von ihr (im letzteren Fall spricht man auch von einer Registrierungsstelle, *registration authority*, RA), ggf. erfolgt dort auch die Übergabe einer Kopie des öffentlichen Schlüssels vom Zertifikatnehmer an die CA/RA
- gegebenenfalls: Übermittlung der Daten aus dem Zertifizierungsantrag von der RA an die Zertifizierungsstelle
- Zertifizierung des Schlüssels (sofern alle Voraussetzungen dafür wie Schlüssellänge, eindeutiger Name des Schlüsselinhabers usw., erfüllt sind)
- Aushändigung oder Übermittlung des Zertifikates an den Schlüsselinhaber

- Veröffentlichung des Zertifikates durch die Zertifizierungsstelle mittels geeigneter Verzeichnis- oder Verteildienste

Hinzu kommen können später noch Dienste wie die Re-Zertifizierung eines Schlüssels, wenn ein bereits erteiltes Zertifikat abzulaufen droht, oder auch die Sperrung des Zertifikates, wenn entsprechende Gründe dafür vorliegen.

Nicht jede Zertifizierungsstelle muß alle diese Tätigkeitsfelder abdecken; es ist gut möglich, daß sich manche Stellen auf einen Teil dieser Aufgaben beschränken und sie andere Teilaufgaben, beispielsweise den Betrieb eines Verzeichnisdienstes, an Partnerunternehmen oder andere Anbieter delegieren. Die Zertifizierung von Schlüsseln als zentraler Tätigkeitsbereich einer Zertifizierungsstelle wird allerdings immer dazugehören.

Einrichtungen, die nicht nur Zertifizierungsdienste anbieten, sondern die beispielsweise zusätzlich auch Schlüssel für Nutzer generieren, Sicherungskopien geheimer Kundenschlüssel verwahren oder Zeitstempeldienste offerieren, werden häufig auch als *Trustcenter* oder auch als *Trusted Third Party* (TTP), also als ein „vertrauenswürdiger Dritter“ bezeichnet (siehe dazu auch 4.16.1). Ihnen muß, im Unterschied zu einer reinen Zertifizierungsstelle, auch der Zertifikat*nehmer* – also derjenige, dessen Schlüssel zertifiziert wird – Vertrauen entgegenbringen, denn er muß sich darauf verlassen, daß das Trustcenter wie zugesichert nach der Schlüsselerzeugung und -aushändigung alle Spuren des geheimen Schlüssels vernichtet bzw. daß das Trustcenter die hinterlegte Sicherheitskopie des Private-Key auch wirklich geheimhält.

2.4 Public-Key-Zertifikate

2.4.1 Funktion

Public-Key-Zertifikate stellen die Verbindung her zwischen einer Identität (einer Person oder einer Instanz) und einem öffentlichen Schlüssel. Indirekt wird damit aufgrund des Zusammenhangs zwischen öffentlichem und geheimem Schlüssel auch ein Bezug zwischen Identität und Secret-Key hergestellt. Zertifikate helfen dabei, auf unsicheren Transportwegen öffentliche Schlüssel authentisch auszutauschen. Derartige Zertifikate sind eine wichtige Voraussetzung/ein wichtiges Hilfsmittel beim Einsatz von Public-Key-Kryptographie.

2.4.2 Allgemeiner Aufbau

Public-Key-Zertifikate enthalten mindestens einen öffentlichen Schlüssel, den Namen (die Identität) der betreffenden Person oder Instanz und eine digitale Signatur, die den Urheber dieser Bestätigung nachprüfbar macht und zugleich unbemerkte Modifikationen des Zertifikates verhindert (Authentizitäts- und Integritätsschutz). Darüber hinaus können Public-Key-Zertifikate noch diverse andere zusätzliche Bestandteile umfassen wie etwa

- Seriennummer des Zertifikates
- Bezeichnung des Schlüssels, der zum Signieren benutzt wurde

- Algorithmus, der zum Signieren des Zertifikates benutzt wurde
- Algorithmus und Parameter, mit denen der zertifizierte Schlüssel benutzt werden soll
- Angaben, zu welchen Zwecken der zertifizierte Schlüssel eingesetzt werden darf
- Gültigkeitsdauer bzw. frühestes Gültigkeitsdatum sowie Verfallsdatum des Zertifikates
- zusätzliche Angaben über den Zertifikatnehmer, z.B. zu dessen Beruf, Qualifikation, Zulassungen usw.
- Haftungsbeschränkungen oder -obergrenzen
- Alternative Namen oder Kommunikationsadressen des Zertifikatausstellers
- Alternative Namen oder Kommunikationsadressen des Zertifikatnehmers
- Verweise auf Zertifikate für den Unterschriftenschlüssel des Ausstellers
- Verweise auf Sperrlisten (siehe 2.5), auf denen dieses Zertifikat gegebenenfalls geführt wird
- Verweise auf Möglichkeiten zur Online-Abfrage des Zertifikatstatus'

Konkrete Beispiele verbreiteter Zertifikat-Formate werden im nächsten Kapitel in Abschnitt 3.1 vorgestellt.

2.5 Widerrufslisten

Es sind Situationen denkbar, in denen ein Zertifikataussteller noch während der Gültigkeitsdauer eines von ihm ausgestellten Zertifikates die darin gegebene Bestätigung für ungültig erklären möchte, beispielsweise weil im Nachhinein bekannt wurde, daß das Zertifikat vom Zertifikatnehmer unter Angabe falscher Daten (Name usw.) erschlichen wurde oder weil der zum zertifizierten öffentlichen Schlüssel gehörende geheime Schlüssel einem Angreifer in die Hände gefallen („kompromittiert“) ist. Zu diesem Zweck werden Zertifikat-Widerrufslisten verwendet (*certificate revocation lists*, CRLs), auch 'Sperrlisten' genannt. Auf ihnen werden üblicherweise die Seriennummern derjenigen Zertifikate einer Zertifizierungsinstanz aufgeführt, die für ungültig erklärt werden und deren regulärer, im Zertifikat angegebener Gültigkeitszeitraum noch nicht abgelaufen ist. (Nach Ablauf dieses Zeitraumes besitzt das Zertifikat in jedem Fall keine Gültigkeit mehr und muß daher auch nicht weiter auf der Sperrliste geführt werden.)

Diese Sperrlisten werden, genau wie die Zertifikate, von der ausgebenden Stelle digital signiert, um Verfälschungen erkennen zu können. In der Regel enthalten sie zusätzlich einen Zeitstempel und eine Angabe, wann die nächste Aktualisierung dieser Sperrliste stattfindet. (Anhand dieser Daten läßt sich gegebenenfalls erkennen, ob eine veraltete Version der Liste vorliegt.)

Vergleichen lassen sich solche Sperrlisten am ehesten mit den Listen mit Seriennummern gestohlener oder in Lösegeldzahlungen verwendeter Geldscheine oder mit Listen der Nummern gestohlener und deshalb gesperrter Sparbücher, Schecks oder Kreditkarten.

Die Zertifikat-Sperrlisten werden ganz analog zu den genannten Beispielen aus dem Geschäftsleben verwendet: Jemand, der sich auf ein Zertifikat zur Authentifizierung eines öffentlichen Schlüssels verläßt, sollte zuvor unbedingt auch die entsprechende Widerrufsliste des betreffenden Zertifikatausstellers konsultieren und sich vergewissern, daß das fragliche Zertifikat dort nicht aufgeführt, also gesperrt wurde.

Sperrlisten haben unter anderem den Nachteil, daß sie immer eine gewisse zeitliche „Unschärfe“ aufweisen: Ein Dritter, der sich auf ein Zertifikat verlassen will, muß u.U. das Erscheinen der nächsten Ausgabe der betreffenden Sperrliste abwarten, um ganz sicher gehen zu können, daß das Zertifikat nicht in der Zeit seit dem Erscheinen der vorigen Widerrufsliste gesperrt wurde. Als eine Ergänzung bzw. eine Alternative zu Sperrlisten, die diesen Nachteil vermeidet, wurde daher das Verfahren erdacht, den Widerrufsstatus eines Zertifikates online genau in dem Moment abzufragen, in dem der Schlüssel aus dem Zertifikat verwendet werden soll. Für diese Anfrage wird die Seriennummer des fraglichen Zertifikates an die Anfrage-Schnittstelle der Zertifizierungsinstanz übermittelt, die daraufhin als Antwort „gültig“ oder „ungültig“ liefert. Die IETF hat ein entsprechendes Verfahren für derartige Status-Abfragen in ihrem RFC 2560 *X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP* [AAG+99] beschrieben.

2.6 Zertifizierungsrichtlinien („Policy“)

> „Das Problem der Zertifizierung öffentlicher Schlüssel läßt sich in folgenden Fragen zusammenfassen: Wer zertifiziert? Wer zertifiziert den Zertifizierer? Nach welchen Regeln und Richtlinien wird zertifiziert? Was wird zertifiziert? – Auf jeden Fall die Bindung zwischen Person, Name und Schlüssel. Aber auch die Qualität der Schlüssel, die zugehörige Zertifizierungskette, die Zertifizierungsregeln? Der soziale Bezug, Rollen und Zugriffsrechte? Welche Dienste gehören zu einer Zertifizierung? Auch die Schlüsselgeneration? [Die] Rekonstruktion verlorener Schlüssel (und wie das)? Widerruf mißbrauchter oder verlorener Schlüssel? Speicherung von Schlüsseln? Ausgabe von Chipkarten mit geheimen Schlüsseln? Welche Garantien gibt eine Zertifizierung? Welche Haftung?“ [Gri95, S. 121]

Antworten auf alle oder wenigstens die meisten dieser Fragen sollten für eine bestimmte Zertifizierungsstelle deren Zertifizierungsrichtlinien (die sogenannte *Policy*, international oft auch *Certification Practice Statement*, CPS) geben.

2.6.1 Zweck einer Policy

Eine nachvollziehbare und nach festgelegten Regeln erfolgende Zertifizierung ist es ja gerade, die eine CA von einem beliebigen, unbekannten Zertifizierer unterscheidet. Das Arbeiten nach selbstgesetzten (oder von außen – z.B. durch Gesetze oder Selbstverpflichtungen einer Branche – vorgegebenen), offengelegten Regeln ermöglicht es einem Dritten, der ein Zertifikat nutzen will, erst, die Glaubwürdigkeit und Verläßlichkeit des Zertifizierers einzuschätzen und sich zu entscheiden, inwieweit er dessen Aussage vertrauen will bzw. kann.

> „TTPs [Trusted Third Parties] benötigen zur Begründung ihrer Vertrauenswürdigkeit eine veröffentlichte Policy, die eine klare Darstellung der Aufgaben und Sicherheitsanforderungen umfaßt und möglichst benutzerüberprüfbar realisiert ist.“ [FHK95, S. 2]

Damit eine Zertifizierungs-Policy diesem Anspruch gerecht werden kann, muß sie auf der einen Seite ausführlich genug sein, damit sich Nutzer ihrer Zertifikate ein Bild von der Arbeitsweise der CA machen können, auf der anderen Seite darf sie aber auch nicht zu umfangreich und detailliert sein, weil sie dann unübersichtlich würde. In Abschnitt 2.6.2 werden einige Policy-Beispiele genannt, die jeweils unterschiedliche Aspekte dieser Anforderungen besonders betonen.

2.6.2 Beispiele

Eine Policy dient nicht nur der Information Dritter, die Zertifikate nutzen wollen, sondern auch zur Absicherung der Zertifizierungsinstanz, die nach ihren Vorgaben zu zertifizieren verspricht. Schließlich geht es bei der Formulierung einer Policy auch darum, Haftungsregeln festzulegen (soweit dies rechtlich zulässig ist) bzw. diese im Sinne der Zertifizierungsstelle zu gestalten und für den Fall von Regreßforderungen und Rechtsstreitigkeiten eine günstige Ausgangsposition zu schaffen. Andererseits darf natürlich vor allem bei einer kommerziell arbeitenden Zertifizierungsstelle die Policy die Interessen und Bedürfnissen der Kunden und von deren Kommunikationspartnern nicht völlig ignorieren, sonst beraubt sich die CA selber ihrer Geschäftsgrundlage. Wenn der Policy-Text nun aber auch den Anforderungen von Juristen genügen muß, dürfte denen im Zweifelsfall Vorrang vor der Verständlichkeit für technische oder juristische Laien gegeben werden. Das kann dann zu solchen CPS’ wie dem von VeriSign Inc. [Ver97] führen, das zwar auf mehr als 50 Seiten sehr genau alle Aspekte der Zertifizierung durch VeriSign und der daraus resultierenden Haftung durch das Unternehmen beschreibt, dadurch aber so umfangreich geworden ist, daß vermutlich kaum ein Nutzer je die Richtlinien vollständig zur Kenntnis nehmen wird.

Das Public-Key-Verfahren bzw. das Zertifikat-Format (siehe auch 3.1), auf das sich die Policy bezieht, kann die Verständlichkeit einer Policy ebenfalls beeinflussen: Ein komplexes Verfahren oder Format mit vielen Optionen wird in den Zertifizierungsrichtlinien sicher nur schwer präzise und verständlich zugleich zu beschreiben sein. Darüber hinaus macht es auch einen Unterschied, ob die Zertifizierung als Dienst für Dritte angeboten oder eher für eigene Zwecke intern betrieben wird. Beide Punkte werden an dem Vorschlag von Dirk Fox für eine „PGP-Policy für Unternehmen“ [Fox98] deutlich, die auf nur zwei Seiten knapp und übersichtlich Zertifizierungs- und Umgangsregeln für die gesicherte firmeninterne und -externe elektronische Kommunikation beschreibt. Ihr kommt zugute, daß sie auf eine überschaubare, geschlossene Benutzergruppe (noch dazu mit Weisungsbefugten) abzielt und sich nicht an einen offenen Nutzerkreis richtet.

Kapitel 3

Public-Key-Zertifizierung in der Praxis

3.1 Gebräuchliche Zertifikat-Formate

In der praktischen Anwendung spielen hauptsächlich zwei Zertifikat-Formate für öffentliche Schlüssel eine Rolle: Das in der ITU-Empfehlung X.509[1] in der Version 3 von 1997 [ITU97] standardisierte Zertifikat-Format und das Nachrichten- und Zertifikat-Format des Programms PGP (“Pretty Good Privacy”) [Zim95] und der dazu kompatiblen Software.

Der Kontrast zur doch recht großen Zahl von Programmen, die Public-Key-Verfahren verwenden, erklärt sich dadurch, daß die meisten von ihnen auf das Format aus X.509 setzen, zumindest was das Austauschformat für Zertifikate angeht. X.509 hat sich hier aufgrund seines Status’ als Standard und dank der Flexibilität, die mit der nunmehr dritten Version dieses Formates erreicht wird, durchgesetzt.

Der grundsätzliche Aufbau von Zertifikaten nach diesen beiden Format-Definitionen wird nachfolgend erläutert.

3.1.1 X.509

Als Teil der X.500-Standard-Serie, die den Verzeichnisdienst (*directory service*) beschreibt, entstand 1991 der CCITT- (heute ITU-)Standard X.509, der ein auf dem X.500-Verzeichnisdienst basierendes *Authentication Framework* beschrieb. In ihm wurde ein Austauschformat für Public-Key-Zertifikate definiert, das im Laufe der Jahre weiter verfeinert und ergänzt wurde:

X.509v1 Die erste Fassung des X.509-Standards von 1991 sah ein einfaches Format für die Zertifikate vor. Neben den typischen Informationen Schlüsselinhaber, Public-Key und Unterschrift des Ausstellers (jeweils noch nebst der Angabe, welcher Algorithmus zu Grunde liegt), die sozusagen das „minimale Zertifikat“ beschreiben, umfaßte es lediglich eine Seriennummer, einen Ausstellernamen und die Angabe eines zeitlichen Gültigkeitsbereiches.

X.509v2 Mit der Überarbeitung des X.509-Standards und der daraus resultierenden Version von 1993 änderte sich am Zertifikat-Format nicht viel. Es wurde lediglich eine Versionsnummer

[1] identisch mit ISO-Standard ISO/IEC 9594-8

hinzugefügt, die dazu diente, Zertifikate im neuen Format von X.509v1-Zertifikaten unterscheiden zu können. Außerdem wurde je ein zusätzliches, optionales Feld für Inhaber (*subject*) und Aussteller (*issuer*) mit aufgenommen (*subjectUniqueIdentifier* respektive *issuerUniqueIdentifier*), in dem jeweils weitere Informationen zu dem Betreffenden angegeben werden können.

X.509v3 Die nochmals überarbeitete Fassung des Standards von 1997 [ITU97] sieht dagegen erhebliche Erweiterungen des X.509-Zertifikat-Formates vor. Es wird dadurch flexibler, um beispielsweise besser die von den X.500-Konventionen abweichenden Internet-Namen für Rechner, Mailadressen oder WWW-Seiten anstelle eines X.500-Namens für den Inhaber oder Aussteller in ein Zertifikat aufnehmen zu können.

In X.509v3 sind generische Zertifikat-Erweiterungen (*extensions*) vorgesehen, die es ermöglichen, bei Bedarf auf einfache Weise zusätzliche Informationen in das Format mit aufzunehmen. Es kann dadurch individuell an die Bedürfnisse z.B. in einem Unternehmen oder die Anforderungen einer bestimmten Anwendung angepaßt werden, ohne dafür den Standard ändern oder von ihm abweichen zu müssen.

Diese neuen Erweiterungen können als ‘kritisch’ markiert werden, dann muß eine Anwendung, die das betreffende Zertifikat verarbeitet, diese Erweiterung auch verarbeiten können; andernfalls muß sie das Zertifikat als ungültig interpretieren. Es wurden für einige häufig auftretende Anforderungen bereits einige Extensions in der neuen Version des Standards definiert. Weiterhin wurden Komponenten für zusätzliche Informationen zu den jeweiligen Schlüsseln vorgesehen (Abb. 3.1 zeigt den schematischen Aufbau eines X.509v3-Zertifikates).

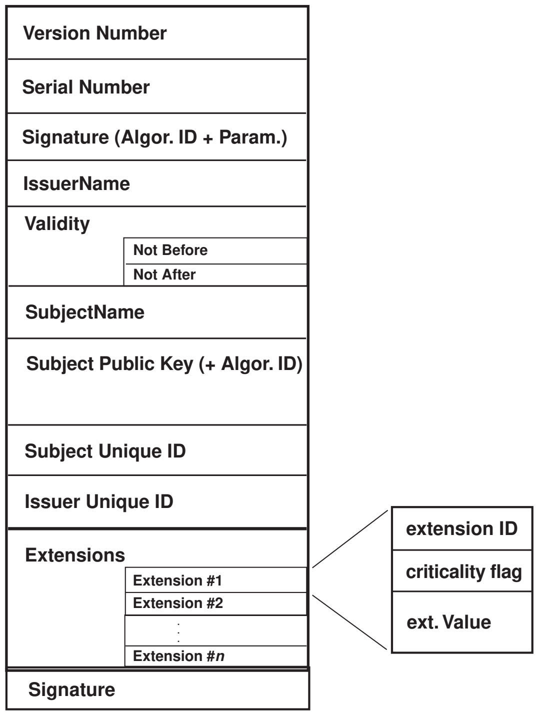


Abb. 3.1: Struktur eines X.509v3-Zertifikates

In Abschnitt H.1 zeigt ein Beispiel, wie man sich bei einem konkreten X.509-Zertifikat diese Datenstrukturen mit verschiedenen Tools anzeigen lassen kann.

3.1.2 PGP

PGP ("Pretty Good Privacy") bezeichnet ein bestimmtes Programm (bzw. inzwischen etliche Varianten und Versionen eines Programms) und zugleich den von diesem Programm gesetzten Standard, was das Format verschlüsselter Nachrichten, der entsprechenden Schlüssel usw. angeht. Erst *nachträglich* wurden die von PGP benutzten Nachrichten-, Schlüssel- und Zertifikatformate in einem RFC [ASZ96] beschrieben und damit „standardisiert“. Man könnte dieses Format auch als „klassisches“ PGP-Format beschreiben, da es schon von 1992 stammt. Mittlerweile existiert mit dem verabschiedeten OpenPGP-RFC [CDFT98] eine Weiterentwicklung, die Anregungen von X.509v3 und aus dem jahrelangen Einsatz von PGP – inzwischen auch in PGP-Zertifizierungsstellen – aufgreift (siehe zu OpenPGP auch Abschnitt 6.1.1).

Der OpenPGP-Standard sieht ein flexibles neues Signatur-Format mit Sub-Paketen vor, die zusätzliche Informationen zu der betreffenden Unterschrift enthalten können, z.B. den Beginn oder das Ende der Gültigkeitsdauer einer Unterschrift, den vom Schlüsselinhaber bevorzugten symmetrischen Verschlüsselungsalgorithmus usw. Eine detaillierte Beschreibung dieses Formates würde allerdings den Rahmen dieses Dokumentes sprengen. (Die Definition des OpenPGP-Paket-Formates im OpenPGP-RFC umfaßt 25 Seiten...)

Zur besseren Veranschaulichung des PGP-Schlüssel-Formates hier zunächst die typische Ausgabe der Kommandozeilen-Version von PGP 2.6. Angezeigt werden alle Schlüssel, die den Teilstring `ingmar` in einer Benutzerkennung enthalten:

```
$ pgp -kvv ingmar
[...]
Type Bits/KeyID    Date       User ID
pub  1024/4F570BA3 1993/06/18 Ingmar Camphausen <ingmar@aurora.in-berlin.de>
sig       BB1D9F6D             ct magazine CERTIFICATE <pgpCA@ct.heise.de>
sig       28CBE7F5             Felix von Leitner <felix@ccc.de>
sig       D5327CB9             wietse venema <wietse@wzv.win.tue.nl>
sig       4F570BA3             Ingmar Camphausen <ingmar@aurora.in-berlin.de>
                              Ingmar Camphausen <ingmar@acm.org>
sig       4F570BA3             Ingmar Camphausen <ingmar@aurora.in-berlin.de>
                              Ingmar Camphausen <ingmar@in-berlin.de>
sig       5CF94C71             Alexander Geschonneck <geschonneck@rz...
sig       476CB3CD             in-ca@in-berlin.de SIGN EXPIRE:1998-12-31 ...
sig       C379A331             Matthias Bruestle <m@mbsks.franken.de> (SIGN)
sig       4F570BA3             Ingmar Camphausen <ingmar@aurora.in-berlin.de>
```

Wir haben es hier also zu tun mit

- dem eigentlichen Public-Key mit der numerischen Kennung (KeyID) 4F570BA3 und 1024 bit Länge, erstellt am 18. Juni 1993
- drei verschiedenen Benutzerkennungen (UserIDs) zu diesem Schlüssel, die sich in diesem Fall nur in den Mailadressen unterscheiden (was recht typisch ist, sofern die Schreibweise des Namens nicht variiert oder dieser weggelassen wird):

  - `Ingmar Camphausen <ingmar@aurora.in-berlin.de>`
  - `Ingmar Camphausen <ingmar@acm.org>`
  - `Ingmar Camphausen <ingmar@in-berlin.de>`

- Signaturen von anderen PGP-Nutzern unter einzelnen der Benutzerkennungen, die die Verbindung zwischen dem Schlüssel und der jeweiligen UserID bestätigen

Etwas schematischer dargestellt sieht die Struktur dieses öffentlichen Schlüssels einschließlich der Zertifikate so aus wie in Abb. 3.2.

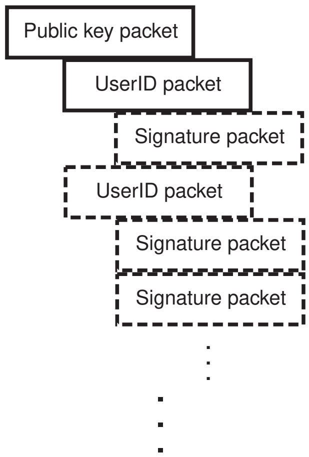


Abb. 3.2: PGP Public-Key – Grobstruktur einschließlich Zertifikaten

Die Teilstrukturen Public-Key-Paket und Signature-Paket sind selbst wieder strukturiert. Sie bestehen aus den in Abb. 3.3 gezeigten Komponenten, wobei Teilbild b sozusagen das PGP-„Zertifikat“-Format beschreibt. (Im Anhang H.2.1 findet sich ein Beispiel dafür, wie diese Strukturen mittels eines Tools angezeigt werden können.)

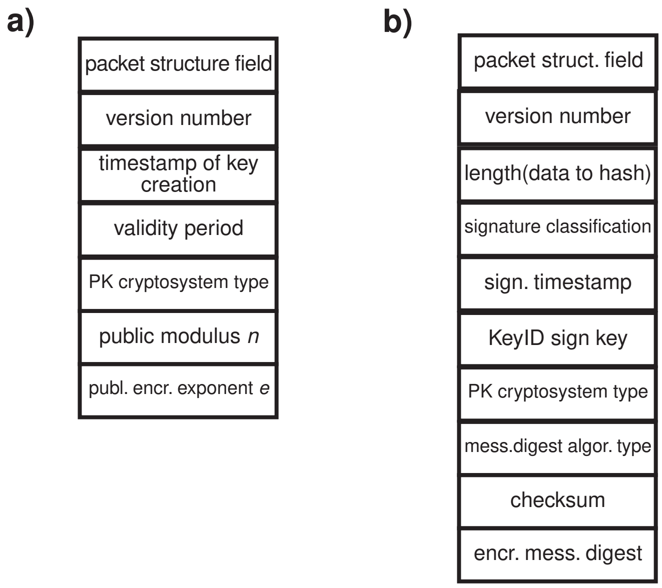


Abb. 3.3: Struktur a) eines PGP Public-Keys b) eines PGP-Key-Zertifikates

Einzelheiten des alten, „klassischen“ PGP-Zertifikat- und Schlüsselformates sind in den Büchern von STALLINGS [Sta94] und KAUFMAN et al. [KPS95] beschrieben.

3.2 Zertifizierungsstellen-Software

Nachdem im vorigen Abschnitt die gängigen Zertifikat-Formate dargestellt wurden, soll hier ein kurzer Überblick über entsprechende CA-Software gegeben werden, mit der solche X.509- bzw. PGP-Zertifikate ausgestellt werden können.

3.2.1 X.509

Da sich X.509 als Standard etablieren konnte und von fast allen Anbietern außer PGP/Network Associates benutzt wird, ist die Zahl der Zertifizierungsprogramme, die auf X.509 basieren, so groß, daß eine umfassende Darstellung den Rahmen dieses Handbuches sprengen würde. Einen Eindruck von der Vielzahl der Angebote Vermittelt der Teil „Toolkits“ der “PKI Link Page”[2] des DFN-PCA-Projektes.

Kommerzielle Produkte

Einen ersten Überblick über (kommerzielle) Zertifizierungssoftware, die X.509-Zertifikate unterstützt, bieten z.B. [Rai98], [GS00], [Cam98a, S. 32 f.] oder die Produktliste der am Projekt SPHINX (siehe 3.4.1.2) beteiligten Hersteller[3].

Frei verfügbare Software

Erwähnt werden muß an dieser Stelle zuerst das wohl (auch) für diese Zwecke meistgenutzte frei und im Sourcecode verfügbare Programm **SSLeay** von ERIC A. YOUNG bzw. dessen Nachfolger **OpenSSL** – in der Hauptsache eine SSL/TLS-Implementierung, die aber darüber hinaus auch Routinen für das Erzeugen und (in Grundzügen) das Verwalten von X.509v3-Zertifikaten bietet. SSLeay/OpenSSL wurde und wird komplett außerhalb der USA entwickelt und und ist insofern von deren Exportbeschränkungen nicht betroffen.

Ein weiteres vielversprechendes Open-Source-Projekt ist **OpenCA**[4]. In diesem Projekt wird an einem Toolkit für typische Operationen zur Verwaltung von X.509-Zertifikaten gearbeitet. Ziel ist

> “... studying and refining the security scheme that guarantees the best model to be used in a CA and developing software to easily setup and manage a Certification Authority.”

pyCA[5] baut auf OpenSSL auf. Es bietet eine Sammlung von in der Sprache Python geschriebenen Skripten und CGI-BIN-Programmen zu OpenSSL, mit denen sich eine Zertifizierungsstelle komfortabler als mit dem „nackten“ OpenSSL aufbauen und betreiben läßt.

[2] `http://www.pca.dfn.de/dfnpca/pki-links.html`
[3] `http://www.bsi.bund.de/aufgaben/projekte/sphinx/herstell.htm`
[4] `http://www.openca.org`
[5] `http://sites.inka.de/ms/python/pyca/`

3.2.2 PGP

Für die Zertifizierung von PGP-Schlüsseln gibt es bislang keine dezidierte Software für Zertifizierungsstellen, da der PGP-Ansatz eine Zertifizierung „von gleich zu gleich“ und keine herausgehobene Instanz wie eine CA vorsieht. Bei PGP-Zertifizierung müssen also auch die Zertifizierungsstellen auf die normale PGP-Anwendungssoftware zurückgreifen – oder mit einer der verfügbaren Software-Bibliotheken PGP-kompatible Software selbst entwickeln.

Frei verfügbare Software

Außerhalb Nordamerikas legal erhältlich und im Sourcecode verfügbar (wenn auch z.T. nicht immer völlig frei von Nutzungsbeschränkungen) sind die folgenden Implementierungen:

- **PGP 2 (“classic”)**, die rein Kommandzeilen-orientierten Versionen von PGP in diversen Varianten, deren Abschluß PGP 2.6.3ia “*i*nternational, Patchlevel *a*” und PGP 2.6.3in „international – *I*ndividual *N*etwork“, eine Weiterentwicklung aus PGP 2.6.3ia für den Einsatz in Zertifizierungsstellen (s. Anhang B.1) darstellen
- **PGP 5.0i ff.**, die PGP-Versionen mit grafischer Bedienoberfläche und Integration in bzw. Plugins für gängige Mailprogramme – den derzeitigen Schlußpunkt dieses Entwicklungszweiges stellt PGP 6.5.1i dar, das auch in einer Kommandozeilen-Version (mit entsprechen geringerer Funktionalität) vorliegt
- **Gnu Privacy Guard (GnuPG, GPG)**, eine unter der *Gnu Public License (GPL)*[6] zugängliche, OpenPGP-konforme Kommandozeilen-Software, die zur Zeit in einem vom Bundesministerium für Wirtschaft und Technologie geförderten Projekt der German Unix User Group (GUUG) um eine grafische Bedienoberfläche und Plugins für verbreitete Mailprogramme ergänzt werden soll

Einzelheiten insbesondere zu den PGP 2.x-Versionen und deren Herkunft sowie zu GnuPG nennt ROESSLER [Roe98]. (Bezugsquellen für die genannten Programme können Anhang B.1 entnommen werden.)

Darüber hinaus existieren mehrere freie Software-Bibliotheken, die PGP-Funktionalität anbieten bzw. das Nachrichtenformat von PGP unterstützen:

- die Library PGPlib[7] von TAGE STABELL-KULOE, Uni Tromsø
- CTClib bzw. CTCjava[8] von IAN MILLER und „Mr. Tines“
- das Java-Krypto-Toolkit Cryptix[9]

[6] `http://www.gnu.org/copyleft/gpl.html`
[7] `ftp://ftp.cert.dfn.de/pub/dfnpca/tools/pgp/utils/PGPlib.tar.gz`
[8] `http://www.bifroest.demon.co.uk/ctc/`
[9] `http://www.cryptix.org`

Kommerzielle PGP-Software

Neben den „freien" Versionen bzw. Implementierungen des PGP-Standards bieten auch einige Firmen kommerzielle PGP-Versionen oder zu PGP kompatible Software an. Zu nennen wäre natürlich an erster Stelle die ehemalige Firma PGP Inc., heute Teil von Network Associates[10] mit ihrem Produkt PGP 6.5 (für Unix, Windows und MacOS) sowie ihrer *PGP Total Network Security*-Produktlinie. Daneben bieten die Ascom Systec AG mit „ascom MAIL / IDEA@exchange" und die Glück & Kanja GmbH mit ihrer „CryptoEx Security Suite" (siehe Anhang B.1) vergleichbar umfassende Lösungen an.

3.3 Zertifizierungsstellen in Deutschland

Das Signaturgesetz (siehe 3.5.1) trat 1997 in Kraft, doch auch zuvor existierten in Deutschland schon Zertifizierungsstellen, die öffentliche Schlüssel beglaubigten. Hier soll eine Auswahl all dieser teilweise kommerziell betriebenen, teilweise nicht gewinnorientiert arbeitenden Einrichtungen kurz vorgestellt werden.

3.3.1 Nichtkommerzielle CAs

Größtenteils schon vor der Verabschiedung des SigG wurden die folgenden „Non-Profit"-Zertifizierungsstellen betrieben. Sie bieten die Zertifizierung von PGP-Schlüsseln, teilweise die Ausstellung von X.509-Zertifikaten und zum Teil auch beides an:

- DFN-PCA[11], PGP + X.509 (siehe 3.4.1.5)
- Zertifizierungshierarchie des Individual Network (IN) e.V.[12] mit Zertifizierungsstellen in einigen der Regionaldomains des IN, PGP (X.509 in Vorbereitung) [Cam98b]
- Krypto-Kampagne und pgpCA der Computerzeitschrift *c't*[13], PGP, auf den großen Computer- und Informationstechnik-Messen in Deutschland (CeBIT, Systems, IFA) am Stand des Heise-Verlages[14], hat bereits über 4000 PGP-Schlüssel zertifiziert [Luc97, Luc98, Luc99a]
- GMD Trustfactory[15], X.509, die ICE-TEL-CA für Deutschland im Rahmen des ICE-TEL-Projektes (siehe 3.4.2.1), betrieben von der Secude GmbH

Darüber hinaus gibt es eine ganze Zahl weiterer, kleinerer nicht-kommerziell arbeitender Zertifizierungsstellen (siehe *The PKI Page*[16] der DFN-PCA, die auch Verweise auf viele kommerzielle und nicht-kommerzielle Zertifizierungsinstanzen in anderen Ländern enthält).

[10] `http://www.nai.com` bzw. `http://www.pgpinternational.com`
[11] `http://www.pca.dfn.de`
[12] `http://www.in-ca.individual.net`
[13] `http://www.heise.de/ct/pgpCA/`
[14] `http://www.heise.de`
[15] `http://www.secude.com/GMD-TrustFactory/`
[16] `http://www.pca.dfn.de/dfnpca/pki-links.html`

3.3.2 Kommerzielle CAs

Bereits seit längerem tätig sind folgende kommerzielle Zertifizierungsdienstleister mit Sitz in Deutschland (ohne Anspruch auf Vollständigkeit):

- TC TrustCenter for Security in Data Networks GmbH[17], Hamburg,
- Competence Center Informatik GmbH[18], Meppen,
- IKS GmbH[19], Jena, und
- Secunet GmbH[20], Essen.

Seit dem 25. Januar 1999 ist die Root-CA nach dem Signaturgesetz (SigG), die sog. „zuständige Behörde“, in Betrieb [Tsp99a, Luc99b]. Die erste von ihr zugelassene Zertifizierungsstelle war das Produktzentrum Telesec der Deutschen Telekom AG. Für 1999 wurden nach Auskunft der „zuständigen Behörde“ noch „zwei bis sieben“ weiteren Genehmigungen für SigG-Zertifizierungsstellen erwartet, offenbar ergaben sich dabei aber Verzögerungen, denn in 1999 wurde keine weitere SigG-Zertifizierungsstelle bestätigt. Im Februar 2000 erhielt nun mit dem Geschäftsfeld SignTrust der Deutschen Post AG, die zweite Zertifizierungsstelle den „Segen“ der SigG-Wurzelinstanz [Pos00].

Insgesamt gibt es über 30 Antragsteller für die Zulassung als SigG-Zertifizierungsstelle; darunter befinden sich Telekommunikationsunternehmen und Krankenhäuser [Tsp99a] ebenso wie die DFN-PCA und die folgenden Unternehmen bzw. Einrichtungen [CZ98k, MoP98, SH98a]:

- D-Trust, Berlin, ein Joint-Venture der Debis IT Security Services GmbH und der Bundesdruckerei GmbH
- TÜV-Informationstechnik (TÜV-IT)
- DATEV Datenverarbeitung und Dienstleistung für den steuerberatenden Beruf eG, Nürnberg
- die Bundesnotarkammer
- der Gesamtverband der Deutschen Versicherungswirtschaft Deutschland
- DE-CODA Gesellschaft zur elektronischen Zertifizierung von Dokumenten mbH, Bonn, eine 100%ige Tochter des Deutschen Industrie- und Handelstages (DIHT)

3.4 Forschungs- und Pilotprojekte

Einige Feldversuche und Pilotprojekte in Deutschland und in Europa haben sichere elektronische Kommunikation zum Gegenstand. Sie betreffen sowohl die Verwaltung auf Bundes- und Länderebene als auch Forschungseinrichtungen und Unternehmen. Wenn auch in der überwiegenden Zahl

[17] `http://www.trustcenter.de`
[18] `http://www.cci.de`
[19] `http://www.iks-jena.de`
[20] `http://www.secunet.de`

dieser Projekte das Zertifikatformat X.509 (siehe 3.1.1) verwendet wird – meist im Zusammenhang mit der in Deutschland entwickelten MailTrusT-Spezifikation –, gibt es doch auch einige öffentliche Stellen, die beispielsweise auf (Open)PGP setzen. Zu diesen zählen u.a. der Städtetag Baden-Württemberg[21] und das Bundesverfassungsgericht, das seine Urteile bzw. Urteilsbegründungen im WWW in PGP-signierter Form nachprüfbar authentisch veröffentlicht[22].

3.4.1 Pilotprojekte in Deutschland

3.4.1.1 Informationsverbund Berlin – Bonn

Infolge des Regierungsumzuges von Bonn nach Berlin kommt der Sicherstellung der Kommunikation und Zusammenarbeit der Parlamentseinrichtungen und Regierungsbehörden innerhalb beider und zwischen beiden Städten eine herausragende Bedeutung zu. Der Informationsverbund Berlin-Bonn (IVBB) liefert die technischen und organisatorischen Voraussetzungen für den Informationsaustausch u.a. zwischen den Ressorts, mit dem Bundestag, dem Bundesrat und dem Bundespräsidialamt. Den Gefährdungen beim elektronischen Dokumentenaustausch im IVBB soll durch den Einsatz von Public-Key-Verfahren entgegengewirkt werden [AE97, S. 146]. Die einzurichtenden Verfahren sollen dabei konform zum Signaturgesetz sein, zum einen im Sinne einer „Vorbildfunktion“ der Bundesverwaltung, zum anderen um den Behörden einen sicheren Austausch digitaler Dokumente mit Dritten zu ermöglichen [AE97, S. 147].

3.4.1.2 SPHINX – Pilotversuch der Bundesverwaltung

Im Bereich der gesamten Bundesverwaltung sollen Verfahren zur Verschlüsselung und digitalen Signatur eingesetzt werden, um beim Dokumentenaustausch in elektronischer Form, z.B. per E-Mail oder über interne Netze, Ende-zu-Ende-Sicherheit zu erreichen.

Im Rahmen des SPHINX-Pilotversuches[23] wurden daher Produkte verschiedener Hersteller nach dem MailTrust-Standard[24] [Bau96] des TeleTrust e.V.[25] zur Realisierung der Ende-zu-Ende-Sicherheit im Bereich der öffentlichen Verwaltung erprobt sowie die für den Einsatz derartiger Sicherheitsmaßnahmen erforderlichen Voraussetzungen geschaffen, u.a. der Aufbau und Betrieb von Zertifizierungsstellen, die die benötigten Schlüssel bereitstellen. Es sollte dabei nachgewiesen werden, daß ein sicherer Dokumentenaustausch auf Basis der verwendeten Standards herstellerübergreifend möglich ist, und es sollten Erfahrungswerte für künftige Beratungs- und Beschaffungsmaßnahmen gewonnen werden über geeignete Produkte, den Aufwand für Installationen und Schulungen, Technikgestaltung und Anwenderakzeptanz sowie über den Aufwand für den Aufbau und Betrieb von Zertifizierungsstellen.[26]

[21] `http://www.redtenbacher.de/signatur/index.htm`
[22] `http://www.bundesverfassungsgericht.de/text/sicherheit.html`
[23] Projekt-Homepage `http://www.darmstadt.gmd.de/SPHINX/`
[24] Die Version 2.0 der MailTrusT-Spezifikation ist seit Mitte März online unter `http://www.secorvo.de/publikat/mttspc20.zip` verfügbar
[25] `http://www.teletrust.de`
[26] `http://www.bsi.bund.de/aufgaben/projekte/sphinx/`

Der Teilnehmerkreis des Pilotversuchs umfaßte circa 200 Mitarbeiter aus dem Bereich der Bundesbehörden mit Sitz in Bonn sowie einige Teilnehmer in den Bundesländern und dem kommunalen Bereich, zwei Teilnehmer aus dem Ausland und circa 20 Mitarbeiter privater Firmen. Dazu gehörten u.a. das Auswärtige Amt, das Bundeskanzleramt, das Bundeskriminalamt, das Bundesministerium der Finanzen, das Bundesministerium des Innern, der Deutsche Bundestag, das Landesamt für Datenverarbeitung und Statistik Potsdam, die Regulierungsbehörde für Telekommunikation und Post und das Sächsische Staatsministerium des Inneren.

Inzwischen läuft eine weitere Phase des Pilotversuchs, an der diesmal rund 600 Teilnehmer aus Bundes-, Länder- und Kommunalverwaltungen, der EU-Kommission sowie aus Firmen beteiligt sind. Ende 2000 soll dann die gesicherte elektronische Kommunikation in den staatlichen Stellen auf breiter Front eingeführt werden (siehe SPHINX-Webseite).

3.4.1.3 Elektronische Post in den obersten Landesbehörden Nordrhein-Westfalens

Das Bundesland Nordrhein-Westfalen startete bereits 1991 ein Pilotprojekt „Einsatz der elektronischen Post (MHS/X.400) in den obersten Landesbehörden des Landes Nordrhein-Westfalen“ und baut seit 1992 ein landesweites Netz für elektronische Post auf. Über dieses Netz können inzwischen von über 50 000 PC-Arbeitsplätzen elektronische Dokumente übertragen werden. Um dabei die Vertraulichkeit datenschutzrechtlich relevanter oder sonst geheimhaltungsbedürftiger Informationen sicherzustellen, setzt die Landesverwaltung ein Produkt zur Verschlüsselung und für elektronische Unterschriften ein. Die Schlüsselverwaltung und Zertifizierung erfolgt dabei zentral über das Landesamt für Datenverarbeitung und Statistik; der Aufbau eines Trustcenters für X.509-Zertifikate ist geplant [Rud97].

3.4.1.4 Digitale Signaturen für die Landesverwaltung von Niedersachsen

Die niedersächsische Landesverwaltung hat im Januar 2000 mit der Deutschen Telekom eine Vereinbarung getroffen, die die Einführung von digitalen Signaturen im Landesdienst ermöglicht.[27] Niedersachsen will nun die elektronischen Unterschriften bei allen Mitarbeitern einführen – seit Jahresbeginn werden bereits digitale Signaturen an Landesbedienstete ausgegeben. Nach Abschluss der Umstellung sollen 12.000 Bedienstete an 700 Standorten mit elektronischen Unterschriften arbeiten.[28]

3.4.1.5 Projekt DFN-PCA

Seit Januar 1996 wird an der Universität Hamburg das Pilotprojekt „Policy Certification Authority“ (PCA) des DFN-Vereins betrieben. Es wurde zwischenzeitlich bis zum 31.12.2000 verlängert.

In diesem Projekt werden die nachstehenden Ziele verfolgt [KL96]:

- Aufbau und Betrieb einer PCA: Die DFN-PCA soll Zertifikate für organisationsweite CAs erstellen, aber auch Netzwerkmanager und Systemadministratoren bei der Einrichtung eige-

[27] `http://www.dtag.de/dtag/presse/artikel/0,1018,x505,00.html`
[28] `http://www.heise.de/bin/nt.print/newsticker/data/jk-28.01.00-003/`

ner CAs unterstützen. Grundlage ist eine hierarchische Struktur von Zertifizierungsstellen innerhalb des DFN, wie sie in PEM [Ken93b] bzw. [Ken93a] vorgesehen ist.

- Betreuung und Unterstützung der Anwender und CA-Administratoren in DFN-Mitgliedseinrichtungen, und zwar bei allgemeinen Fragen zu Zertifizierung, Zertifizierungsinstanzen, kryptographischen Verfahren, digitalen Signaturen etc., bei der Zertifizierung untergeordneter Organisations-CAs durch die DFN-PCA, bei der Zertifizierung von End-Anwendern, in deren Einrichtung es noch keine eigene Zertifizierungsstelle gibt (zu diesem Zwecke betreibt die DFN-PCA bis auf weiteres eine eigene zusätzliche CA unterhalb der DFN-PCA) und bei der Kooperation mit anderen Gruppen.

- Unterstützung der gesicherten Kommunikation zwischen DFN-Mitgliedern (untereinander) und mit anderen, z.B. durch Kooperation mit kommerziell betriebenen Zertifizierungsstellen und durch Integration in entsprechende europäische Infrastrukturen.

- Untersuchungen und Forschungen, vor allem praktische Tests relevanter Softwarepakete für den Einsatz innerhalb einer CA, sowie Analysen von neuen Programmen und Konzepten.

Die DFN-PCA war darüberhinaus die nationale CA für Deutschland im Rahmen des EU-Forschungsprojektes ICE-TEL (siehe 3.4.2.1).

3.4.2 EU-Projekte

Im folgenden sind diejenigen Vorhaben und abgeschlossenen EU-Projekte aufgeführt, die Zertifizierungsinstanzen oder -Infrastrukturen zum Hauptgegenstand haben. Daneben gibt es die in Kapitel 1 bereits erwähnte Möglichkeit, Geboten und Anträgen zu Forschungs- und Entwicklungs-Programmen der EU auf elektronischem Wege einzureichen, nachdem sich der Antragsteller hat zertifizieren lassen. Weitere Forschungs- und Entwicklungsprojekte im Rahmen verschiedener EU-Programme, die ebenfalls CAs oder TTPs involvieren, können der Übersicht `http://www.ispo.cec.be/ecommerce/issues/digisig/digisign4.htm` entnommen werden.

3.4.2.1 ICE-TEL

ICE-TEL[29] sollte die Vertrauenswürdigkeit des Internet für die industrielle universitäre Forschung erhöhen, Nutzern in mehreren europäischen Ländern sollten durch das Projekt Zertifizierungsdienste für öffentliche Schlüssel und entsprechende Werkzeuge für deren Einsatz zur Verfügung gestellt werden. Getestet wurden die entwickelten Verfahren bei der Kommunikation der nationalen *Computer Emergency Response Teams* (CERTs). Ausserdem wurde eine X.509v3-Zertifizierungsinfrastruktur mit CAs in Dänemark (ICE-TEL-Root-CA[30]), Deutschland (siehe 3.4.1.5), Italien, Norwegen, Slowenien, Spanien und England aufgebaut.

[29] `http://www.darmstadt.gmd.de/ice-tel/`
[30] `http://ice-tel.uni-c.dk/ice-ca/`

3.4.2.2 ICE-CAR

Während ICE-TEL den Aufbau einer europäischen Zertifizierungs-Infrastruktur zum Ziel hatte, soll das seit 1. Januar 1999 laufende Anschlußprojekt ICE-CAR[31] Anwendungsdemonstrationen der ICE-TEL Sicherheitstechnologien geben und helfen, die im Rahmen von ICE-TEL eingerichtete Infrastruktur mit einer erweiterten Zielsetzung weiter zu betreiben. Insbesondere soll das Projekt die Komponenten bereitstellen helfen, die für eine sichere Nutzung des Internet in Handel und Verwaltung in Europa benötigt werden. Dazu sollen die vorhandenen (in ICE-TEL entwickelten) Sicherheits-Werkzeuge unter den Gesichtspunkten der Bedienbarkeit und der Interoperabilität angewendet und weiterentwickelt werden.

3.4.2.3 EUROTRUST

Im EuroTrust-Projekt[32] wird ein ähnliches Ziel wie in ICE-CAR verfolgt (Aufbau einer CA-/TTP-Infrastruktur), wobei der Schwerpunkt stärker auf den kommerziellen Aspekten des CA-Betriebs sowie auf *key recovery* bzw. *key escrow* liegt als bei ICE-CAR. Es soll die Einrichtung von TTP- und CA-Diensten in mehreren Staaten der EU untersucht werden, und sie sollen in einen zu entwickelnden europaweiten Rahmen eingebunden werden [GNRS98].

3.4.2.4 Trust Infrastructure for Europe (TIE)

Im Rahmen des 1999 gestarteten EU-Forschungsprojektes TIE soll eine offene, interoperable und sichere Vertrauensinfrastruktur „entwickelt“ werden, indem die technischen, rechtlichen und Geschäftspraktiken etabliert und interoperable Produkte entwickelt werden, die die Voraussetzung für eine solche Infrastruktur darstellen.[33] Das Hauptziel ist dabei die Entwicklung neuer sicherer kommerzieller Dienste rund um die Anwendung sicheres *messaging*. Kosten und Nutzen der Einrichtung öffentlicher, privat betriebener Zertifizierungsstellen sollen im Rahmen des Projektes ebenso ermittelt werden wie die Auswirkungen auf die Öffentlichkeit und die Verbraucher.

3.4.2.5 Sicheres transeuropäisches Verwaltungs-Netz

Alcatel und die Belgische Firma GlobalSign wurden Ende Februar 1999 von der EU-Kommission damit beauftragt, ein mit Public-Key-Verfahren abgesichertes Kommunikationsnetz zwischen Regierungen und Verwaltungen der EU-Mitgliedsstaaten aufzubauen – nach der GlobalSign-Pressemitteilung[34] ein „wichtiger Schritt in Richtung einer größeren Verbreitung digitaler Zertifikate in Wirtschaft und Verwaltung“.

---

[31] `http://ice-car.darmstadt.gmd.de`
[32] `http://www.cordis.lu/infosec/src/study7.htm`
[33] `http://www.ispo.cec.be/ecommerce/issues/digisig/digisign4.htm`
[34] `http://www.globalsign.net/company/press/alcatel.cfm`

3.5 Rechtlicher Rahmen

Für den Betrieb einer Zertifizierungsstelle sind viele gesetzliche Vorschriften relevant. Die wichtigsten, weil am konkretesten auf Zertifizierungsstellen abzielenden, sind das deutsche Signaturgesetz mit der zugehörigen Signaturverordnung und die Anfang 2000 in Kraft getretene EU-Richtlinie zur elektronischen Signatur. Darüber hinaus sind die Datenschutzgesetze der Länder bzw. des Bundes maßgeblich. Wenn digitale Signaturen und/oder Verschlüsselungsverfahren betrieblich eingeführt werden sollen (und nicht nur als Angebot zur *freiwilligen* Nutzung), kommen außerdem noch die entsprechenden arbeitsrechtlichen Regelungen hinzu (Arbeitnehmerdatenschutz, Mitbestimmung etc.)[35]

3.5.1 Gesetz zur digitalen Signatur

Das Gesetz zur digitalen Signatur [Sig97a], auch Signaturgesetz oder kurz SigG, ist seit 1. August 1997 in Kraft. Es zielt darauf ab, Rahmenbedingungen zu schaffen, „unter denen digitale Signaturen als sicher gelten und Fälschungen digitaler Signaturen oder Verfälschungen von signierten Daten zuverlässig festgestellt werden können“ (§ 1 Abs. 1). – Insofern ist es also eher ein „Sicherungsinfrastruktur-Gesetz“ als ein „Signaturgesetz“ und seine Bezeichnung leicht irreführend. – Die Bundesrepublik hat damit als eines der ersten Länder den Versuch unternommen, einen Rahmen für den Einsatz und die Anerkennung eines Äquivalents zur eigenhändigen Unterschrift in der Welt der digitalen Kommunikation zu etablieren.

Der Anwendungsbereich des SigG umfaßt Zertifizierungsstellen, für deren Betrieb es einer Genehmigung der im Gesetz als „zuständige Behörde“ bezeichneten obersten Zertifizierungsinstanz bedarf (§ 4 Abs. 1). Die Aufgabe dieser „Wurzelinstanz“ wird im SigG der Regulierungsbehörde für Telekommunikation und Post zugewiesen (§ 3). Leider mangelt es dem Signaturgesetz in einem wichtigen Punkt an Bestimmtheit bzw. Eindeutigkeit: Eine Zertifizierungsstelle wird in ihm definiert als „eine natürliche oder juristische Person, die die Zuordnung von öffentlichen Signaturschlüsseln zu natürlichen Personen bescheinigt und dafür eine Genehmigung gemäß § 4 besitzt“ (§ 2 Abs. 2). Damit liegt ein Zirkelschluß vor, der Unklarheit darüber bewirkt, ob die Lizensierung freiwillig oder obligatorisch sein soll. § 1 Abs. 2 stellt ausdrücklich die Anwendung anderer Verfahren frei, während § 4 Abs. 1 und die Begründung zum Gesetz eine Lizensierungspflicht für jegliches Ausstellen von Zertifikaten nahelegen [Roß97].

Die Erteilung einer Lizenz zum Betrieb einer Zertifizierungsstelle ist nach dem SigG an drei Voraussetzungen geknüpft:

- Zuverlässigkeit des Antragstellers
- Fachkunde des in der Stelle tätigen Personals
- Erfüllung der technischen und organisatorischen Sicherheitsanforderungen des Gesetzes an die Zertifizierungsstelle (Nachweis durch Vorlage eines Sicherheitskonzeptes und die Bestä-

[35] Auf einige interessante Aspekte im Zusammenhang mit der betrieblichen Nutzung von Public-Key-Verfahren geht GERLING ein [Ger00], so u.a. den Umstand, daß die Gültigkeitsdauer von Schlüsseln oder Zertifikaten, sofern sie veröffentlicht werden, keinen Rückschluß z.B. auf ein „nur“ befristetes Arbeitsverhältnis zulassen darf.

tigung der Umsetzung des Konzeptes durch eine von der zuständigen Behörde anerkannte Prüfstelle)

Personen, die den Anschein erwecken, über eine Lizenz als Zertifizierungsstelle nach dem SigG zu verfügen, ohne daß dies der Fall ist, kann die Tätigkeit der Lizensierung untersagt werden (§ 13 Abs. 1).

Das SigG sieht eine zweistufige Zertifizierungshierarchie vor: Auf oberster Ebene befindet sich die „zuständige Behörde" genannte Wurzelzertifizierungsinstanz, die die eigentlichen, privatwirtschaftlich betriebenen Zertifizierungsstellen (zweite Ebene) zertifiziert. Diese wiederum stellen die Zertifikate für die Anwender aus.

So begrüßenswert das SigG als schnelle Reaktion auf technische Neuerungen und die durch das Gesetz geschaffene Rechtssicherheit auch sein mögen, es bleiben auch einige Kritikpunkte, von denen hier nur die wichtigsten kurz erwähnt werden sollen (ausführlichere Darstellungen finden sich z.B. in [Roß97] und [Rei97c]): Die vorgegebene Zertifizierungsstruktur ist sehr rigide und orientiert sich vor allem an bereits existierenden Trustcentern; es sind keine spezifischen Haftungsregelungen vorgesehen (vgl. 4.19.2); die Sicherheitsanforderungen sind sehr hoch und lassen kaum Raum für von einem Trustcenter abweichende Geschäftsmodelle – kleinere Unternehmen oder Spartenanbieter, die nur eine Teil der Trustcenter-Dienste offerieren wollen, dürften kaum in der Lage sein, die hohen Anforderungen zu erfüllen. Darüber hinaus ist der Gesetzestext an manchen Stellen unpräzise, so bei der Verwendung des Begriffes 'öffentlicher Signaturschlüssel', wenn eigentlich ein (öffentlicher) Schlüssel zum *Prüfen* und nicht zum Erstellen von Signaturen gemeint ist (§ 2 Abs. 3 u.a.), oder in § 3, wo von „Zertifikaten [sic], die zum Signieren von Zertifikaten eingesetzt werden", die Rede ist.

Obschon seit 1997 in Kraft, verläuft die Umsetzung des Signaturgesetzes und der es begleitenden Bestimmungen nur schleppend. Die darin vorgesehene Wurzelinstanz („zuständige Behörde") ist erst seit Herbst 1998 betriebsbereit, und bis Ende 1999 erhielt von ihr erst eine SigG-Zertifizierungsstelle (Telesec) ihre Zulassung; im Februar 2000 dann mit der Deutsche Post SignTrust die zweite (siehe 3.3.2). Die Anlaufphase gestaltete sich dabei durchaus nicht ohne Probleme: So gab es bei den wenigen hundert Zertifizierungen, die die Telesec bislang nach dem SigG durchgeführt hat, aus der Sicht eines kritischen Teilnehmers durchaus einiges anzumerken, wie KELM in [Kel99] schreibt.

1999 stand die Evaluierung des SigG nach den – wenigen bis dahin vorliegenden – Erfahrungen der ersten zwei Jahre an. Der Evaluierungsbericht[36] sieht wenig Änderungsbedarf. Im großen und ganzen habe sich das SigG in der geltenden Form „bewährt". Lediglich aus der zum Zeitpunkt seiner Erstellung schon im Entwurf vorliegenden EU-Richtlinie (siehe 3.5.3) leitet er einen – geringen – Änderungsbedarf in einigen Punkten ab. Seine Grundaussage ist aber „Weiter so!".

[36] *Bericht der Bundesregierung über die Erfahrungen und Entwicklungen bei den neuen Informations- und Kommunikationsdiensten im Zusammenhang mit der Umsetzung des Informations- und Kommunikationsdienste-Gesetzes (IuKDG)* vom 18. Juni 1999, `http://www.iid.de/iukdg/berichte/BERICHTiukdg-neu-2.html`

3.5.2 Verordnung zur digitalen Signatur

In § 16 SigG wird die Bundesregierung ermächtigt, durch Rechtsverordnung Details zur Digitalen Signatur und zu SigG-Zertifizierungsstellen zu regeln, die im Signaturgesetz selbst ausgespart wurden. Dies betrifft alle wesentlichen Einzelheiten der Zertifizierung, der Pflichten einer Zertifizierungsstelle und der Maßnahmen, mit denen ihre Einhaltung kontrolliert werden soll.

Die Bundesregierung hat in der *Verordnung zur digitalen Signatur (Signaturverordnung – SigV)* [Sig97b] die entsprechende Ausgestaltung der o.g. Bereiche geregelt.

In der SigV ist unter anderem festgelegt,

- daß die Identifikation des in der SigV als 'Antragsteller' bezeichneten Zertifikatnehmers bei der erstmaligen Beantragung anhand des Personalausweises oder des Reisepasses „oder auf andere geeignete Weise" vorzunehmen ist (bei Folgeanträgen kann von einer erneuten Identifikation abgesehen werden, wenn der Folgeantrag mit einer digitalen Signatur des Antragstellers versehen ist) (§ 3 Abs. 1)
- daß die Zertifizierungsstelle den Zertifikatnehmer über die Verhaltensweisen und Maßnahmen aufklären muß, mit denen er seinen geheimen Signaturschlüssel vor unbefugtem Zugriff schützen muß bzw. die er bei der Prüfung fremder digitaler Signaturen vornehmen sollte (§ 4 Abs. 1), es sei denn, der Betreffende ist bereits entsprechend unterrichtet worden (§ 4 Abs. 2)
- daß Signaturschlüssel durchaus auch vom Zertifikatnehmer selbst unter eigener Kontrolle (und nicht von der Zertifizierungsstelle) generiert werden können (§ 5 Abs. 1)
- daß SigG-Zertifikate eine maximale Gültigkeitsdauer von fünf Jahren aufweisen dürfen (§ 7)
- daß eine Fotokopie des Personalausweises oder Reisepasses des Zertifikatnehmers anzufertigen (§ 13 Abs. 1) und zusammen mit seinem Zertifizierungsantrag sowie allen anderen zugehörigen Unterlagen 35 Jahre lang aufzubewahren ist (§ 13 Abs. 2).

Bemerkenswert ist, daß darüber hinaus nicht nur dem Zertifikatinhaber oder seinem Vertretungsbevollmächtigten, sondern auch der *SigG-Wurzelinstanz* von jeder Zertifizierungsstelle eine Möglichkeit geboten werden muß, das betreffende (bzw. im Falle der „zuständigen Behörde" wohl *jedes*) Zertifikat auf telefonischem Wege nach geeigneter Authentifizierung des Anrufers unverzüglich sperren zu lassen (§ 9 Abs. 1 SigV). Hierdurch wird eine Möglichkeit geschaffen, mit der der Staat letztlich jede SigG-konforme digitale Signatur einzelner Personen ab einem bestimmten Zeitpunkt verhindern könnte – eine sehr weitgehende Eingriffsbefugnis, die ggf. für den Betroffenen quasi der technisch durchgesetzten Beschränkung gleichkäme, keinerlei Verträge mehr unterzeichnen und sich nicht mehr ausweisen zu können. Der Betroffene wäre geradezu seiner digitalen „Identität" beraubt. Diese Durchgriffsbefugnis einer übergeordneten Zertifizierungsinstanz auf die Nutzerzertifikate einer nachgeordneten CA ist außergewöhnlich. Dieser Teil von § 9 Abs. 1 wird interessanterweise auch in der Begründung der Bundesregierung zur SigV [BRD97] mit keinem Wort kommentiert oder auch nur erwähnt. Dort heißt es zu § 9 Abs. 1 lediglich:

> „Die Regelung dient dem Schutz der Signaturschlüssel-Inhaber sowie der dritten Personen, deren Angaben zur Vertretungsmacht in ein Zertifikat aufgenommen wurden. Durch die verlangte Bekanntgabe einer Rufnummer (Telefon) [der Zertifizierungsstelle] soll eine unverzügliche

> Sperrung [eines Zertifikats] ermöglicht werden, da eine Telefonverbindung praktisch jederzeit hergestellt werden kann. Die Bekanntgabe weiterer Telekommunikationsanschlüsse (z.B. Fax) bleibt unbenommen. Als Authentisierungsverfahren kommt z.B. ein Paßwortverfahren in Betracht.“

§ 9 Abs. 1 SigV schreibt letztlich eine Schnittstelle vor, mit der die Bestimmungen von § 13 Abs. 5 SigG besonders schnell durchgesetzt werden können. Dort ist zwar vorgesehen, daß Zertifikate einer Zertifizierungsstelle ihre Gültigkeit behalten, selbst wenn die Wurzelinstanz die Genehmigung für den Betrieb dieser Stelle widerruft; die „zuständige Behörde“ kann aber eine Sperrung von Zertifikaten anordnen, „wenn Tatsachen die Annahme rechtfertigen, daß Zertifikate gefälscht oder nicht hinreichend fälschungssicher sind oder ... zur Anwendung der Signaturschlüssel eingesetzte technische Komponenten Sicherheitsmängel aufweisen, die eine unbemerkte Fälschung digitaler Signaturen oder eine unbemerkte Verfälschung signierter Daten zulassen.“

Auch die Signaturverordnung bleibt noch bewußt allgemein und unspezifisch, was geeignete technische Sicherungsverfahren und Schutzmaßnahmen angeht; derartige Festlegungen wurden aus der SigV in eine mehrteilige SigG-Interoperabilitätsspezifikation („Schnittstellenspezifikation zur Entwicklung interoperabler Verfahren und Komponenten nach SigG/SigV“, SigI), die zur Zeit noch entwickelt wird, und zwei Maßnahmenkataloge für technische Komponenten und für Zertifizierungsstellen [RTP98] ausgelagert (§ 12 Abs. 2 und § 16 Abs. 6), die von der Regulierungsbehörde geführt und im Bundesanzeiger veröffentlicht werden. (Siehe auch Abschnitt 4.19.4 in Kapitel 4 zu Details der Anforderungen.)

3.5.3 EU-Richtlinie zu Rahmenbedingungen elektronischer Signaturen

Die EU hat die Gefahr divergierender nationaler Regelungen der digitalen Signatur sowie unterschiedlicher Maßnahmen auf dem Gebiet der Kryptographie und drohender daraus resultierender Hemmnisse für den freien Waren- und Dienstleistungsverkehr frühzeitig erkannt und bereits 1997 die Mitteilung der EU-Kommission zu „Sicherheit und Vertrauen in elektronische Kommunikation – ein europäischer Rahmen für digitale Signaturen und Verschlüsselung“ [EUK97] vorgelegt. Zuvor waren bereits vorbereitende Gutachten und Studien zu diesem Gebiet eingeholt worden. In dieser Mitteilung heißt es – im Kontrast zum deutschen Signaturgesetz:

> „Da die Lizenzierungspflicht für CAs nicht der einzige Weg ist zu einer Übereinstimmung zwischen den Aktivitäten der CA einerseits und den Vorstellungen der Öffentlichkeit, wie Vertrauen in digitale Signaturen gefördert werden kann andererseits, müßte ein Regulierungsrahmen auf EU-Ebene die Koexistenz sowohl lizenzierter als auch unlizenzierter CAs vorsehen.“ [EUK97]

Mit dem ersten Entwurf einer EU-Richtlinie über gemeinsame Rahmenbedingungen für elektronische Signaturen [EUK98b] vom 13. Mai 1998 wurden die Ziele der Mitteilung von 1997 dann konkretisiert. Im Unterschied zum deutschen Signaturgesetz sah der erste Richtlinienentwurf die Durchsetzung konkrete Haftungsanforderungen vor, die als marktwirtschaftliches Steuerungsinstrument ein hohes Sicherheitsniveau sicherstellen sollten. Der Entwurf blieb bewußt funktional gehalten statt technisch, um eine Offenheit für zukünftige Verfahren zu gewährleisten. In einigen

Punkten hätten, wäre dieser Entwurf verabschiedet worden, die Bestimmungen des SigG angepaßt werden müssen, und zwar sowohl, was die bereits erwähnten Haftungsregelungen angeht, als auch den im Richtlinienentwurf verlangten freien Marktzugang von Zertifizierungsdiensten. Demzufolge sind nationale Akkreditierungsverfahren (wie in der Bundesrepublik mit dem SigG) auf *freiwilliger* Basis zwar zulässig, Zulassungsverfahren und Anforderungen müssen jedoch transparent und nichtdiskriminierend sein – ein Punkt, zu dem die im SigG vorgesehene starre, zweistufige Zertifizierungshierarchie wohl nicht konform wäre [FG98].

Am 25. Januar 1999 wurde ein zweiter, überarbeiteter Entwurf für die EU-Signatur-Richtlinie vorgelegt [EUK99]. Der Rat blieb darin bei seiner Auffassung, daß den Anbietern von Zertifizierungsdiensten ein freier Marktzugang gewährt werden müsse und sie ihre Dienste ohne vorherige Zulassung anbieten können sollten; er schrieb auch vor, daß die rechtliche Anerkennung von digitalen Signaturen anhand objektiver Kriterien erfolgen und nicht von der Lizenzierung eines Dienstanbieters abhängig gemacht werden sollte. Der zweite Richtlinienentwurf unterschied sich vom ersten allerdings dahingehend, daß darin – offensichtlich auf Betreiben Deutschlands – neben dem Konstrukt der *electronic signature* auch eine *advanced electronic signature* definiert wurde (Art. 2 Abs. 1bis), die rechtlich der eigenhändigen Unterschrift gleichgestellt und als Beweismittel in Gerichtsverfahren zugelassen werden sollte (Art. 5 Abs. 1). Die einfachen *electronic signatures* sollten aber andererseits nicht alleine deswegen ihre Beweiskraft verlieren, weil es sich um Beweismittel in elektronischer Form handelt, sie nicht auf einem Zertifikat eines lizenzierten Anbieters beruhen oder sie nicht mit einem sicheren Unterschriften-Gerät (*secure signature creation device*) erzeugt wurden (Art. 5 Abs. 2).

Mit einigen Korrekturen des Europäischen Parlamentes wurde der zweite Richtlinien-Entwurf schließlich zum „Gemeinsamen Standpunkt“ von EU-Parlament und -Rat, der schließlich am 30. Novemver 1999 verabschiedet wurde [EU99]. Mit ihrer Veröffentlichung im Amtsblatt der Europäischen Gemeinschaften trat die EU-Richtlinie dann am 19. Januar 2000 in Kraft.

Den Mitgliedsstaaten der EU bleibt nun eine Frist von 18 Monaten, die Richtlinie in nationales Recht umzusetzen. In Deutschland sollen die erforderlichen Anpassungen des SigG im Rahmen der ohnehin anstehenden Überarbeitung dieses Gesetzes erfolgen; ein erster Entwurf eines Signaturgesetz-Änderungsgesetzes[37] – SigGÄndG – wird zur Zeit diskutiert.

3.5.4 Datenschutz

> „Da CAs in der Lage sein müssen, Schlüsselinhaber zu identifizieren und deshalb Informationen über diese sammeln müssen, sind sie den in der EU-Datenschutzrichtlinie festgelegten Bestimmungen der Gemeinschaft über Weiterverarbeitung und Sicherheit von Daten sowie Übertragung von Daten in Drittstaaten unterworfen. Zum Beispiel dürfen CAs Daten nur dann sammeln und weiterverarbeiten, wenn die einzelne Person zugestimmt hat oder eine gesetzliche Ermächtigung hierfür besteht.“ [EUK97]

Diese Anforderung, die sich auch im o.g. zweiten Entwurf einer EU-Signatur-Richtlinie wiederfindet (Art. 8 Abs. 2), deckt sich mit den Bestimmungen, die das Signaturgesetz bzw. die deutschen Datenschutzgesetze auf Bundes- bzw. Landesebene vorsehen.

[37] `ftp://ftp.pca.dfn.de/pub/dfnpca/sigg/SigGAendG.pdf`

Das Signaturgesetz enthält ausdrückliche Regelungen zum Datenschutz bei der Zertifizierung, insofern hat es als die spezifischere gesetzliche Regelung Vorrang vor den Bestimmungen des Bundes- oder der Landesdatenschutzgesetze. In § 12 Abs. 1 SigG heißt es:

> „Die Zertifizierungsstelle darf personenbezogene Daten nur unmittelbar beim Betroffenen selbst und nur insoweit erheben, als dies für Zwecke eines Zertifikates erforderlich ist. Eine Datenerhebung bei Dritten ist nur mit Einwilligung des Betroffenen zulässig. Für andere als die in Satz 1 genannten Zwecke dürfen die Daten nur verwendet werden, wenn dieses Gesetz oder eine andere Rechtsvorschrift es erlaubt oder der Betroffene eingewilligt hat.“

Es wäre zwar noch zu diskutieren, ob diese Bestimmungen nicht lediglich für solche Zertifizierungsstellen gelten, die nach § 4 SigG lizenziert sind. Letztlich würde das aber kaum etwas an den Voraussetzungen für eine zulässige Verarbeitung personenbezogener Daten ändern, denn privatwirtschaftlich organisierte Zertifizierungsstellen zählen zum nicht-öffentlichen Bereich und fallen somit in den Geltungsbereich des Bundesdatenschutzgesetzes (BDSG) [BDS90], und für Zertifizierungsstellen in einer Körperschaft des öffentlichen (Landes-)Rechts wie eine Universität gilt – soweit nicht bereichsspezifische Normen existieren (z.B. das Hochschulgesetz des jeweiligen Bundeslandes oder die Satzungen der Hochschulen) – das jeweilige Landes-Datenschutzgesetz.[38] Sowohl BDSG als auch Landesdatenschutzgesetze lassen ebenfalls die Erhebung und Verarbeitung personenbezogener Daten nur entweder auf der Grundlage des jeweiligen Datenschutzgesetzes bzw. einer besonderen Rechtsvorschrift oder aufgrund einer expliziten Einwilligung des Betroffenen zu (siehe z.B. § 4 Abs. 1 BDSG).

Da es die Bundesrepublik Deutschland versäumt hat, die EU-Datenschutzrichtlinie [EUP95] fristgemäß in nationales Recht umzusetzen, gilt die Richtlinie nun unmittelbar – im Falle von Rechtsstreitigkeiten sind die bestehenden Datenschutzgesetze daher in ihrem Sinne auszulegen [DSB98].

Immerhin liegt mittlerweile ein Entwurf einer neuen Fassung des BDSG mit Begründung vor,[39] und die Bundesregierung hofft, diesen noch im Laufe des Jahres 2000 zu verabschieden.

[38] Es gibt aber auch Ausnahmen: So gilt beispielsweise für Mitarbeiter der Berliner Universitäten und ihre Daten gemäß § 34 Abs. 2 Berliner Datenschutzgesetz das BDSG und nicht das Berliner Datenschutzgesetz.

[39] `http://www.dud.de/dud/files/bdsg0799.zip`

Kapitel 4

Konzept für eine Zertifizierungsstelle (*“plan”*)

Nachdem die theoretischen Grundlagen für Public-Key-Verfahren und -Zertifizierung bereits kurz umrissen wurden, soll nun in diesem Abschnitt Gedanken zur Einrichtung einer Zertifizierungsstelle in einer DFN-Mitgliedseinrichtung dargelegt werden. Bei der praktischen Umsetzung sollten diese theoretischen Hinweise u.a. durch Ortstermine in der Einrichtung, die die Zertifizierungsstelle später beherbergen soll, ergänzt werden, damit die baulichen und sonstigen Gegebenheiten vor Ort in Augenschein genommen werden und in die Planung mit einfließen können.

Die nachstehenden Überlegungen sind so allgemein bzw. so grundsätzlicher Natur, daß sie fast immer auf eine konkrete CA-Planung anwendbar sein oder zumindest wertvolle Anregungen geben sollten.

Im ersten Abschnitt dieses Kapitels werden die Ziele beschrieben, die diesem (exemplarischen) Konzept zugrunde liegen. Danach wird erläutert, welche Vorteile die Realisierung einer Zertifizierungsstelle mit eigenen Mitteln für die jeweilige Einrichtung bietet. Es folgt ein Überblick über die wichtigsten Punkte der dafür vorgeschlagenen Lösung, deren Einzelheiten dann in den nachfolgenden Abschnitten dargestellt und begründet werden.

4.1 Entwurfsziel

Die Universität zu ..., im folgenden UNI genannt, verfügt bislang nicht über eine eigene Zertifizierungsstelle für öffentliche Schlüssel. Es hat bereits erste – wenn auch noch vereinzelte – Nachfragen nach Schlüsselzertifizierungen an das Rechenzentrum der UNI und an die Benutzerbetreuung gegeben. Außerdem betreibt das ebenfalls in ... ansässige Forschungsinstitut ... bereits eine eigene CA (wenn auch bislang nur mit mäßiger Resonanz). Es ist allerdings absehbar, daß die Zahl der Nachfragen nach Schlüsselzertifizierung und Beratung rund um Verschlüsselung und digitale Signaturen aufgrund der Umsetzung des Signaturgesetzes und der wachsenden Zahl entsprechender Angebote am Markt in nächster Zeit eher zu- als abnehmen wird. Es ist ferner nicht auszuschließen, daß ein einzelnes Ereignis (das könnte beispielsweise das Angebot einer großen Bank sein, die Kontoführung bei ihr jetzt auch per verschlüsselter und signierter E-Mail abzuwickeln) eine plötzliche große

Nachfrage nach Zertifizierungsdiensten auslöst, der sich das Rechenzentrum dann nicht entziehen könnte. Auch für diesen Fall will die UNI gerüstet sein.

Das vorliegende Konzept soll einen Vorschlag oder eine Handreichung für eine mögliche Vorgehensweise bei der Realisierung einer eigenen CA an der UNI sein. Dabei soll die UNI beispielhaft für jede größere Hochschule oder sonstige Einrichtung stehen, die Mitglied im DFN-Verein ist.

In diesem Abschnitt wird dazu ein Konzept entwickelt werden, auf dessen Grundlage eine Zertifizierungsstelle für öffentliche Schlüssel an der UNI betrieben werden könnte. Das Konzept soll die Vorgehensweise beim Aufbau und beim späteren Betrieb der CA möglichst verständlich beschreiben, so daß anhand dieses Konzeptes auch Uni-Angehörige, die zwar eine gewisse Erfahrung im Umgang mit Computern haben, die jedoch keine Zertifizierungsspezialisten oder Kryptographie-Experten sind, die vorgeschlagene Arbeits- und Funktionsweise der Zertifizierungsstelle nachvollziehen und die zu ihrer Einrichtung erforderlichen Schritte unternehmen können.

Da die Hochschulen Mitglieder des DFN-Vereins sind und dieser mit dem Projekt einer DFN-"Policy Certification Authority" (PCA) eine Wurzel-Zertifizierungsstelle innerhalb des Deutschen Forschungsnetzes etabliert hat, soll das Konzept es ermöglichen, die Zertifizierungsstelle oder einen Teil von ihr als *certification authority* (CA) der UNI (im folgenden UNI-CA genannt) im Sinne der Zertifizierungsrichtlinien der DFN-PCA zu betreiben.

Im Rahmen des hier dargelegten Konzeptes wird u.a. eine geeignete Zertifizierungsrichtlinie für die UNI-CA entworfen, die den Rahmen für den späteren Praxisbetrieb setzt. Dabei sollen auch die typischerweise an einer Universität gegebenen Rahmenbedingungen berücksichtigt werden, so z.B. die Fluktuation unter den Studierenden, Zeitverträge bei vielen der wissenschaftlichen Universitätsmitarbeitern, ausländische Studierende und Gastwissenschaftler (die u.U. andere Personaldokumente als einen Personalausweis besitzen!), beschränkte bis geringe Sach- und Personalmittel speziell für eine solche Zertifizierungsstelle etc. Die UNI-CA soll Zertifizierungsdienste sowohl für die Sicherung der E-Mail-Kommunikation von Universitätsangehörigen als auch für den gesicherten Zugriff auf WWW-Ressourcen der UNI anbieten.

Eine Betrachtung der rechtlichen Rahmenbedingungen für den Betrieb einer Zertifizierungsstelle nebst möglicher Haftungsfragen erfolgt in diesem Handbuch ebenfalls. Allerdings ist unter seinen Autoren kein Jurist, insofern sollte, bevor/wenn es zur Umsetzung kommt, auch die Rechtsabteilung der jeweiligen Einrichtung konsultiert werden. In diesem Zusammenhang ist auch von Interesse, ob – und wenn ja unter welchen Voraussetzungen – der Betrieb einer Signaturgesetz-konformen Zertifizierungsstelle an der UNI denkbar wäre. Es sollen daher entweder die dafür zusätzlich erforderlichen Vorkehrungen beschrieben oder aber es soll begründet werden, warum ein Signaturgesetz-konformer Betrieb einer zukünftigen UNI-CA bei der jetzigen Fassung des Signaturgesetzes, der entsprechenden Verordnung und der zugehörigen Ausführungsbestimmungen nicht möglich scheint.

Wichtig ist, folgendes nicht aus den Augen zu verlieren: Das Ziel ist keine *Forschungs*-CA, sondern der *Regelbetrieb* einer Zertifizierungsstelle als Service für die Angehörigen der UNI (und eventuell auch für Dritte).

4.2 *Outsourcen* oder *Do-it-yourself*?

Viele Unternehmen würden in dieser Situation einen entsprechenden Auftrag an Dienstleister vergeben oder zumindest versuchen abzuschätzen, ob es nicht womöglich günstiger ist, die Dienstleistungen der UNI-CA von einem externen Anbieter einzukaufen, oder doch insgesamt die Vorteile überwiegen, wenn man eine solche CA mit eigenen Mitteln realisiert.

Es gibt inzwischen erste Praxisbeispiele, welche Kosten einzelnen Unternehmen bei der Einführung einer Public-Key-Infrastruktur entstanden sind: Über einen Zeitraum von fünf Jahren kommen bei 5 000 Nutzern gut 100 Dollar pro User [Mac98] zusammen. Andere Studien nennen zwischen 400 000 und 1,6 Millionen DM [NC98] für die gleiche Anzahl von Zertifizierungen und den gleichen Zeitraum.

An diesen Summen wird deutlich, daß es für die UNI kaum zu finanzieren wäre, wenn sie diese Dienstleistung out-sourcen würde und tatsächlich ein großer Teil der Studenten oder auch nur der Rechenzentrumsnutzer Gebrauch von einer Zertifizierungsmöglichkeit machen würde.[1]

Eine *Inhouse*-Lösung hätte einen weiteren, nicht unerheblichen Vorteil: Durch die Verknüpfung von Rechenzentrum und Zertifizierungsstelle könnte Know-how erworben und könnten Synergieeffekte genutzt werden für alle anderen (Netzwerk-)Bereiche, in die jetzt oder in Zukunft ebenfalls Public-Key-Verfahren Einzug halten werden (siehe dazu auch 4.20.5).

4.3 Überblick über das Konzept

Die UNI-CA soll grundsätzlich nach den Richtlinien der DFN-PCA arbeiten und eine Zertifizierung durch die DFN-PCA anstreben. Das Konzept ist so gewählt, daß diese Prämisse erfüllt wird. Die UNI-CA kann bis auf weiteres nicht als Zertifizierungsstelle nach dem deutschen Signaturgesetz betrieben werden, weil dessen Anforderungen an organisatorische, technische, personelle und bauliche Maßnahmen so hoch sind, daß sie die Möglichkeiten der UNI bzw. ihres Rechenzentrums übersteigen – zumal ja auch noch nicht abzusehen ist, ob ein entsprechender Bedarf für Signaturgesetz-konforme Zertifizierung vorhanden wäre, der einen solchen Aufwand rechtfertigen würde.

Den Schwerpunkt der Arbeit der UNI-CA (und deswegen auch der Betrachtungen in diesem Teil des Handbuches) stellt die sichere Kommunikation mittels PGP dar. Dieses Programm bzw. Austauschformat besitzt eine hohe Verbreitung – weltweit nutzen zwischen sechs und 20 Millionen Menschen PGP [Moe99, SW97] – und stellt zur Zeit den De-facto-Standard für vertrauliche und/oder authentische Kommunikation per E-Mail und im Usenet dar. (DFN-CERT: *„Die Signatur von DFN-CERT-Mitteilungen erfolgt bis auf weiteres ausschließlich mit PGP, da dies das zur Zeit am verbreitesten* [sic] *Verfahren ist.“* [CER94])

Hinzu kommt, daß die X.509-Zertifizierung im DFN sich derzeit in einer Umbruchsphase befindet. Es wurden schon seit längerem X.509-Zertifikate von der DFN-PCA bzw. deren User-CA ausgestellt; diese zielten jedoch auf den Einsatz in PEM ab, einem Standard, der im Vergleich zu PGP keine große Verbreitung im Internet erlangt hat. Mittlerweile werden unter einer dritten, vorläufigen

[1] Zu weiteren Argumenten für oder wider einen Auftrag an ein externes Unternehmen siehe MOSES [Mos97].

DFN-Policy aber auch X.509v3-Zertifikate für den Einsatz mit HTTPS-Servern und in WWW-Browsern von der DFN-PCA ausgestellt, was eine Betrachtung zur Zeit etwas schwierig macht.

Mit nur geringen Anpassungen bzw. geringem Mehraufwand sollte aber auch die X.509-Zertifizierung von der UNI-CA angeboten werden können, wenn Bedarf dafür besteht. Die Vorschläge dieses Konzeptes sind so angelegt, daß die UNI-CA damit alle technisch-organisatorischen Voraussetzungen auch für eine X.509-Zertifizierung durch die DFN-PCA erfüllt und sie sich eventuell auch schon „X.509-zertifizieren“ lassen könnte. Teil II dieses Handbuches befaßt sich allgemein (nicht speziell auf die UNI-CA bezogen) mit der X.509-Zertifizierung unter Verwendung der Software OpenSSL; an diesen Ausführungen könnte sich bei Bedarf auch die UNI-CA orientieren.

Die DFN-PCA hat zwei separate Gruppen von Zertifizierungsrichtlinien, nach denen sie arbeitet, eine Low- und eine Medium-Level Policy, die sich in den unterstützten Formaten und in den Sicherheitsanforderungen unterscheiden. Daneben wird seit Ende 1998 auch noch eine „World Wide Web-Policy“ für die Zertifizierung von Web-Browsern und -Servern angewendet. Eine PGP-Zertifizierung ist nur in der DFN-Low-Level-Policy vorgesehen. Diese ist gleichzeitig aber an vielen Stellen auslegungsfähig formuliert, enthält viele „Kann“-Bestimmungen und Empfehlungen anstelle konkreter Anforderungen oder Vorgaben und ermöglicht es, mit wenig Sicherheitsaufwand, aber eben auch mit einem begrenzten Sicherheitsniveau eine Zertifizierungsstelle zu betreiben. Die DFN-Medium-Level-Policy hingegen stellt höhere Sicherheitsanforderungen an Betreiber und Zertifizierungswillige, sieht aber keine PGP-Zertifizierung vor. Die technische Ausstattung und die Arbeitsabläufe der UNI-CA sollten daher grundsätzlich (auch hinsichtlich der PGP-Zertifizierung) so ausgelegt werden, daß eine Zertifizierung nach der DFN-Medium-Level-Policy möglich ist.

Die UNI-CA soll nach diesem Konzept die Zertifizierung nach zwei unterschiedlich strengen Policies anbieten:

- einer **„Low-Level Security“ Policy**, nach der ausschließlich PGP-zertifiziert wird und
- einer **„Medium-Level Security“ Policy**, nach der sowohl PGP- als auch X.509-Zertifizierungen vorgenommen werden (können)

Als Zertifizierungsrechner wird ein unvernetzter, tragbarer Computer vorgeschlagen, der mit zwei austauschbaren Festplatten ausgestattet wird, von denen eine die Low-Level- und eine die Medium-Level-CA „beherbergt“ (jeweils Software *und* Zertifizierungsdaten). Dieser Rechner sollte in einem abschließbaren Schrank, über dessen Schlüssel nur die autorisierten CA-Mitarbeiter verfügen, im Serverraum des Rechenzentrums aufbewahrt werden, wenn er nicht benutzt wird. Nur zur Zertifizierung darf der Rechner nebst der entsprechenden Festplatte für die CA entnommen werden; er ist nach Gebrauch umgehend wieder in diesem Schrank einzuschließen.

4.4 Begründung für den vorgeschlagenen Ansatz

4.4.1 Warum zwei Policies?

Eine Universitäts-Zertifizierungsstelle wird sich in besonderem Maße dem Spagat zwischen zwei relativ gegensätzlichen Anforderungen stellen müssen: Sie muß auf der einen Seite Zertifizierungsdienste mit einem angemessen hohen Sicherheitsniveau anbieten, sie muß auf der anderen Seite aber auch Aufklärungs- und Schulungsarbeit leisten und so erst für eine breite(re) Nutzerbasis sorgen. Um neue Nutzer für den Einsatz von Verschlüsselung bzw. digitalen Unterschriften zu gewinnen, dürfen die Zugangs- oder Einstiegshürden nicht zu hoch sein, sonst werden Interessierte abgeschreckt und aufgrund des hohen Aufwandes von einer Zertifizierung abgehalten. Sind andererseits die Zertifizierungsrichtlinien zu lax, werden erfahrenere Nutzer dies erkennen, kritisieren und sich mit der Nutzung der CA-Dienste zurückhalten.

Die vorgeschlagenen zwei CAs mit unterschiedlich hohen Sicherheitsanforderungen stellen eine Lösung für dieses Problem dar. Die Hürden für die Erst-Zertifizierung sind bei der Low-Level-CA bewußt niedrig gehalten, und wessen Interesse einmal geweckt ist und wer für Sicherheitsfragen sensibilisiert wurde, der wird auch einen höheren Aufwand bei der Zertifizierung in Kauf nehmen, wenn dadurch ein höheres Zertifizierungs- bzw. Sicherheitsniveau erreicht wird.

4.4.2 Warum ein unvernetzter Zertifizierungsrechner?

Ein über Netzwerkverbindungen erreichbarer Rechner eröffnet einem Angreifer derartig viele Möglichkeiten, aus der Anonymität des Internet heraus einen Angriffsversuch zu unternehmen, daß ein wirksamer Schutz nur mit sehr hohem administrativem Aufwand erreicht werden kann (wenn überhaupt): „Bei vernetzten Rechnern ist die Abstrahlsicherheit vernachlässigbar, da [dort] andere Angriffe sehr viel einfacher sind.“ [Wol98] Das IT-Grundschutzhandbuch des BSI stellt dazu fest:

> „Sind die meisten IT-Anwendungen auf einem IT-System nur mittelschutzbedürftig und sind nur eine oder wenige hochschutzbedürftig, so ist unter Kostengesichtspunkten zu prüfen, diese hochschutzbedürftigen auf ein isoliertes IT-System auszulagern.“ [BSI98]

Insofern ist es unter dem Gesichtspunkt der sparsamen Ressourcenverwendung – hier: Arbeitszeit – sinnvoll, einen unvernetzten Rechner einzusetzen. Auch die DFN-PCA legt aus diesen Gründen in jeder ihrer drei Policies den CAs jeweils nahe, nach Möglichkeit einen *unvernetzten* Rechner als Zertifizierungsrechner einzusetzen.

4.4.3 Warum ein mobiler Zertifizierungsrechner?

Bei der Frage des Zertifizierungsrechners stellt ein mobiler Computer einen günstigen Kompromiß zwischen den verschiedenen Anforderungen dar, die an ein solches Gerät zu richten sind bzw. die durch den Rahmen ‘Universitätsrechenzentrum’ vorgegeben sind: Der Zugang zu bzw. Zugriff auf diesen Rechner darf im Interesse von Verfügbarkeit und Integrität dieses Gerätes und der darauf befindlichen Software nur kontrolliert und nur dem berechtigten (möglichst kleinen) Personenkreis möglich sein. Das bedeutet, daß für einen stationären Zertifizierungsrechner ein eigener Raum erforderlich wäre, zu dem nur die autorisierten CA-Mitarbeiter Zugang haben. Das wiederum hieße, daß zum einen ein Raum dafür bereitgestellt werden müßte, der für anderweitige Nutzung ausfiele, zum anderen würde das aber auch bedeuten, daß die CA-Mitarbeiter für die Zertifizierung eben diesen Raum aufsuchen und dort arbeiten müssen. Es müßte, da wohl keiner der Rechenzentrumsmitarbeiter seinen Arbeitsraum aufgeben würde, ein bislang ungenutzter Raum sein – der dann vermutlich unattraktiv z.B. im Tiefgeschoß läge, schlecht beleuchtet wäre o.ä. (ansonsten würde er sicher nicht leerstehen). Dies wiederum trüge nicht zur Motivation der CA-Administratoren bei, wenn sie sich zur Arbeit am Zertifizierungsrechner in einen solchen Raum begeben müßten. Während der Aufbauphase und zu Beginn des Regelbetriebes wäre wegen der voraussichtlich zunächst geringen Nachfrage vermutlich kein Mitarbeiter nur mit Zertifizierungsaufgaben betraut, sondern diese Arbeit müßte von den betroffenen Mitarbeitern neben ihren übrigen Aufgaben wahrgenommen werden.

Ein solcher dezidierter CA-Rechnerraum müßte – nach Möglichkeit – auch durch eine Einbruchs- und Feuermelde-Anlage usw. geschützt werden, was weiteren zusätzlichen Aufwand für entsprechende Installationen bedeuten würde. Demgegenüber hätte ein tragbarer, unvernetzter Rechner den Vorteil, daß man ihn in einem (eventuell schon vorhandenen) Tresor, Schutzschrank oder abschließbaren Schrank aufbewahren könnte, während er nicht benutzt wird. Dieser Schrank könnte im Serverraum des Rechenzentrums untergebracht werden, das in der Regel bereits über Feuer- und Einbruchsmelder verfügt, oder er befindet sich ohnehin schon dort.

Neben dem isolierten Zertifizierungsrechner wird bei vielen Tätigkeiten im Rahmen der Zertifizierung zusätzlich ein zweiter vernetzter Rechner benötigt, beispielsweise um Zertifikate per E-Mail an die Schlüsselinhaber zu verschicken, Zertifikate über einen Verzeichnisdienst zugänglich zu machen oder auch die zu zertifizierenden Schlüssel von einem Keyserver abzurufen. Ein mobiler Zertifizierungsrechner kann am Arbeitsplatz des betreffenden CA-Mitarbeiters benutzt werden, während anderenfalls in dem dezidierten CA-Rechnerraum auch noch ein zweiter, *vernetzter* Rechner untergebracht werden müßte. (Alternativ müßte ein aufwendiger Datenträgeraustausch zwischen beiden Rechnern praktiziert werden.)

Anschluß- und Verlängerungskabel erleichtern das Abhören [Wol98]. Insofern bietet ein Laptop zusätzlich den (kleinen) Vorteil, daß keine externe Verkabelung anfällt und es Lauschern nicht zusätzlich einfacher gemacht wird.

Ein Vorteil eines Laptops als Zertifizierungsrechner: Tragbare Computer sind (zumindest im Akku-Betrieb) nicht mit dem Stromnetz im Gebäude verbunden; dadurch entfällt ein weiterer möglicher Weg, auf dem ansonsten Signale bis über den Sicherungskasten oder sogar den Hausanschluß hinaus abgehört werden können [Wol98]. – Das LC-Display eines Laptops bietet hingegen nach WOLFF *keinen* oder keinen nennenswerten Vorteil gegenüber herkömmlichen Röhrenmonitoren, was die Abstrahlsicherheit angeht, denn es ist meist die *Grafikkarte*, die die stärkste Abstrahlung liefert, und nicht der Bildschirm.

Ein tragbarer Computer bietet als Zertifizierungsrechner noch einen weiteren Vorteil: Er ermöglicht es, Zertifizierungen *vor Ort* vorzunehmen, beispielsweise an anderen UNI-Standorten, von denen aus das UNI-Rechenzentrum nur schwer zu erreichen ist, oder direkt im Anschluß an Vorträge oder Informationsveranstaltungen über Verschlüsselung bzw. digitale Signaturen.

4.5 Low-Level CA

Die Low-Level UNI-CA nutzt die Freiheiten der DFN-Low-Level-Policy und legt deren niedrige Sicherheitsanforderungen zu ihren Gunsten aus. Sie dient dazu, die UNI-CA und ihre Service-Angebote öffentlichkeitswirksam präsentieren zu können, Interesse bei potentiellen Nutzern zu wecken und die Zugangshürden für sie so niedrig wie möglich zu halten, indem die Low-Level UNI-CA quasi „zu den Nutzern kommt“. Die Low-Level-Zertifizierung kann dank des mobilen Zertifizierungsrechners auch außerhalb des Rechenzentrums durchgeführt werden, beispielsweise im Rahmen von Schulungen oder Kursen in anderen Teilen der Universität oder im Rahmen von Festivitäten (Sommerfest o.ä.).

Die Low-Level-CA verwahrt den geheimen Signierschlüssel auf der Low-Level-CA-Festplatte des Zertifizierungsrechners, ihre Zertifizierungen darf von einem CA-Mitarbeiter alleine vorgenommen werden (kein Vier-Augen-Prinzip). Die Low-Level-CA darf, wenn dies gewünscht wird, auch Schlüsselpaare für die Nutzer erzeugen, damit Interessenten, die noch nicht über ein Schlüsselpaar verfügen, es sich vor Ort bei der CA erzeugen und gleich zertifizieren lassen können.

Zertifiziert werden kann jeder, der bei entsprechenden Gelegenheiten (s.o.) bei der Low-Level CA vorstellig wird. Das Low-Level-Zertifikat ist also nicht an den Studenten- oder Mitarbeiter-Status an der UNI gebunden. (In den Situationen oder Umgebungen, in denen die Low-Level-CA arbeiten soll, wäre eine solche Überprüfung kaum sinnvoll möglich bzw. würde UNI-externe Interessenten abschrecken, statt sie für das Thema zu gewinnen.)

Vor der Low-Level-Zertifizierung findet lediglich eine Identitätsprüfung anhand eines Personaldokumentes statt; bei der Low-Level-Zertifizierung ist keine weitere Prüfung der Erreichbarkeit unter einer eventuell in der Benutzerkennung vorhandenen Mailadresse vorgesehen, und es findet auch keine Prüfung statt, ob der Antragsteller auch im Besitz des Private-Keys ist, der mit dem zu zertifizierenden Public-Key korrespondiert. Diese Prüfung wäre, wenn sich die UNI-CA im Rahmen einer Veranstaltung präsentiert, wohl kaum möglich, denn wer hat schon seinen Private-Key immer bei sich und wäre dann auch noch bereit, ihn vor anderen Menschen – womöglich noch auf einem fremdem Rechner! – einzusetzen?

Die hier beschriebene Low-Level UNI-CA ist also als eine „Jedermann-“ und „Schnupper-Zertifizierungsstelle“ konzipiert; sie ist sozusagen eine CA „zum Anfassen“ (oder Kennenlernen). Das mit ihrer Policy erreichte Sicherheitsniveau ist eher niedrig. Daher setzt die Low-Level CA auch keine Registrierungsstellen oder nachgeordneten CAs ein und führt auch keine Cross-Zertifizierungen mit anderen Zertifizierungsstellen durch, und ihre Zertifikate haben im Vergleich zur Medium-Level Zertifizierungsstelle eine kurze Gültigkeitsdauer. Auch ist keine Verlängerung der Low-Level-Zertifikate vorgesehen, nur die Ausstellung eines neuen Zertifikates nach erneutem persönlichem Kontakt. Dadurch soll ein Anreiz geschaffen werden, sich um die – in dieser Hinsicht vorteilhafte – Medium-Level-Zertifizierung zu bemühen.

4.6 Medium-Level CA

Die Sicherheitsanforderungen an den Betrieb der Medium-Level CA sind um einiges strenger als bei der Low-Level-CA, schließlich gibt auch die Medium-Level DFN-Policy höhere Anforderungen vor:

Die Zertifizierung durch die Medium-Level UNI-CA richtet sich an diejenigen, die entweder von vornherein hohe Sicherheitsanforderungen haben oder die sozusagen durch die Low-Level UNI-CA „auf den Geschmack gebracht“ worden sind und nun die Medium-Level-Zertifizierung ausprobieren oder von deren Vorteilen profitieren wollen.

Bei der Medium-Level CA wird der geheime Signierschlüssel verschlüsselt und Paßsatz- bzw. PIN-geschützt auf einem Wechselmedium (Diskette, ZIP-Disk, PCMCIA-Steckkarte, ggf. auch Chipkarte) gespeichert, das unter Verschluß gehalten wird, solange es nicht benutzt wird. Sowohl dieses Wechselmedium als auch die Festplatte, auf der sich die Daten und die Software für die Medium-Level-CA befinden, darf das schützende Rechenzentrum nicht verlassen; die Medium-Level-Zertifizierung findet nur dort statt. (Die *Registrierung* der Zertifizierungswilligen ist ggf. auch an anderen Standorten zulässig.)

Der Zugriff auf den Signierschlüssel der Medium-Level-CA kann nur von zwei befugten CA-Mitarbeitern zusammen erfolgen, so daß hier also das Vier-Augen-Prinzip durchgesetzt wird, um Fehlern oder Betrug durch „Insider“ vorzubeugen.

Das Schlüsselpaar für eine Medium-Level-Zertifizierung muß vom Schlüsselinhaber erzeugt worden sein; die Medium-Level CA generiert keine Nutzerschlüssel.

Für eine Medium-Level-Zertifizierung müssen die folgenden Bedingungen erfüllt sein:

- Der Zertifizierungswillige ist bei einem persönlichen Kontakt mit den CA-Mitarbeitern unter Vorlage eines gültigen amtlichen Personaldokumentes identifiziert worden
- der Betreffende ist Angehöriger (Student/Mitarbeiter) oder Projektpartner der UNI und kann dies durch geeignete Unterlagen nachweisen (z.B. Studentenausweis, Deckblatt des Gehaltsbogens, Kooperationsvertrag)
- die ggf. zu zertifizierende Mailadresse ist eine Mailadresse einer Einrichtung der UNI oder eines ihrer Projektpartner
- der Betreffende konnte von der UNI-CA unter dieser Mailadresse erreicht werden
- der Zertifizierungswillige hat gegenüber der UNI-CA nachgewiesen, daß er im Besitz des Private-Key ist, der zu dem zu zertifizierenden Public-Key gehört (sog. *proof of possession*, PoP – das kann z.B. dadurch geschehen, daß der Betreffende eine *challenge* genannte Zufallszahl mit dem geheimen Schlüssel signiert und das Ergebnis an die UNI-CA übermittelt, die dann unter Verwendung des Public-Keys prüft, ob die Signatur wirklich mit dem korrespondierenden geheimen Schlüssel geleistet wurde)

Die Zertifikate der Medium-Level-CA haben eine längere Gültigkeitsdauer als die der Low-Level-CA (nahe an der Obergrenze, die die DFN-Policy noch zuläßt). Eine Re-Zertifizierung kann auf eine entsprechende digitale signierte Nachricht hin erfolgen (u.U. mit erneutem Nachweis, daß der Schlüsselinhaber weiterhin im Besitz des passenden Private-Keys ist).

Die Medium-Level UNI-CA darf, wenn sie es für angebracht hält, Registrierungsstellen (RAs) einrichten und bei Bedarf auch nachgeordnete CAs aus anderen Einrichtungen der UNI zertifizieren. Es ist allerdings maximal eine CA-Ebene unterhalb der UNI-CA zulässig; die Sub-CAs dürfen also ihrerseits keine ihnen nachgeordneten CAs betreiben oder zertifizieren (wohl aber eigene Registrierungsstellen, falls der Bedarf dafür vorhanden ist). Darüber hinaus darf die Medium-Level- CA Cross-Zertifizierungen mit anderen Zertifizierungsstellen inner- und außerhalb der DFN-Zertifizierungshierachie eingehen, wenn ihre Verantwortlichen dies für sinnvoll und vorteilhaft halten.

4.7 Sicherheitsbedrohungen für die CA und Gegenmaßnahmen

The obvious mathematical breakthrough
to break modern encryption
would be development of an easy way
to factor large prime[!] *numbers.*
— BILL GATES, *The Road Ahead* [Gat95, S. 265]

Diese „Bedrohung“ für Public-Key-Kryptographie, die BILL GATES in seinem Buch formuliert, ist eine der harmlosesten für eine Zertifizierungsstelle, denn ihr Eintrittsrisiko ist gleich null. (Es wird niemals gelingen, *Prim*zahlen zu faktorisieren.) Die wohl eigentlich gemeinte Gefahr, daß große Nicht-Primzahlen effizient faktorisiert werden könnten, ist hingegen eine reale Bedrohung, die allerdings nicht nur Zertifizierungsstellen beträfe, wenn sie Wirklichkeit würde, sondern die dann ganze Klassen von Public-Key-Algorithmen unbrauchbar machen würde.

Insofern gelten für eine Zertifizierungsstelle auch alle anderen Bedrohungen, denen die jeweils verwendeten Public-Key-Verfahren allgemein ausgesetzt sind.[2] Sollten sich hier Angriffsmöglichkeiten ergeben, so wäre zwar auch die UNI-CA als Anwenderin der betroffenen Verfahren gefährdet, es gäbe aber für den betreffenden Angreifer wesentlich lohnendere Ziele.

Neben den kryptographischen Schwachstellen oder Angriffsmöglichkeiten bietet eine Zertifizierungsstelle einem Angreifer aber auch andere Ansatzpunkte für sein Vorgehen. Er muß beispielsweise nicht den geheimen Schlüssel auf kryptographischem Wege „brechen“, sondern er kann auch versuchen, sich Schwachstellen in den Arbeitsabläufen oder der Zugangskontrolle in der Zertifizierungsstelle zunutze zu machen.

Eine Zertifizierungsstelle ist spezifischen Bedrohungen ausgesetzt [ISO96]:

> unauthorized disclosure of the key
>
> corruption/unauthorized modification/deletion of the key [...]
>
> unauthorized revocation of the key
>
> delay in executing key management functions [...]

Die schwerwiegendste Bedrohung für eine Zertifizierungsstelle ist mit Sicherheit das Bekanntwerden oder der Mißbrauch ihres geheimen Zertifizierungsschlüssels, denn der CA droht dabei ein enormer Vertrauensverlust in den Augen aller derjenigen, die sie bislang für zuverlässig und glaubwürdig gehalten haben. Dies gilt es daher durch technische und sonstige Vorkehrungen zu verhindern.

Der Private Key der UNI-CA ist, um ein Bekanntwerden zu verhindern und um die Anforderungen der DFN-Zertifizierungsrichtlinien zu erfüllen, auf einem separaten Speichermedium aufzubewahren, wenn er nicht gerade benutzt wird. Das Speichermedium könnte in diesem Fall eine Diskette oder eine CD-ROM sein, die wie der Rechner selber bei Nichtgebrauch in einem Schutzschrank unter Verschluß gehalten wird. Der Schlüssel muß auf diesem Speichermedium durch eine PIN oder

[2] Beispiele für konkrete Angriffsmöglichkeiten speziell auf PGP nennt [inf96].

ein nicht-triviales Paßwort geschützt vorliegen. So ist sichergestellt, daß selbst der Zugriff auf diesen Schlüsseldatenträger einem Angreifer noch nicht den geheimen Schlüssel preisgibt. Durch diese Verschlüsselung würde es einem Angreifer auch erheblich erschwert, den CA-Schlüssel unbemerkt durch einen anderen Schlüssel seiner Wahl zu ersetzen.

> „Insgesamt muß festgestellt werden, daß das Sicherheitsbewußtsein generell schlecht ausgeprägt ist. Dies ist einer der wichtigsten Punkte, die im Rahmen einer Sicherheits-Policy angegangen werden muß, umso mehr als die Mitarbeiter mit ihrem Wissen nicht nur die wichtigste Ressource einer Unternehmung sondern auch das größte Bedrohungspotential darstellen.“
> (HANNES LÖHR, `<Loehr@TIC.Thyssen.com>`, in einer E-Mail an Ingmar Camphausen)

Um einen Insider-Mißbrauch, beispielsweise durch einen der CA-Administratoren, zu erschweren, wären zusätzliche Schutzvorkehrungen möglich. Sie würden aber die Zertifizierungsarbeit um einiges umständlicher machen, wären jedoch bei einer CA mit mittlerem oder hohem Sicherheitsniveau durchaus angemessen oder sogar unumgänglich.

Diese Vorkehrungen können z.B. darin bestehen, daß es einem einzelnen Mitarbeiter unmöglich gemacht wird, alleine sicherheitsrelevante Aktionen, beispielsweise mit dem geheimen Zertifizierungsschlüssel, auszuführen. Bei der UNI-CA ließe sich dies für die Medium-Level CA mit einfachen Mitteln erreichen, indem bei der Schlüsselerzeugung und initialen Festlegung der *Passphrase*, also des verlängerten Paßwortes, unter dem der geheime CA-Signierschlüssel verschlüsselt wird, zwei Personen jeweils einen Teil dieser Passphrase festlegen, ohne daß der jeweils andere diesen Teil erfährt. So ließe sich ein Vier-Augen-Prinzip für den Zugriff auf den geheimen Schlüssel erzwingen. (Diese einfache Umsetzung ist nur wirkungsvoll, wenn nicht einer der beiden Beteiligten dem anderen sein Paßwort verrät; sollte dies geschehen, hätte zumindest einer der Beteiligten alleine – unauthorisiert – Zugriff auf den Private-Key.)

Weitere Angriffe, die auf Angriffsmöglichkeiten im ganzen System einer Public-Key-Infrastruktur abzielen und eine CA nicht physikalisch, wohl aber logisch attackieren, schildert ZIESCHANG in [Zie97].

Eine *umfassende* Risiko-Analyse nebst Ableitung entsprechender Gegenmaßnahmen, wie sie an dieser Stelle eigentlich durchgeführt werden sollte, würde den Rahmen dieses Handbuchs sprengen, aus diesem Grund kann hier nur eine erste grobe Einschätzung vorgenommen werden.

Nach den Kriterien des BSI IT-Grundschutzhandbuches [BSI98] erscheint eine Einstufung der UNI-CA in den *mittleren* Schutzbedarf angemessen, da die darin genannten Orientierungskriterien für den Schutzbedarf „hoch“

- Verstöße gegen Vorschriften und Gesetze mit erheblichen Konsequenzen
- Vertragsverletzungen mit hohen Konventionalstrafen
- Möglicher Mißbrauch personenbezogener Daten hat erhebliche Auswirkungen auf die gesellschaftliche Stellung oder die wirtschaftlichen Verhältnisse des Betroffenen
- Eine Beeinträchtigung der körperlichen Unversehrtheit kann nicht absolut ausgeschlossen werden

- Ein IT-Systemausfall ist nur bis zu einem Tag tolerabel

als nicht zutreffend eingeschätzt werden. „Niedrig“ dürfte der Schutzbedarf einer Zertifizierungsstelle auf der anderen Seite auch nicht sein.

Bei maximal mittlerem Schutzbedarf liefert das GSHB mit seinem Baustein-Konzept eine Vorgehensweise, bei deren Umsetzung eine ausreichende Grundsicherung der einbezogenen Systeme erzielt wird. Dazu sollten die GSHB-Bausteine

- Organisation
- Personal
- Notfall-Vorsorgekonzept
- Datensicherungskonzept
- Gebäude
- Büroraum
- Datenträgerarchiv
- Schutzschränke
- Unix-System
- Tragbarer PC
- E-Mail

genutzt werden.

Ein wichtiger Punkt muß in diesem Zusammenhang hervorgehoben werden: Die getroffene grobe Einstufung kann nur den späteren Betriebszustand in der Form abbilden, in der er in dem hier vorliegenden Konzept antizipiert wurde. Wenn sich nach der Inbetriebnahme die Nutzung oder die Bedeutung der Zertifizierungsstelle erheblich verändert, muß die Schutzbedarfsermittlung regelmäßig anhand der dann vorliegenden konkreten Nutzungssituation und der dann geltenden Rahmenbedingungen überprüft und gegebenenfalls korrigiert oder präzisiert werden. Unter Umständen ergeben sich daraus später auch höhere Anforderungen an das Schutz- und Sicherheitsniveau, beispielsweise dann, wenn größere finanzielle Werte von der ordnungsgemäßen Arbeit der Zertifizierungsstelle abhängen oder wenn sich die gesetzlichen Rahmenbedingungen für ihre Zertifizierungstätigkeit gegenüber der jetzigen Situation erheblich verändern. Ein weiterer Anlaß, das Sicherheitskonzept und den Schutzbedarf einer erneuten Prüfung zu unterziehen, können sicherheitsrelevante Vorfälle sein, da sie möglicherweise Schwachstellen im existierenden Konzept offenbaren.

4.8 Die Zertifizierungsrichtlinien

Wie in Kapitel 3 beschrieben, gibt es mit PGP und X.509 zwei verschiedene etablierte Standards für das Format von Public-Key-Zertifikaten. Beide haben ihren jeweiligen Anwendungsbereich, in dem sie jeweils dominieren – PGP bei E-Mail-Kommunikation, X.509 bei anderen Anwendungen wie z.B. der gesicherten Nutzung des World Wide Web.

Bei der Festlegung von Zertifizierungsrichtlinien steht man nun vor der Frage, ob diese beiden unterschiedlichen Formate, die jeweils einige Besonderheiten nach sich ziehen, in einer allgemein geltenden Policy für die betreffende Zertifizierungsstelle zusammen abgehandelt werden sollten oder ob nicht getrennte Policies für jedes der beiden Formate (oder sogar für jede spezifische Anwendung) die bessere Lösung sind.

Eingedenk der Tatsache, daß die Policy zu allererst dazu dienen soll, denjenigen eine Einschätzung der Arbeit der betreffenden Zertifizierungsstelle zu ermöglichen, die sich auf ein von ihr ausgestelltes Zertifikat stützen wollen (vgl. Abschnitt 2.6.1), dürfte es sinnvoll sein, durch eine klare Trennung für mehr Übersichtlichkeit zu sorgen. Dies gilt umso mehr, als es in Zukunft eher mehr als weniger Anwendungen für Public-Key-Verfahren geben wird und insofern eine einzige, umfassende Policy immer umfangreicher und komplexer werden müßte. Außerdem braucht auf diese Weise der Zertifikatnutzer nur den Text zu lesen, der für ihn bzw. das vorliegende Zertifikatformat oder die jeweilige Anwendung relevant ist. Die jeweilige Policy kann so knapper und klarer formuliert werden, und es sind keine bedingten Teile oder Fallunterscheidungen in ihr erforderlich, die sie schwerer verständlich und unübersichtlich machen würden.

Grundlagen, auf denen alle Policies einer CA aufbauen – gewisse Procederes, Minimalanforderungen usw. – dürften für diejenigen, die das interessiert, auch beim Lesen und Vergleichen der spezifischen Policies deutlich werden. Eine Voraussetzung dafür ist natürlich, daß eine gewisse Einheitlichkeit sowohl in der Struktur der Dokumente (soweit möglich) als auch in den Sicherheitsanforderungen gewahrt bleibt. Das erleichtert es einem Zertifikatnutzer auch, sich in einer der anderen anwendungsspezifischen Policies einer Zertifizierungsstelle zurechtzufinden, wenn er eine oder mehrere andere davon bereits kennt.

Wichtig für die Glaubwürdigkeit der Zertifizierungsstelle ist dabei, daß die Policies zusammengenommen nicht widersprüchlich sind und daß, wenn Punkte auftreten, die widersprüchlich sind, verständlich erklärt wird, warum eine bestimmte Regelung in diesem Zusammenhang so und in einem anderen so getroffen wurde. Ungünstig wäre es, wenn beispielsweise in einer PGP-Policy für RSA-Schlüssel eine Mindestlänge von 1024 bit gefordert und mit der kryptographischen Unsicherheit von Schlüsseln, die wesentlich kürzer sind, begründet wird, dann aber in einer Policy derselben CA, die ein vergleichbares Sicherheitsniveau bei einer anderen Anwendung garantieren soll, plötzlich 512 bit RSA-Schlüssellänge als völlig ausreichend dargestellt werden.

Ein höheres Sicherheitsniveau beispielsweise einer Hochsicherheits-Policy gegenüber einer als weniger streng bezeichneten Policy für die gleichen Schlüssel- bzw. Zertifikatformate sollte sich im Sinne der Nachvollziehbarkeit und Glaubwürdigkeit auch in entsprechenden höheren Anforderungen und schärferen Bestimmungen in dieser Hochsicherheits-Policy widerspiegeln. Es wäre wenig überzeugend und Anlaß zu Mißtrauen, wenn beide Policies identische Anforderungen formulieren würden und dennoch die eine als „Hochsicherheits-Policy“ deklariert würde.

Im Anhang K dieses Handbuches ist ein Vorschlag für eine Low-Level UNI-CA-Policy wiedergegeben, die die in diesem Konzept beschriebenen Anforderungen in entsprechende Regeln umsetzt. Eine entsprechende Medium-Level Policy könnte darauf aufbauen; in 4.8.16 werden die Punkte benannt, in denen sie gegenüber dem Low-Level-Policy-Entwurf geändert oder erweitert werden müßte.

4.8.1 Welche Algorithmen sollen unterstützt werden?

Bei der PGP-Zertifizierung sollten nur RSA-Schlüssel im PGP 2.6-Format unterstützt werden, und zwar aus folgenden Gründen:

- Kompatibilitätsgründe: Mit der Einführung neuer PGP-Versionen können nicht mehr alle PGP-Anwender gesichert miteinander kommunizieren – zumindest dann nicht, wenn sie unterschiedliche Verfahren benutzen. Die PGP-Versionen bis PGP 2.6 benutzen RSA-Verschlüsselung und -Signaturen und MD5 als Hash-Algorithmus. PGP 5 und 6 verwenden hingegen den *Digital Signature Standard* (DSS) als Signatur- und Diffie-Hellman/ElGamal als Verschlüsselungsverfahren.
- Im DSS ist – ohne zwingenden Grund – eine maximale Schlüssellänge von 1024 bit festgelegt worden. Damit kann, wer standardkonform bleiben will, nicht mehr auf längere DSS-Schlüssel ausweichen, falls Fortschritte in der Rechnerleistung und/oder der Kryptanalyse 1024 bit-DSS-Schlüssel als unsicher erscheinen lassen.
- Es steht bislang keine stabile PGP-Version für Unix-Betriebssysteme zur Verfügung, die die neuen Schlüsselformate unterstützt. Die PGP-Versionen, die dies tun, sind bislang nur für die „Wintel-“ und die MacOS-Plattformen erhältlich. (Ob die neue Version 6.5.1i die erforderliche Stabilität aufweist, muß sich im Test und im Betrieb erst noch erweisen.)

Unter `http://www.shub-internet.org/why_not_pgp_5.html` finden sich weitere Gründe, die gegen das neue Schlüsselformat bzw. die neuen PGP-Versionen sprechen; eine neutrale Gegenüberstellung der Pro- und Contra-Argumente bietet [Sim99].

RSA-Signaturen haben gegenüber DSS-Signaturen noch den Vorzug, daß sie sich schneller verifizieren lassen als beispielsweise DSA-1024 bit-Signaturen. Da eine Signatur nur einmal erzeugt, aber gerade in Umgebungen mit Zertifizierungsstellen mehrfach verifiziert wird, ist in solchen Umgebungen RSA vorteilhaft [Wie98].

Ein Risiko bei dieser Entscheidung soll nicht verschwiegen werden: Auf die Message-Digest-Funktion, die bei den PGP-RSA-Keys verwendet wird (Ronald Rivests MD5 Algorithmus), ist inzwischen ein leichter kryptanalytischer Schatten gefallen. HANS DOBBERTIN hat gezeigt [Dob96], daß eines der Design-Ziele, die beim Entwurf von MD5 zugrunde gelegt wurden, verfehlt wurde. Damit ist das Verfahren noch nicht gebrochen worden, aber sein Ruf ist unter Kryptographen nicht mehr der Beste.

Ungeachtet dieses einen Nachteils sollte im Interesse der Gewährleistung einer Interoperabilität auch mit den Anwendern, die mit PGP 2.6 arbeiten, und angesichts der Vorzüge, die das RSA-Verfahren bietet, bis auf weiteres nur die Zertifizierung von RSA-Schlüsseln angeboten werden.

Die DFN-PCA unterstützt bislang ebenfalls ausschließlich RSA-Schlüssel, insofern gibt es, zumindest hinsichtlich der CA-Schlüssel, keine große Wahlmöglichkeit, sofern eine DFN-Zertifizierung angestrebt wird. (Allerdings könnte eine nachgeordnete CA sich einen RSA-Schlüssel DFN-zertifizieren lassen, mit diesem dann aber auch DSS/DH-Keys zertifizieren, indem sie eine der neueren PGP-Versionen einsetzt.)

4.8.2 Separate Signier- und Entschlüsselungsschlüssel

Die Medium-Level DFN-Policy schreibt für Zertifizierungsstellen eine Funktionstrennung bei ihren Schlüsseln vor: Für Zertifizierungsaufgaben und für die vertrauliche Kommunikation mit der jeweiligen Stelle müssen unterschiedliche Schlüssel verwendet werden.

Durch diese Trennung wird zum einen das kryptographische Material verringert, das einem Angreifer für die Kryptanalyse zur Verfügung steht (jeder der beiden Schlüssel wird weniger oft benutzt, als wenn nur ein Schlüssel für alle Anwendungszwecke verwendet würde), zum anderen werden die Folgen einer Schlüssel-Kompromittierung begrenzt – es kann dann im Falle eines Falles eben nur entweder die verschlüsselte Kommunikation vom Angreifer gelesen, *oder* Unterschriften der CA gefälscht werden, aber nicht beides, wenn nur ein Schlüssel kompromittiert wurde [For94].

Eine weitere vorbeugende Schadensbegrenzung ist möglich, indem die Schlüssel regelmäßig gewechselt werden und nur eine begrenzte Lebens- und Benutzungsdauer haben (siehe Abschnitt 4.8.7.2).

Ein weiterer Vorteil ergibt sich für die UNI-CA, wenn auch die Low-Level CA mit getrennten Schlüsseln arbeitet (obwohl sie dazu nach der entsprechenden DFN-Policy nicht verpflichtet wäre): Beide CAs können einen *gemeinsamen* Schlüssel als Kommunikationsschlüssel verwenden, so daß es für Kommunikationspartner der UNI-CA einfacher wird, verschlüsselt an sie zu mailen, weil dabei nicht noch einmal zwischen Low-Level- und Medium-Level-Verschlüsselungsschlüssel unterschieden werden muß.

Da bei PGP-RSA-Schlüsseln die PGP-Versionen mit Ausnahme von PGP2.6.3in keine Software-Unterstützung für eine automatische Berücksichtigung einer solchen Funktionstrennung bieten, sollte diese (auch) durch entsprechende Klartext-Angaben in der Benutzerkennung wie etwa *Zertifizierungsschlüssel* oder *SIGN ONLY key* signalisiert werden, so daß alle PGP-Anwender unabhängig von der verwendeten Software-Version erkennen können, für welchen Zweck ein bestimmter CA-Schlüssel vorgesehen ist.

4.8.3 Schlüssellängen

Bereits 1994 wurde ein 426-Bit-RSA-Key gebrochen – zwar dauerte dies damals acht Monate, aber dafür wurde auch nicht der effizienteste bekannte Algorithmus für diesen Angriff verwendet. 1977 hatte der Auslober dieser Aufgabe, RSA-Miterfinder RON RIVEST, noch prognostiziert, er würde deren Brechen nicht mehr miterleben [Fox95]. Anfang 1999 wurde bereits ein 465-Bit-RSA-Key (die “RSA-140-*Challenge*”) geknackt [Con99], und inzwischen sind sogar 512-Bit-RSA-Schlüssel gebrochen worden[3].

[3] `http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html`

Die Faktorisierung von RSA-140 hat nach konservativer Schätzung [Con99] rund 2000 MIPS-Jahre Rechenzeit beansprucht.[4] Es waren also rund $2000 \cdot 10^6 \cdot 365 \cdot 24 \cdot 3600 \approx 6 \cdot 10^{16}$ Rechenschritte zum Brechen von RSA-140 erforderlich. Bei RSA-155, dem geknackten 512-bit-RSA-Schlüssel, betrug der Rechenaufwand ca. 8000 MIPS-Jahre.[5]

Ein Linux-Cluster von Alpha-Rechnern mit einer Rechenleistung von etwa 20 GFLOPS kostet heute nur noch ca. 150 000 Dollar; man landet damit in den „Top 500" der Superrechner auf Platz 315 [Die98]. Nimmt man nun einmal sehr vorsichtig an, daß eine Fließkomma-Operation pro Sekunde einer Mikroprozessor-Instruktion entspricht (was so nicht der Fall ist – es ist, wie gesagt, eine sehr zurückhaltende Abschätzung), dann könnte ein solcher Cluster mit seinen 20 GFLOPS $= 2 \cdot 10^{10}$ Operationen pro Sekunde RSA-140 in weniger als $3 \cdot 10^6$ Sekunden = 876 Stunden oder einem guten Monat brechen. Die Faktorisierung eines 512-bit-RSA-Keys ist nach [Con99] rund sieben Mal aufwendiger als die von RSA-140. Bei entsprechend höherem Kapitaleinsatz ließe sich mit mehreren derartigen Clustern und einem Aufwand von vielleicht einer Million Dollar ein 512-bit-Schlüssel ebenfalls in etwa 30 Tagen, bei einem Etat von zehn Millionen Dollar also in weniger als drei Tagen brechen.

Mittlerweile plant das US-Energieministerium für die ersten Jahre des neuen Jahrtausends Rechner mit einer Leistung von bis zu 100 Tera-FLOPS, also mehr als 100 Billionen Fließkommaberechnungen pro Sekunde.

In Zukunft werden vielleicht DNA-Computing [Adl98] oder Quanten-Computing [Sho97, Rin97, Lom98] weitere, signifikante Steigerungen der Rechengeschwindigkeit ermöglichen. Da man außerdem nicht ausschließen kann, daß die kryptanalytischen Attacken weiter verbessert werden – das eingangs erwähnte Beispiel mit RON RIVEST belegt, wie schwierig es selbst für Fachleute auf diesem Gebiet ist, eine zutreffende längerfristige Prognose zu geben –, erscheint es umso angemessener, wenn die Medium-Level DFN-Policy *mindestens* 1024-Bit-Signierschlüssel verlangt, aber ausdrücklich sagt „oder nach Möglichkeit länger".

Die Regulierungsbehörde für Telekommunikation und Post (RegTP) als „zuständige Behörde" gemäß SigG schreibt folgende Schlüssellängen für RSA-Schlüssel vor[6]: 768 bit oder länger, wenn bis Ende des Jahres 2000, und 1024 bit oder länger, wenn bis Ende 2003 die Sicherheit des Verfahrens gewährleistet sein soll. Sie selber verwendete zunächst RSA-Schlüssel mit einer Länge von 1024 bit [BAz98], inzwischen solche mit 2048 bit[7].

SCHNEIER ist in [Sch96, S. 162] sogar noch vorsichtiger – oder unbefangener als eine staatliche Behörde? Er empfahl bereits 1995 die in der nachstehenden Tabelle genannten Schlüssellängen, wenn man auch in dem jeweiligen Jahr noch vor Angriffen eines einzelnen Unternehmens sicher sein will.

[4] Ein MIPS-Jahr ist die Zahl der Rechenoperationen, die ein Computer innerhalb eines Jahres ausführen kann, wenn er eine Million Anweisungen pro Sekunden (*million instructions per second*, MIPS) ausführen kann.

[5] `http://www.rsasecurity.com/news/pr/990826-2.html`

[6] Bekanntmachung im Bundesanzeiger Nr. 31 vom 14. Februar 1998 bzw. online unter `http://www.regtp.de/imperia/md/content/tech_reg_t/digisign/8.pdf`

[7] `http://www.regtp.de/tech_reg_tele/in_06-02-02-00-00_m/01/index.html`

Jahr	empfohlene Schlüssellänge
1995	1280 bit
2000	1280 bit
2005	1536 bit
2010	1536 bit
2015	2048 bit

Wenn man sich auch vor Geheimdiensten schützen will, sollten die Schlüssellängen nach SCHNEIER noch einmal um einiges größer ("substantially bigger") gewählt werden.

Angesichts dieser Angaben sollte die UNI-CA selber RSA-Schlüssel mit mindestens 1536 bit Länge verwenden und die Schlüssellänge gegebenenfalls in den nächsten Jahren beim Schlüsselwechsel erhöhen.

Hier muß die weitere Entwicklung besonders aufmerksam verfolgt werden, um gegebenenfalls rechtzeitig die Policies und die Länge der eigenen Schlüssel an sich ändernde Anforderungen oder bei Bekanntwerden neuer Angriffsmöglichkeiten anpassen zu können. Es empfiehlt sich, z.B. die diesbezüglichen Vorgaben der RegTP bzw. des BSI für SigG-CAs zu beachten. LENSTRA und VERHEULS Papier "Selecting Cryptographic Key Sizes"[8] kann dabei als Entscheidungshilfe mit einbezogen werden.

Im Sinne der Interoperabilität verschiedener PGP-Versionen bzw. -Schlüssel kann es sinnvoll sein, auch eine *Obergrenze* für die Schlüssellänge vorzugeben. Es sind Probleme einzelner PGP-Implementierungen bzw. auf einzelnen Betriebssystem-Plattformen mit Schlüsseln größer als 2048 bit bekannt. Daher wird für die UNI-CA-Policies vorgeschlagen, im Interesse der Interoperabilität bis auf weiteres neben der Mindest- auch eine Maximallänge für alle Schlüssel von 2048 bit vorzuschreiben.

4.8.4 Zulässige Benutzerkennungen

Bei der Low-Level-Zertifizierung ist eine Beschränkung der zulässigen Benutzerkennungen nicht sinnvoll möglich, da ja auch Universitätsfremde die Gelegenheit zur Zertifizierung wahrnehmen können und sollen. Deren Mailadressen ließen sich ja vom unvernetzten Zertifizierungsrechner aus gar nicht und auch sonst nicht unverzüglich prüfen – andernfalls wäre eine sofortige Zertifizierung nicht möglich (und damit die Grundidee der Low-Level CA hinfällig). In diesem Fall kann daher bei der Benutzerkennung lediglich darauf geachtet werden, daß sie den Vornamen und Namen des Schlüsselinhabers enthält.

Bei der Zertifizierung nach der Medium-Level-Policy sind die Abgabe des Antrags nebst Identitätsprüfung zeitlich von der eigentlichen Zertifizierung entkoppelt. Aufgrund der Prüfung der Mailadresse kommt es ja in jedem Fall zu einer gewissen Verzögerung. Da die Medium-Level UNI-CA nur UNI-Angehörige zertifizieren soll, ist eine Beschränkung auf solche Benutzerkennungen, die eine UNI-Mailadresse enthalten, sinnvoll und praktikabel.

Abweichungen von der genauen Schreibweise von Vorname(n) und Zuname im vorgelegten Personaldokument sind nur bei Sonderzeichen (Umlauten, Accents) erlaubt; die deutschen Umlaute und

[8] `http://www.cryptosavvy.com/cryptosizes.pdf`

das scharfe s können durch die übliche Umschreibung (‘ae’ für ‘ä’, ... ‘sz’ oder ‘ss’ für ‘ß’), andere akzentuierte Buchstaben durch den entsprechenden nicht-akzentuierten Buchstaben ersetzt werden (‘e’ statt ‘é’, ‘c’ statt ‘č’ oder ‘ç’ usw.).

Gegebenenfalls ist dem Benutzer Gelegenheit zu geben, innerhalb einer von der CA festzusetzenden Frist eine Benutzerkennung für den betreffenden PGP-Schlüssel nachzureichen, die die o.g. Anforderungen erfüllt.

Wenn bei nicht-lateinischen Schriften im vorgelegten Personaldokument die Schreibweise in lateinischen Buchstaben fraglich ist, sollte die Schreibung gewählt werden, die auch auf dem Studentenausweis oder dem Deckblatt des Gehaltsnachweises verwendet wird.

4.8.5 Prüfung der Mailadresse

Mit einer *verschlüsselten* Mail an den Zertifizierungswilligen kann die CA testen, ob derjenige unter einer bestimmten Mailadresse erreichbar ist. Die Test-Mail sollte eine wechselnde, nicht vorhersagbare Kennung oder Code-Nummer (*challenge*) enthalten, die der Empfänger an die CA zurücksenden muß. Die Mail der CA mit dieser Challenge muß verschlüsselt sein, damit nicht jemand, der zufällig oder absichtlich den Mail-Verkehr des eigentlichen Empfängers belauscht, diese Challenge mitlesen kann. Aus demselben Grund sollte sie von dem Betreffenden auch wieder *verschlüsselt* an die Zertifizierungsstelle zurückgeschickt werden. Zusätzlich kann die Zertifizierungsstelle in dieser Mail auch noch ein Codewort oder eine Geheimzahl angeben, mit der der Nutzer sich gegenüber der CA als berechtigt ausweisen kann, falls er einmal eine Sperrung des Zertifikates für seinen Schlüssel veranlassen wollen sollte. (Der Vorteil, dieses Codewort in derselben Nachricht wie die Challenge zu verschicken, besteht darin, daß die CA mit der Response des Zertifizierungswilligen quasi über eine Empfangsbestätigung desjenigen auch für das Codewort verfügt.)

4.8.6 *Proof of Possession*

Um neben der Erreichbarkeit auch noch zu prüfen, ob ein Nutzer auch im Besitz des Private-Keys ist, der zu dem zu zertifizierenden Public-Key gehört, muß der Betreffende die Challenge der Zertifizierungsstelle nicht nur verschlüsselt an sie zurücksenden, sondern er muß sie zuvor bzw. zugleich auch signieren. Damit erhält die CA, sofern diese Antwort (*response*) sie erreicht, einen nachprüfbaren Beleg dafür, daß der Absender über den geheimen Schlüssel verfügen muß, der mit dem öffentlichen Schlüssel korrespondiert, der zertifiziert werden soll.

4.8.7 Gültigkeitsdauer

4.8.7.1 Gültigkeitsdauer der Policy

Die Policy sollte eine Angabe enthalten, bis wann sie gilt oder, wenn ihre Gültigkeit nicht vorab zeitlich beschränkt ist, unter welchen Umständen und wann sie geändert werden kann.

Auch bei einer gegebenen Gültigkeitsdauer erscheint es im Sinne größtmöglicher Klarheit für die Nutzer überlegenswert, ob nicht in jedem Fall beschrieben werden sollte, wer über eventuelle Änderungen oder eine Neufassung oder Verlängerung der Zertifizierungsrichtlinien der betreffenden CA zu befinden hat.

4.8.7.2 Gültigkeitsdauer der Zertifikate

Die Gültigkeitsdauer der Zertifikate und eine eventuelle Verlängerungsmöglichkeit sollten sich an der Handhabung der Benutzerbereiche orientieren. Für die Low-Level-Policy könnte beispielsweise eine Gültigkeitsdauer von sechs Monaten sinnvoll sein, wenn die Benutzerbereiche der Studierenden an der UNI jedes Semester verlängert werden müssen.

Bei PGP-Zertifikaten kann eine Gültigkeitsdauer lediglich *implizit* durch die Gültigkeitsdauer des Unterschriftenschlüssels angegeben werden (vgl. [USC99]), insofern ergibt sich hier kein über das Jahr hinweg „gleitendes“ Gültigkeitsfenster, sondern es existieren Stichtage (Verfallsdatum des Signierschlüssels), an denen damit auch eine große Zahl von Zertifikate ungültig wird.[9] Bei X.509-Zertifikaten gibt es dieses Phänomen nicht, da jedes Zertifikat einen individuellen Gültigkeitsbeginn und ein individuelles Verfallsdatum aufweisen kann, das jeweils vom Zertifizierer bestimmt wird.

Die DFN-Policies legen für Nutzer-Zertifikate eine maximale Gültigkeitsdauer von einem Jahr fest. (Diese Aussage wird allerdings nur für X.509-Zertifikate gemacht, da ja bei den PGP-Zertifikaten nicht direkt eine Gültigkeitsdauer angegeben werden kann.) Dieser Punkt ist sicherlich diskussionswürdig, da bei strenger Prüfung insbesondere nach den Medium-Level Richtlinien eine längere Gültigkeitsdauer vertretbar sein müßte – selbst das Signaturgesetz sieht mit maximal fünf Jahren bzw. dem Ablauf der Eignung eines kryptographischen Verfahrens eine sehr viel höhere maximale Gültigkeitsdauer vor – und bei einer größeren Zahl ausgestellter Zertifikate ansonsten auch der Arbeitsaufwand durch ständige (Re-)Zertifizierungsanträge erhebliche Ausmaße annimmt. Da für PGP-Zertifikate in der DFN-Low-Level-Policy keine maximale Gültigkeitsdauer vorgeschrieben ist, bestünde hier nach den Buchstaben der Policy die Möglichkeit, einen beliebigen Gültigkeitszeitraum zu wählen. Geht man jedoch nach dem *Geist* der Policy, also der Absicht, die bei ihrer Abfassung zu Grunde lag, so muß man auch für PGP-Zertifikate dieselbe Obergrenze wie bei X.509-Zertifikaten von einem Jahr für die Gültigkeitsdauer ansetzen.

Für die UNI-CA Low-Level Policy wird daher eine maximale Gültigkeitsdauer der Nutzerzertifikate von 6 Monaten vorgeschlagen, für die Medium-Level Policy eine solche von 12 Monaten.

Für PGP, bei dem die Gültigkeitsdauer des Zertifizierungsschlüssels implizit über die Gültigkeitsdauer der Zertifikate mit entscheidet, wird vorgeschlagen, sich an den Anfangsterminen der Hochschulsemester zu orientieren. Da zum jeweiligen Semester- und dann auch CA-Schlüsselwechsel einige Arbeit anfällt und außerdem viele Studierende erst zum Vorlesungsbeginn wieder an ihrem Studienort sind, sollte der Gültigkeitszeitraum von altem und neuem Signierschlüssel sich leicht überlappen, damit es nicht zu Problemen an einem Übergangstag oder mit eventuell erforderlichen Nach-Zertifizierungen kommt (siehe 4.8.10). Der betreffende Schlüssel für ein neues Semester könnte dann schon einige Tage vor dessen Beginn generiert und z.B. auch schon von der DFN-PCA zertifiziert werden (der Beginn der Gültigkeitsdauer eines PGP-Keys ergibt sich implizit aus seinem Erstellungsdatum) und würde dann ab dem ersten Tag des neuen Semesters für die Zertifizierung eingesetzt.

[9] Eine Ausnahme stellt in dieser Hinsicht die aktuelle PGP-Version 6.5.1i dar: Sie ermöglicht es dem Nutzer erstmals (bei PGP), eine Gültigkeitsdauer für Zertifikate anzugeben.

Es läßt sich bei den PGP-Zertifikaten nach diesem Ansatz leider nicht vermeiden, daß der Gültigkeitszeitraum eines Zertifikates umso kürzer ist, je später im Semester es ausgestellt wurde. Dies wird sich erst ändern, wenn OpenPGP-kompatible Implementierungen zur Verfügung stehen und eingesetzt werden, da damit dann auch den *Signaturen* und *Zertifikaten* eine eigene, vom Signierschlüssel unabhängige Gültigkeitsdauer zugewiesen werden kann. (PGP 6.5.1i ist die erste Version, in der dieses Feature des OpenPGP-Standards unterstützt wird.)

Würde man anders vorgehen und beispielsweise das Verfallsdatum des Signierschlüssels so wählen, daß auch Zertifikate, die am Semesterende ausgestellt werden, noch ein Jahr gültig sind, bestünde das Problem, daß sich für früher im Semester erteilte Zertifikate eine Gültigkeitsdauer von mehr als einem Jahr ergäbe.

> „... sollte die Zertifizierungsinfrastruktur so ausgelegt sein, daß im Normalfall weder eine gültige Signatur zu einem bestimmten Prüfzeitpunkt als ungültig erscheint noch umgekehrt. [...] Eine Zertifizierungsstelle stellt keine rückwirkenden Zertifikate aus: Der Gültigkeitsbeginn eines angeforderten Zertifikats ist frühestens der Zeitpunkt der Zertifikatserstellung.“ [BS97, S. 335]

Diese Aussage betrifft ausschließlich X.509-Zertifikate; bei PGP beginnt die Gültigkeitsdauer eines Zertifikates mit dem Zeitpunkt seiner Erstellung (und der läßt sich ohne Manipulation der Systemzeit auf dem Zertifizierungsrechner nicht verändern, sondern hat automatisch die von BAUSPIESS und SCHEERHORN geforderte Eigenschaft).

> „Werden Schlüssel in einer Anwendung häufig gewechselt, dann verringert dies die Wahrscheinlichkeit einer Kompromittierung und damit auch den entstehenden Schaden. Verwendet man Schlüssel über einen langen Zeitraum, dann gibt es zu diesem Schlüssel eine große Menge an Klartext- und Schlüsseltextpaaren. Unter der Voraussetzung, daß man auf diese Paare zugreifen kann, können Attacken auf Kryptosysteme ... schnell zum Erfolg führen.“ [HW98, S. 161]

In diesem Sinne wäre es also günstig, kürzere Wechselintervalle bzw. Gültigkeitszeiträume für die CA-Schlüssel zu verwenden, insbesondere, wenn viele Schlüssel zertifiziert werden. Da zur Zeit bei der gängigen Public-Key-Software noch kein automatischer Schlüsselwechsel unterstützt wird, sollte die Gültigkeitsdauer der CA-Schlüssel und der Zertifikate nicht zu kurz gewählt werden, damit die Nutzer nicht mit dem manuellen Key-Update und einer Vielzahl von Schlüsseln der CA überfordert werden.

Der (oder die) Kommunikationsschlüssel einer CA ist bei Verwendung separater Schlüssel für Zertifizierung und vertrauliche Kommunikation unabhängig von der Gültigkeitsdauer der Zertifikate oder des Signierschlüssels. Da im Falle seiner Kompromittierung die Vertraulichkeit der gesamten damit verschlüsselten Kommunikation mit der Zertifizierungsstelle gefährdet ist, sollte, um das potentielle Ausmaß des Schadens in solch einem Falle zu begrenzen, der Kommunikationsschlüssel häufiger gewechselt werden. (Auch hier gilt wieder die vorstehend gemachte Einschränkung, daß die Nutzer nicht durch zu häufigen Schlüsselwechsel überfordert werden dürfen.) Es wird daher vorgeschlagen, für jedes Semester einen separaten Kommunikationsschlüssel zu benutzen. Dieser Gültigkeitszeitraum sollte für die Nutzer noch am einfachsten nachzuvollziehen sein. Alternativ könnte auch ein jährlich wechselnder Kommunikationsschlüssel vorteilhaft sein, da eventuelle externe Kommunikationspartner u.U. nicht so gut mit dem Beginn bzw. Ende der einzelnen Semester vertraut sind und dann aus Unwissenheit möglicherweise häufig einen veralteten Kommunikationsschlüssel verwenden.

4.8.8 Sperrung von Zertifikaten

4.8.8.1 Sperrung eines PGP-Zertifikats

Die Sperrmethode, ein Zertifikat auf eine „schwarze Liste“ zu setzen und es so als gesperrt oder ungültig zu kennzeichnen, läßt sich bei PGP-Zertifikaten nicht ohne weiteres umsetzen. PGP kennt kein Sperrlisten-Konzept, was dazu führt, daß alle Versuche, eine solche Lösung zu etablieren, an der fehlenden Unterstützung von Widerrufslisten durch die verfügbaren PGP-Implementierungen scheitern.

Grundsätzlich gibt es mindestens drei Möglichkeiten, wie die Sperrung eines PGP-Zertifikates durch den Zertifizierenden ausgedrückt werden kann:

- Sperrlisten (auch Widerrufslisten, CRLs)
- Zertifikat-Widerrufszertifikate (Certificate Revocations)
- Zertifizierung mit einem ausdrücklichen „Widerrufsschlüssel“

Die letzte Variante wird ebenso wenig wie die erste von den gängigen PGP-Programmen unterstützt und ist insofern ganz auf die Aufmerksamkeit des Nutzers angewiesen; außerdem läßt sie sich leicht fälschen, da auch ein Angreifer einen PGP-Schlüssel erzeugen kann, dessen Zertifikate auf den ersten Blick genau wie die des Widerrufsschlüssels aussehen. (vgl. 4.16.5)

Die Widerrufslisten erfordern ebenfalls die manuelle Intervention des Nutzers. Bei ihnen kommt erschwerend hinzu, daß nicht einmal ihr Format oder die Fundstelle, an der die Sperrliste einer CA zu finden sein sollte, einheitlich gehandhabt wird geschweige denn standardisiert ist.

Da PGP-„Widerrufslisten“ in den DFN-Zertifizierungsrichtlinien für PGP nicht zwingend vorgesehen sind, es dafür auch keinerlei standardisiertes Format oder eine verbreitete automatische Abfragemöglichkeit gibt, sollte auf Versuche, das CRL-Konzept von X.509 auch auf die PGP-Zertifikat-Handhabung zu übertragen, ganz verzichtet werden. Die wenigsten Anwender werden (so sie die betreffende Adresse überhaupt anhand Signatur unter einem Public-Key ermitteln können) den Web- oder FTP-Server einer CA nach einer PGP-Widerrufsliste durchsuchen, diese „händisch“ herunterladen, prüfen und ggf. in ihr enthaltene Zertifikat-Rückrufe berücksichtigen.

Einzig die Zertifikat-Widerrufszertifikate sind schon früh im PGP-Paketformat definiert worden [ASZ96, S. 15], PGP-Versionen bis PGP2.6.3ia werteten die entsprechende Struktur jedoch noch nicht aus. Neuere PGP-Versionen (ab PGP2.6.3in) „verstehen“ entsprechende Zertifikatwiderrufe und zeigen sie an, so daß es bei Verwendung dieser Versionen dem Unterzeichner möglich ist, eine Signatur unter einem PGP-Key nachträglich für ungültig zu erklären, also dieses PGP-„Zertifikat“ zu „sperren“. Diese Sperrungen werden ansonsten wie Unterschriftenpakete unter einem PGP-Public-Key behandelt und können genau wie die Public-Keys extrahiert und verteilt werden, z.B.

über den PGP-Keyserver-Verbund. Dieses Konzept des Zertifikat-Widerrufs ist auch von anderer Stelle unabhängig von PGP als Alternative zur Verwendung von Sperrlisten (CRLs) aufgegriffen und verallgemeinert worden [Rue95].

Eine vierte, nicht wirklich sinnvolle Variante besteht in der Möglichkeit, daß der Zertifizierer seinen Zertifizierungsschlüssel mit einem Key Revocation Certificate widerruft – dann aber sind *alle* jemals mit diesem Schlüssel unterschriebenen Zertifikate ungültig und nicht nur das eine, das ursprünglich gesperrt werden sollte. Diese Lösung verbietet sich daher in den meisten Fällen von selbst.

4.8.8.2 Sperrung auf Veranlassung des Schlüsselinhabers

Bevor die CA ein Zertifikat auf Veranlassung des Zertifikatnehmers sperrt, muß sie sich vergewissern, daß die entsprechende Aufforderung tatsächlich vom Zertifikatnehmer stammt und nicht von einer nicht-authorisierten anderen Person. Ohne Anspruch auf Vollständigkeit sind die folgenden drei Möglichkeiten gegeben, wie sich der Betreffende eindeutig identifizieren kann:

- durch persönliches Erscheinen bei der CA und Vorlage eines Personaldokumentes (wie bei der Beantragung des Zertifikats)
- durch eine signierte E-Mail an die CA mit dem Widerrufswunsch, die mit dem geheimen Schlüssel signiert ist, dessen öffentliche Komponente gesperrt werden soll
- durch Angabe eines geheimen Codewortes per Mail oder auch telefonisch, das vertraulich zwischen dem Schlüsselinhaber und der Zertifizierungsstelle für genau diesen Fall vereinbart wurde (kann z.B. als Teil einer verschlüsselten *Challenge*-Mail geschickt worden sein, mit der die CA die Erreichbarkeit des Antragstellers unter der zu zertifizierenden Mailadresse getestet hatte)

4.8.8.3 Sperrung auf Veranlassung der CA

Eine sinnvolle Nutzung der Zertifikate als digitaler Werks- oder Studentenausweis wäre nur dann möglich, wenn ein Zertifikat für den Schlüssel eines ausscheidenden Universitätsangehörigen zeitnah zu dessen Ausscheiden gesperrt würde. Das setzt wiederum voraus, daß die CA über Mitarbeiter bzw. Studierende, die die UNI verlassen, umgehend informiert wird. Dies könnte zentral durch die Immatrikulations- bzw. Personalstelle passieren, es wäre aber – zumindest bei den Mitarbeitern – vorstellbar, daß sie sich selber in der Zertifizierungsstelle abmelden müssen, diese quasi auf einem entsprechenden Laufzettel quittieren müßte, daß der Betreffende sein Ausscheiden angezeigt hat. Beide Varianten dürften jedoch mit einigem Aufwand verbunden sein.

Ein Abgleich der Personal- oder Studierendenliste mit den Zertifikatnehmern der CA wäre eine andere Alternative, die allerdings aus Datenschutzgründen problematisch erscheint und die, da der Abgleich vermutlich nicht täglich stattfinden könnte, auch nur eine grobe zeitliche Auflösung erreichen würde, z.B. ein quartalsweiser oder monatlicher Abgleich. Damit wäre aber an einen Einsatz der Zertifikate als elektronischer Werks- oder Studentenausweis der UNI kaum zu denken, es sei denn, maximal drei Monate bzw. vier Wochen mögliche Mißbrauchsfrist erschienen vertretbar. (Auf der anderen Seite müßte man hier sicher auch berücksichtigen, wie genau herkömmliche Ausweise oder Nachweise wie der Studentenausweis den Status des Inhabers zu einem bestimmten Zeitpunkt wiedergeben.)

Das UNI-Immatrikulationsbüro übermittelt schon jetzt jeweils zu Beginn des Semesters die zurückgemeldeten bzw. neu-immatrikulierten Studierenden an das Rechenzentrum, so daß die Benutzerbereiche ausgeschiedener Studenten gesperrt werden können. Anhand dieser Informationen wäre es zusätzlich möglich, noch nicht abgelaufene Medium-Level-Zertifikate zu sperren, falls der Betreffende seinen Status als Studierender verloren hat. (Wenn derjenige Mitarbeiter der Universität geworden sein sollte, kann er seinen Schlüssel neu zertifizieren lassen.) Auf diese Weise ist zwar noch keine hohe zeitliche Auflösung möglich, aber zu Beginn eines neuen Semesters wären dann nur noch die alten Zertifikate derjenigen Studierenden weiter gültig, die nach wie vor an der UNI eingeschrieben sind. Diese Sperrung ist auch deswegen erforderlich, weil der Benutzername einige Monate nach der Sperrung bzw. Löschung wieder neu vergeben werden kann. Im Extremfall würde dann ein Namensvetter des vorherigen Inhabers dieselbe Mailadresse innerhalb der UNI bekommen können wie dieser. Damit es dann nicht zu der unerwünschten Situation kommt, daß die UNI-CA womöglich für dieselbe Benutzerkennung in einem Schlüssel zwei unterschiedliche Schlüssel für zwei Personen zertifiziert hätte und beide Zertifikate gleichzeitig gültig wären, sollte mit der Sperrung bzw. Löschung eines Benutzerbereiches auch ein eventuell für die zugehörige Mailadresse ausgestelltes Zertifikat der UNI-Zertifizierungsstelle widerrufen werden.

Ähnlich sieht der Ablauf bei den Benutzerbereichen der Universitätsmitarbeiter aus: Dort übermittelt die Personalstelle die Daten aller Mitarbeiter an das Rechenzentrum, so daß deren Benutzerbereiche verlängert (und die der ausgeschiedenen Mitarbeiter entsprechend gesperrt) werden können.

4.8.9 Re-Zertifizierung

Eine Re-Zertifizierung oder auch Nachzertifizierung ist nur für die Medium-Level-Zertifizierung vorgesehen. Nur dort sind die Sicherheitsprüfungen bei der Erst-Zertifizierung so gründlich, daß eine Re-Zertifizierung aufgrund eines digital signierten Antrages vertretbar erscheint. Außerdem soll für die Zertifikatnehmer ein Anreiz geschaffen werden, sich nach der Medium-Level Policy zertifizieren zu lassen. Die Möglichkeit, dort sein Zertifikat verlängern zu lassen und nicht jedes Semester wieder persönlich vorstellig werden zu müssen, dient diesem Ziel ebenso wie die grundsätzlich längere Gültigkeitsdauer der Medium-Level Zertifikate.

Die Re-Zertifizierung findet nur auf Antrag des Benutzers statt, wenn der Signierschlüssel, mit dem das Zertifikat für seinen Key unterschrieben wurde, abläuft. Bei einem Schlüsselwechsel auf Seiten des Benutzers steht es dem Betreffenden frei, eine Zertifizierung des neuen Schlüssels zu beantragen, indem er persönlich bei der CA vorstellig wird. Ein Zertifizierungsantrag für einen neuen Schlüssel kann nicht auf elektronischem Wege gestellt werden; auf diese Weise ist nur die Re-Zertifizierung eines bereits zertifizierten Schlüssels (also eine „Verlängerung“ von dessen Zertifikat) möglich. Es bleibt dem Nutzer überlassen, ob er nach Erhalt des Zertifikates für seinen neuen Schlüssel die Sperrung des alten Zertifikates veranlaßt (oder selbst einen Schlüssel-Widerruf generiert) oder ob sein alter und neuer Schlüssel beide weiter gültig sein sollen.

Eine Re-Zertifizierung kann bei Bedarf zusätzlich auch davon abhängig gemacht werden, daß der Schlüsselinhaber weiterhin Universitätsangehöriger ist, also nicht inzwischen seinen Mitarbeiter-oder Studentenstatus verloren hat. Dies könnte ggf. anhand der Daten der Personalstelle oder des Immatrikulationsamtes verifiziert werden.

4.8.10 Key-Rollover

Die leicht überlappende Gültigkeitsdauer, die für die CA-Schlüssel vorgeschlagen wurde (s. 4.8.7.2), hat auch noch einen anderen Grund:

> „[Die CA muß] dafür sorgen, daß ihr bis zum Ende der Gültigkeit ihres bisherigen Schlüssels ‘genügend’ Zeit verbleibt, um beantragte Nachzertifizierungen zu erfüllen. Offenbar hängt die dafür erforderliche Zeitspanne davon ab, wieviele Nachzertifizierungen die CA auszuführen hat.“ [BS97, S. 336]

Kann dies nicht gewährleistet werden, so wären Zertifikatinhaber trotz rechtzeitigen Antrags auf Nachzertifizierung nach Ablauf des alten CA-Schlüssels ohne gültiges Zertifikat. Diese unerfreuliche Situation sollte wenn irgend möglich vermieden werden.

4.8.11 Gruppenschlüssel

Die Zertifizierung von Gruppenschlüsseln durch die UNI-CA ist vorerst nicht vorgesehen. Deren Zertifizierung erscheint als nicht sinnvoll, da eine Zurechenbarkeit nicht gegeben ist (jedes Gruppenmitglied kann eine mit diesem Schlüssel signierte Nachricht unterschrieben haben) und außerdem Gruppenschlüssel auf einfache Weise vereinbart werden können, wenn die Gruppenmitglieder über individuelle Schlüssel verfügen: Das Ziel einer Zertifizierung eines Gruppenschlüssels durch eine CA läßt sich ebenso gut erreichen, indem einer der Ansprechpartner der Gruppe mit seinem Schlüssel einen separaten Gruppenschlüssel signiert.

Gruppenschlüssel bringen weiterhin das Problem mit sich, daß beim Ausscheiden eines Gruppenmitgliedes der Schlüssel gewechselt werden müßte, wenn die Vertraulichkeit der auf diese Weise geschützten Kommunikation weiterhin sichergestellt bleiben soll.

Die einzige Ausnahme von dieser Regel stellt der Kommunikationsschlüssel der Zertifizierungsstelle selber dar; er darf von der CA zertifiziert werden.

4.8.12 Pseudonyme und anonyme Zertifikate

Da wissenschaftliche Arbeiten in der Regel unter dem eigenen Namen und nicht unter Pseudonym veröffentlicht werden, liegt es nahe, bis auf weiteres keine pseudonymen oder anonymen Zertifikate ausstellen.

Andererseits fordern die Datenschutzbeauftragten mit einiger Berechtigung, daß es im Sinne der Datensparsamkeit möglich sein muß, Dienste auch anonym oder unter Pseudonym zu nutzen, sofern die technisch möglich ist. Insofern würde also auch das Angebot einer anonymen oder pseudonymen Zertifizierung durchaus Sinn machen.

Da die DFN-PCA selbst bislang noch keine Erfahrung mit entsprechenden anonymen oder pseudonymen Zertifizierungen gesammelt hat, können an dieser Stelle noch keine Hinweise oder Empfehlungen ausgesprochen werden.

4.8.13 Einbindung in die DFN-Zertifizierungshierarchie

4.8.13.1 Registrierungsstellen

Die DFN-Zertifizierungshierarchie sieht vor, daß von den Zertifizierungsstellen in den einzelnen DFN-Mitgliedseinrichtungen mehrere Registrierungsstellen (RAs) betrieben werden können, um für die Nutzer den Weg zur Zertifizierungsstelle zu verkürzen. Zertifizierungswillige können ihren Zertifizierungsantrag statt bei der CA bei einer der RAs stellen, die dann auch gleich die Identitätsprüfung vornimmt.

Bei der Entscheidung für oder gegen Registrierungsstellen sollte abgewogen werden zwischen den Vorteilen durch die Nähe zum Benutzer und die leichtere Erreichbarkeit dieser „Außenstellen“ der CA und den Nachteilen, die sich durch eine zu starke Aufsplitterung ergeben (Mehrfacharbeit, Hardware-Ressourcen in der RA). Die (ggf. erst erwartete) Nachfrage sollte ausreichend groß sein, so daß sich der organisatorische Mehraufwand einer zusätzlichen Registrierungsstelle lohnt.

Potentielle RA-Standorte sind zum einen Teile der Universität, die sehr weit vom Campus entfernt liegen, so daß es den dort Beschäftigten oder Studierenden nicht zugemutet werden soll, für eine Zertifizierung einen längeren Anfahrtsweg oder eine längere Abwesenheit vom Arbeitsplatz in Kauf nehmen zu müssen.

Registrierungsstellen sind für die Zertifizierung nach der *Low-Level* Policy nicht erforderlich, da ja die Low-Level CA selber mobil ist und bei Bedarf vor Ort zertifizieren kann.

4.8.14 Nachgeordnete Zertifizierungsstellen

Wenn die Nachfrage nach Zertifizierungsdiensten so stark zunimmt, daß sie selbst mit Registrierungsstellen an den Orten mit besonders großer Nachfrage kaum zu befriedigen ist, oder wenn aus anderen Gründen eine eigene Zertifizierungsstelle in einer Einrichtung der UNI betrieben werden soll, dann dürfen nach der Low-Level- und der Medium-Level DFN-Policy auch nachgeordnete Zertifizierungsstellen (Sub-CAs) innerhalb einer DFN-Mitgliedseinrichtung etabliert werden. Diese können dann eigenverantwortlich die Schlüssel ihrer Nutzer (z.B. der jeweiligen Fachbereichsangehörigen) zertifizieren, stehen aber unter der Aufsicht der Zertifizierungsstelle der gesamten Mitgliedseinrichtung und müssen sich an deren Richtlinien (und die des DFN) halten.

RAs und Sub-CAs stellen eine Möglichkeit der „Lastverteilung“ bzw. -entzerrung dar, erfordern aber auch einigen zusätzlichen Betreuungs-, Prüf- und Koordinationsaufwand von seiten der CA.

4.8.15 Abweichungen von den DFN-Policies?

CAs, die nach einer der DFN-Policies zertifiziert werden wollen, müssen die entsprechende Policy erfüllen/einhalten. Das muß aber nicht bedeuten, daß eine Abweichung von der betreffenden Policy immer – quasi zwangsläufig – zur „Disqualifikation“ für eine Zertifizierung durch die DFN-PCA führen muß. Vielmehr steht es der jeweiligen CA durchaus frei, in einzelnen Punkten *schärfere* Anforderungen zu stellen als in der entsprechenden DFN-Policy. In 5.1 *Regeln für die Zertifizierung von CAs der DFN-WWW-Policy* heißt es dazu beispielsweise:

> „[...] über diese Policy hinausgehende Richtlinien können bei Bedarf von dieser CA [der obersten innerhalb der jeweiligen Organisation, sozusagen die organisationsweite „Root-CA“] in einer eigenen Policy festgelegt werden.“

Mißverständliche, widersprüchliche oder unklare Formulierungen sollten natürlich, wenn eine eigene Policy formuliert wird, darin nach Möglichkeit vermieden werden.

Wenn die Policy in verschiedenen Formaten (HTML, ASCII, Postscript) bereitgehalten wird, sollte unbedingt sichergestellt sein, daß diese Dokumente ungeachtet der unterschiedlichen Formate *inhaltlich* identisch sind und sich nicht in Details unterscheiden, was zu Irritationen und einem Glaubwürdigkeitsdefizit der betreffenden Zertifizierungsstelle führen könnte.

Ein Hinweis, wer nach dem Ablauf der aktuell gültigen Policy über die Annahme einer neuen oder die Verlängerung der alten Policy entscheidet, könnte zu mehr Transparenz für die Nutzer beitragen. Auch das Procedere, in dem dies geschieht (Werden mögliche Änderungen zuvor irgendwo diskutiert? Wer ist an der Überarbeitung beteiligt? In welcher Form wird letztlich entschieden?), könnte in der eigenen Policy dokumentiert werden.

Abschnitt 4 über die Sicherheitsanforderungen an die eingesetzte Rechentechnik ist sowohl in der Medium-Level als auch in der Low-Level DFN-Policy mit „Sicherheit der PCA-Ausstattung“ überschrieben. Diese Überschrift sollte bei einer eigenen Policy (und wird bei einer Überarbeitung der DFN-Policies) dem tatsächlichen Abschnittsinhalt angepaßt werden, der jeweils nicht nur die PCA-, sondern auch die CA-, RA- und Nutzer-Rechnerausstattung umfassen muß (bzw. umfaßt).

4.8.16 Unterschiede zwischen der Low- und der Medium-Level Policy

Um aus der der Low-Level UNI-CA-Policy eine Medium-Level Policy im Sinne der in Abschnitt 4.6 gegebenen Beschreibung der Medium-Level-Anforderungen zu entwickeln, müßte die im Anhang K wiedergegebene Low-Level-Policy in folgender Hinsicht bzw. in folgenden Punkten geändert bzw. ergänzt werden:

- höhere Sicherheitsanforderungen an die CA:
  - Aufbewahrung des geheimen CA-Signierschlüssels auf externem Datenträger
  - Vier-Augen-Prinzip beim Zugriff auf den geheimen CA-Signierschlüssel ∗
- höhere Sicherheitsanforderungen an die Zertifikatnehmer:
  - Prüfung der UNI-Zugehörigkeit
  - nur UNI-Mailadressen zertifizierbar
  - Prüfung der Mailadresse durch Zusenden einer „Challenge“ ∗

  - Nachweis über Besitz des korrespondierenden Private-Keys $*$

- andere Gültigkeitsdauer der Schlüssel bzw. damit auch der Zertifikate
- keine Schlüsselgenerierung für den Zertifikatnehmer (außer bei der Zertifizierung der Schlüssel der CA selbst)
- Sub-CAs sind möglich $*$, deren Zertifizierung etc. muß beschrieben werden
- Registrierungsstellen sind möglich $*$; es muß festgelegt werden, welche technischen Voraussetzungen dort gegeben sein müssen und wie die Weiterleitung der Zertifizierungsdaten erfolgen soll
- Cross-Zertifizierungen mit anderen CAs sind erlaubt $*$
- eine Re-Zertifizierung ist vorgesehen, die entsprechenden Abläufe müssen beschrieben werden $*$

Die mit einem $*$ markierten Punkte sind nach der DFN-Medium-Level-Policy nicht zwingend erforderlich; sie ergeben sich aus den in diesem Konzept vorgeschlagenen Medium-Level-Anforderungen.

Einige der entsprechenden Passagen könnten – unter Berücksichtigung der im vorigen Abschnitt gemachten Vorschläge – mit den erforderlichen Anpassungen aus der Medium-Level-Policy der DFN-PCA übernommen werden.

4.9 X.509-Zertifizierung

Auf die Ausstellung von X.509-Zertifikaten kann hier aus Platzgründen nicht in allen Einzelheiten eingegangen werden. Es sollen daher nur die wesentlichen Unterschiede zur PGP-Zertifizierung umrissen werden. Eine detaillierte Beschreibungen der X.509-Zertifizierung unter Verwendung der frei verfügbaren X.509-Software OpenSSL findet man in Teil II dieses Handbuches oder auch in [Rei97a, Rei98].

Der vielleicht grundlegendste Unterschied zur PGP-Zertifizierung besteht darin, daß X.509 im Regelfall eine Zertifizierung in einer Richtung zwischen ungleichen Partnern vorsieht: Eine herausgehobene Zertifizierungsinstanz bestätigt die Authentizität des Schlüssels eines Zertifikatnehmers (der ein Nutzer oder eine untergeordnete Zertifizierungsstelle sein kann). Der Zertifikatnehmer wird bei X.509 in den meisten Fällen nicht seinerseits ein Zertifikat für die CA ausstellen, während das bei der PGP-Zertifizierung „von gleich zu gleich“ den Normalfall darstellt.

Aufgrund dieser Situation kann die Zertifizierungsstelle, anders als bei PGP, alleine entscheiden, welche Angaben sie in ein Zertifikat mit aufnimmt. Der Zertifikatnehmer kann zwar seine entsprechenden Wünsche an die CA übermitteln, ob sie berücksichtigt werden, bleibt allein die Entscheidung der Zertifizierungsstelle. Die ist also anders als bei der Benutzerkennung von PGP nicht an einen vom Nutzer vorgegebenen Namen gebunden, sondern kann diesen bei Bedarf selbst modifizieren oder ergänzen (z.B. falls er ansonsten nicht eindeutig oder schon anderweitig vergeben ist), wobei die Namensgebung in X.509 auf dem Konzept der strukturierten *Distinguished Names* basiert. Dabei setzt sich ein vollständiger Name aus mehreren einzelnen Komponenten mit vorgegebener Struktur und Semantik zusammen.

X.509-Zertifikate, insbesondere jene nach der neuesten Version des Standards, weisen erheblich mehr Bestandteile auf als PGP-Zertifikate (siehe dazu auch Kapitel 3.1). Dies ermöglicht anders als bei PGP u.a. eine flexiblere und explizite Angabe der individuellen Gültigkeitsdauer für jedes einzelne X.509-Zertifikat. (Dadurch werden manche Abläufe gegenüber der PGP-Zertifizierung erheblich einfacher.) Da außerdem im X.509-Standard bereits Komponenten im Zertifikat für Funktionsbeschränkungen („nur zum Signieren“, „nur zum Verschlüsseln“, „nur zur Zertifizierung“ u.a.) definiert sind, ist die Umsetzung der in der DFN-Policy für CAs nahe gelegten (Low-Level-Policy) bzw. geforderten Verwendung (Medium-Level-Policy) getrennter Schlüsselpaare erheblich einfacher und wird besser durch existierende Software unterstützt.

In X.509 sind von der ersten Version von 1988 an Widerrufslisten als Möglichkeit vorgesehen, einmal erteilte Zertifikate zu sperren. Daher muß jede Software, die X.509-Unterstützung für sich reklamiert, mit derartigen Sperrlisten umgehen können. Auch das *Format* der Sperrlisten ist im Standard genau definiert, so daß hier nicht so unterschiedliche, individuell gewählte Sperrlisten-Formate entstehen, wie dies bei PGP leider der Fall ist.

X.509 bietet Möglichkeiten, bei der Zertifizierung einer nachgeordneten CA im Zertifikat für diese CA festzulegen, daß sie beispielsweise nur für einen bestimmten Namensraum zuständig ist und nur für Nutzer oder Sub-CAs mit bestimmten Namensbestandteilen Zertifikate ausstellen darf.

4.10 Erstellung einer X.509-UNI-CA-Policy

Mit dem Entwurf für eine PGP-Low-Level-Policy für die UNI-CA im Anhang K dieses Handbuches und den in Abschnitt 4.8.16 genannten Änderungen, die daraus eine Medium-Level-Policy machen würden, ist die PGP-Zertifizierung nach diesem Konzept abgedeckt. Da sich, wie im vorigen Abschnitt beschrieben, die X.509-Zertifizierung in einigen Punkten nicht unerheblich von der von PGP-Schlüsseln unterscheidet, stellt sich nun die Frage, wie gegenüber einer Medium-Level-PGP-Policy eine X.509-Zertifizierungsrichtlinie mit vergleichbaren Sicherheitsanforderungen aussehen könnte.

Die nachfolgenden Punkte beschreiben grob die Arbeitsschritte, die unternommen werden könnten, um aus der Medium-Level PGP-Policy eine X.509-Policy für die UNI-CA zu machen:

- Die Abschnitte über die Wahl einer PGP-Benutzerkennung könnten entfallen; stattdessen müßte festgelegt werden, welche Anforderungen zulässige X.500-Distinguished Names erfüllen müssen. Der X.500-Name der UNI-CA müßte genannt werden. Alle Angaben, die bei PGP-Schlüsseln mangels anderer Möglichkeiten im Klartext in der Benutzerkennung angegeben wurden (Anwendungsbeschränkungen für den Schlüssel, Gültigkeitsdauer), müßten in ihrem Format nicht mehr explizit in der Policy festgelegt (höchstens referenziert oder erklärt) werden, weil sie in X.509 bereits definiert sind.
- Passagen, die sich ausschließlich auf PGP beziehen (wie die Nennung der unterstützten Schlüsselformate), könnten entfallen bzw. wären durch entsprechende Angaben für X.509-Zertifikate zu ersetzen.

- Die PEM-Namensunterordnung (Namensbestandteile des CA-Distinguished Names müssen unverändert im Nutzernamen enthalten sein) verhindert eine Zertifizierung für Externe. Das heißt, daß die X.509-CAs als reine *organizational CAs* fungieren, ohne daß dies ausdrücklich in der DFN-Policy gesagt wird. Wenn diese Beschränkung beabsichtigt ist, sollte sie auch offen genannt werden; wenn nicht, müßte die Policy an dieser Stelle weniger strikt formuliert werden.

- Die Sperrung von Zertifikaten erfolgt mittels Widerrufslisten (CRLs). Einzelheiten zu deren Veröffentlichung müßten genannt werden.

- Die Abschnitte zur Gültigkeitsdauer könnten kürzer gefaßt werden und die Gültigkeitsdauer der Zertifikate unabhängig von der der CA-Schlüssel oder von Stichtagen regeln.

- Der Verweis auf die Medium-Level-Policy müßte durch einen entsprechenden auf die Low-Level-Policy ersetzt werden.

- Eine ausdrückliche Prüfung mittels Challenge-Response-Verfahren, ob der Nutzer auch über den Private-Key verfügt (PoP), der zu dem zu zertifizierenden Public-Key gehört, könnte entfallen, wenn der elektronisch übermittelte X.509-Zertifizierungsantrag (*Certificate Request*) bereits signiert ist. Die entsprechenden Passagen zur PoP-Challenge könnten entfallen oder allgemeiner formuliert werden.

- X.509-Zertifikate werden nicht auf PGP-Keyservern, sondern mittels anderer Dienste zugänglich gemacht, z.B. im X.500-Verzeichnisdienst veröffentlicht. Der betreffende Abschnitt wäre entsprechend umzuformulieren.

- Besonderheiten, die bei speziellen Formen der X.509-Zertifizierung (Zertifizierung von WWW-Servern, Software-Publisher-Zertifikaten oder Zertifizierung von WWW-Clients) vorzusehen sind, könnten aus der WWW-Policy der DFN-PCA [KL99] übernommen werden. Gleiches gilt für die Zertifikat-Erweiterungen.

Selbstverständlich wäre es ebenso möglich, die entsprechende DFN-PCA-Policy zur X.509-Zertifizierung als Grundlage zu verwenden und dann anhand der in den vorigen Abschnitten beschriebenen Änderungsvorschläge und Besonderheiten der UNI-CA daraus auf diesem Wege eine eigene (Medium-Level) UNI-CA X.509-Policy zu erstellen.

4.11 Datensicherung

Eine regelmäßige Datensicherung ist auch bei der UNI-CA geboten, liefe diese doch sonst Gefahr, bei einem Hardware-Defekt, einem Brand o.ä. erst ihre Arbeitsmittel, dann ihr Kapital – das Vertrauen der Nutzer – und dann schließlich ihre Arbeitsgrundlage zu verlieren. Der Software- und Datenbestand des Zertifizierungsrechners sollte daher in kurzen Abständen (wöchentlich) komplett auf Magnetband gesichert werden. Eine Datensicherung über eine Netzwerkverbindung zu einem anderen Rechner darf nicht erfolgen; der Zertifizierungsrechner muß ausschließlich "standalone" betrieben werden.

Allerdings gilt es dabei eines zu beachten, um sich den Umgang mit den Sicherungskopien nicht unnötig kompliziert zu machen:

> “A digital signature private key is never backed-up – if it is lost, a new key pair is simply generated for the signer. ... A digital signature private key is never archived ...” [For94, S. 1]
>
> “Non-repudiation requires that users generate and control access to [their] signing keys. Unlike decryption keys, backing up signing keys is neither required nor acceptable.” [Cur98, S. 3]

Es besteht in der Tat kein Grund, eine Sicherungs- oder Archivkopie des Signierschlüssels anzufertigen: Wenn der Schlüssel durch Löschung oder einen Schaden des Datenträgers, auf dem er gespeichert ist, nicht mehr verfügbar ist, wird sein Zertifikat von der übergeordneten Zertifizierungsstelle widerrufen, und die UNI-CA generiert sich einen neuen Signierschlüssel und läßt diesen zertifizieren. Mit dem alten Key geleistete Unterschriften können dann trotzdem weiterhin verifiziert werden, solange nur der öffentliche Schlüssel wenigstens noch in einer Kopie vorliegt (z.B. in einem Verzeichnisdienst). Und wenn der Signierschlüssel durch Diebstahl o.ä. einem Angreifer – günstigenfalls verschlüsselt, aber man weiß ja nicht, ob es dem Angreifer nicht doch gelingt, diesen letzten Schutz auch noch zu brechen – in die Hände fällt, so hilft auch eine Sicherungskopie nicht gegen die Kompromittierung des Schlüssels.

4.12 Arbeitsorganisation

Bei der Auswahl der mit dem Betrieb der CA betrauten Mitarbeiter sollten einige wichtige Punkte beachtet werden, denn

> „Das Vertrauen in die neu entstehenden ... Trust Center kann sich bisher kaum aus Erfahrungen speisen, sondern wird eher auf einem Vertrauensvorschuß bzw. bei denjenigen, die aus bekannten Institutionen hervorgegangen sind, auf einer Vertrauensübertragung beruhen.“ [BBFK98, S. 21]

Postulieren wir, daß das UNI-Rechenzentrum bei den Angehörigen der Universität einen guten Ruf hat, ihm und seinen Mitarbeitern aufgrund der von ihnen geleisteten Arbeit Vertrauen entgegengebracht wird, so muß es das Ziel sein, dieses Vertrauen, diesen Vertrauensvorschuß auch auf die neue Institution UNI-CA zu übertragen oder günstige Voraussetzungen dafür zu schaffen, daß die Rechenzentrumsnutzer dies tun.

> „Vertrauen in Institutionen umfaßt sowohl Systemvertrauen als auch Personenvertrauen in Vertreter einer Institution, mit denen Erfahrungen gemacht wurden. Solche ‘*Zugangspunkte*’ ... sind wichtig für die Vertrauensbildung in Institutionen.“ [BBFK98, S. 16]

Es spielt also durchaus eine Rolle, *wer* in der Zertifizierungsstelle arbeitet. Längerfristig dürfte auch eine gewisse Kontinuität im Personal der CA wichtig sein, die nach außen hin sichtbar werden muß. Daher empfiehlt es sich, auch mindestens einen *festangestellten* Rechenzentrumsmitarbeiter (Dauerstelle) für die CA-Arbeit abzustellen oder einzuteilen, um zu vermeiden, daß die CA im halbjährlichen oder jährlichen Wechsel „nur“ von studentischen Mitarbeitern oder kurzzeitig beschäftigten RZ-Mitarbeitern betrieben wird. Ein zu häufiger Wechsel im CA-Personal würde aufgrund der unvermeidlichen Einarbeitungszeit auch jedesmal eine Unterbrechung darstellen und zu Verzögerungen führen.

Neben einem „festen“ CA-Mitarbeiter können selbstverständlich auch studentische Hilfskräfte oder Mitarbeiter mit Zeitverträgen beteiligt werden, es sollte aber eben nicht vorkommen, daß die Stellen solcher Leute auslaufen, sie nicht weiterbeschäftigt werden können und die Zertifizierungsstelle womöglich plötzlich personell völlig von vorne anfangen muß.

Weiterhin sind zwei Punkte von Bedeutung für eine erfolgreiche Arbeit der CA: die Motivation sowie die Sorgfalt und Zuverlässigkeit der CA-Administratoren. Insofern sollten nach Möglichkeit Personen diese Arbeit übernehmen (dürfen) – das wäre der Idealfall –, die schon vorher Interesse an Sicherheits-, Datenschutz oder Vertraulichkeitsaspekten hatten und/oder die für ihre besonders umsichtige und gewissenhafte Arbeitsweise bekannt sind.

4.12.1 Mehr-Augen-Prinzip

Viele Angriffe erfolgen durch „Insider“ [CZ98d, CZ98f, CZ98m], also durch Mitarbeiter oder sonstige Angehörige der geschädigten Institution. Daher sollten vorsorglich auch gegen Mißbrauch oder Schädigungen durch CA-Mitarbeiter Vorkehrungen getroffen werden.

Hierzu zählt zum Beispiel, den Zugriff auf wichtige Ressourcen so zu regulieren, daß er nicht von einem Mitarbeiter alleine ausgeübt werden kann. Für den Signierschlüssel der Medium-Level UNI-CA ist ein solches Vorgehen vorgesehen; der Schlüssel soll bei der Generierung durch eine Aufteilung des Paßwortes zum Zugriff auf den Schlüssel so gesichert werden, daß nur die zwei betreffenden CA-Mitarbeiter *zusammen* ihn für eine Operation freigeben können. Das würde allerdings den eigentlichen Zertifizierungsvorgang um einiges aufwendiger machen, weil dafür dann immer beide Mitarbeiter zugleich zugegen sein müßten.

Diese Maßnahme ist eine Möglichkeit, einen gewissen Schutz vor Insider-Mißbrauch zu erzielen. Sie ist nicht absolut wirksam, da sie umgangen werden könnte, indem einer der betreffenden CA-Mitarbeiter dem anderen seinen Teil des Doppel-Paßwortes verrät (was er eigentlich nicht dürfte, aber es läßt sich letztlich nicht verhindern).

4.12.2 Zurechenbarkeit

Eine andere Stelle, an der durch Insider-Manipulationen falsche Zertifikate bewirkt werden könnten, ist die Identitätsprüfung bzw. die Entgegennahme des Zertifizierungsantrages. Wenn hier einer der CA-Mitarbeiter falsche Angaben in einen Zertifizierungsantrag einträgt und diesen unter die übrigen Anträge mischt, so könnte er damit erreichen, daß ein Zertifikat ausgestellt wird, das die gefälschten Angaben bestätigt. Wirksam vermeiden ließe sich dieser Angriff nur, indem auch bei der Antragsentgegennahme mehrere CA-Mitarbeiter beteiligt würden.

Unter Umständen reicht aber auch schon die Aussicht, daß der Verantwortliche für ein erschlichenes Zertifikat nachträglich festgestellt werden und zur Rechenschaft gezogen werden kann, um potentielle Insider-Täter von ihrem Vorhaben abzubringen. Dies ist einer der Gründe dafür, warum in der Signaturverordnung vorgesehen ist, daß vom Personaldokument des Antragstellers eine Ablichtung anzufertigen und zu den Antragsunterlagen zu nehmen ist. Eine Fälschung könnte, wenn sie dann später auffällt, an dieser Stelle nicht nur festgestellt, sondern auch der Verantwortliche (Fehler oder Betrug durch den CA-Mitarbeiter oder Fälschung des Personaldokumentes durch einen externen Angreifer) festgestellt werden. Dies setzt voraus, daß bei jedem Zertifizierungsantrag vermerkt wird, wer die Identitätsprüfung vorgenommen hat.

Falls die UNI-CA also besonders stark abgesichert betrieben werden soll, wäre es zu überlegen, ob nicht auch dort eine Kopie vom Ausweisdokument jedes Antragstellers angefertigt werden sollte. In jedem Fall muß der CA-Mitarbeiter, der eine Identitätsprüfung vorgenommen hat, den betreffenden Antrag namentlich mit einem Bearbeitungsvermerk kennzeichnen, damit sich, falls Unregelmäßigkeiten festgestellt werden, zurückverfolgen läßt, wer den Antrag entgegengenommen (und bei der Identitätskontrolle möglicherweise fahrlässig oder vorsätzlich falsche Angaben aufgenommen) hat.

4.12.3 Vertretungsregelung

Wenn der Zugriff auf den Signierschlüssel der Zertifizierungsstelle nur zwei Mitarbeitern gemeinsam möglich ist und dafür in jedem Fall die Kenntnis eines Paßwortes erforderlich ist, kommt einer umsichtigen und rechtzeitigen Planung der Urlaubs- und eventueller Krankheitsvertretung eine besondere Bedeutung zu. Wenn nur ein Mitarbeiter ein wichtiges Paßwort kennt und gerade nicht im Hause ist, dann wird dadurch u.U. die CA-Arbeit in starkem Maße beeinträchtigt, mindestens verzögert. Auf der anderen Seite sollen aber auch nur so wenig wie möglich Mitarbeiter Zugriff auf diesen besonders sensiblen Key haben, um die Zahl potentieller Mißbraucher möglichst klein zu halten.

Denkbar wäre hier als eine Lösungsmöglichkeit, daß beide CA-Administratoren mit Schlüsselzugriff ihren Teil der Passphrase jeweils einem anderen Rechenzentrumsmitarbeiter ihres Vertrauens weitersagen. So wären in jedem Fall Regelverletzungen von zwei Personen erforderlich, um Zugriff auf den geheimen Signierschlüssel zu bekommen. Da ein Insider alleine nicht das zweiteilige Paßwort ändern könnte, bestünde auch nicht die Gefahr, daß einer der CA-Mitarbeiter alleine den geheimen Schlüssel durch Änderung der Passphrase „blockieren“ und für alle anderen CA-Mitarbeiter unzugänglich machen könnte.

4.12.4 Ausscheiden eines Mitarbeiters

Das Ausscheiden eines Mitarbeiters (und ggf. die Einstellung einer Ersatzkraft) zieht immer einige typische Aktivitäten nach sich [Lic97]. Ist der Betreffende einer der CA-Mitarbeiter mit Paßwortkenntnis, so müssen neben der obligatorischen Rückgabe von Raum- und Schrankschlüsseln auch Paßworte dokumentiert oder weitergegeben und anschließend aus Sicherheitsgründen – der Ausscheidende ist ja in Kürze ein potentieller externer Angreifer – geändert werden.

Weiterhin sollte, so noch die Gelegenheit dazu besteht, der Betreffende seinen Nachfolger einweisen oder sogar noch einarbeiten. Ist der ausscheidende Mitarbeiter von der CA als Registrierungsinstanz benannt worden, so muß bei seinem Weggang die Liste der RAs aktualisiert werden; war derjenige der Ansprechpartner der UNI-CA gegenüber der DFN-PCA, so sollte diese unverzüglich, nach Möglichkeit noch durch den bisherigen Ansprechpartner, über die sich ergebende personelle Veränderung informiert werden.

4.13 Qualitätssicherung (der Sub-CAs)

> „Das muß nicht bedeuten, daß Verstöße ... das Vertrauen in die Institution sofort zerstören. Voraussetzung ist allerdings, daß sie als Einzelfälle auftreten und sanktioniert werden. Ein gewisses Maß an institutionellem Mißtrauen in Form von Kontrollen kann das Vertrauen in eine Institution sogar stärken ...“ [BBFK98, S. 16]

Wenn die UNI-CA sich dazu entschließt, Registrierungsinstanzen oder sogar nachgeordnete Zertifizierungsstellen einzurichten, dann muß sie auch sicherstellen, daß diese die Policy wie bei der Ernennung schriftlich garantiert einhalten. Derartige Kontrollen sind denkbar z.B. in Form von Stichproben-Prüfungen bei der betreffenden CA/RA mittels unangekündigter oder sogar verdeckter Kontrollen (Test-Zertifizierungen) oder in Form von *agents provocateurs*, die die CA/RA zu Fehlern zu provozieren versuchen. Entscheidend dafür, daß die Regeln durch die Sub-CAs/RAs auch eingehalten werden, sind die Einsicht in deren Sinn und in deren Notwendigkeit – beides sollte also ausreichend vermittelt werden – und für den Fall eines Verstoßes auch eine Auswahl an abgestuften Sanktionsmitteln bis hin zum Entzug der Zertifizierung oder der Registrierungsbefugnis.

4.14 Dokumentation

Einer guten Dokumentation kommt ebenso wie der Öffentlichkeitsarbeit der CA (siehe 4.16) eine wesentliche Bedeutung zu:

> „Neben möglichst niedrigen Eingangshürden ist die Durchschaubarkeit (Transparenz) des Systems ein zentrales Element der Bürgerfreundlichkeit. Dieser Aspekt steht auch in Verbindung mit der Sicherungsinfrastruktur. Denn die Sicherheit der Kommunikationsbeziehungen ist wesentlich davon abhängig, daß der Umgang mit digitalen Signaturen verständlich und die Bedeutung und Zusicherung von Zertifikaten ... ausreichend nachvollziehbar ist.“ [Drä96]

Die Dokumentation ist sowohl nach außen hin für die Nutzer als auch zu internen Zwecken der CA (z.B. für den Fall eines Mitarbeiterwechsels) wichtig. Schon die Policy zwingt dazu, gewisse Angaben oder Informationen zu dokumentieren und teilweise zugänglich zu machen.

4.14.1 Liste nachgeordneter Zertifizierungsstellen

Wenn Sub-CAs eingerichtet werden, sollte eine stets auf dem aktuellen Stand befindliche Übersicht über alle Sub-CAs der betreffenden Zertifizierungsstelle geführt werden. Intern braucht die UNI-CA diese Daten sowieso, und ihre Zusammenstellung erleichtert es Nutzern, einen Überblick über den Teil der Zertifizierungsinfrastruktur zu behalten oder zu erlangen, an dessen Spitze die UNI-CA steht.

4.14.2 Liste der Registrierungsstellen

Die Medium-Level-Policy der DFN-PCA verlangt ausdrücklich, daß halbjährlich eine Liste mit allen von der herausgebenden CA benannten Registrierungsstellen bzw. den mit dieser Funktion betrauten Personen publiziert wird. Dies dient dem Schutz Zertifizierungswilliger, die sich durch einen Blick auf diese Liste (bei großer Skepsis auch durch direkte Nachfrage bei der CA selbst) darüber informieren können, ob eine bestimmte Person tatsächlich befugt ist, als RA zu agieren.

4.14.3 Installations- und Bedienungsanleitungen

Dokumentiert werden sollte, welche Mindest-Schlüssellängen empfohlen bzw. in der Policy vorgeschrieben werden, welche der verfügbaren Schlüsselformate bzw. Verschlüsselungsalgorithmen unterstützt werden und wie in einem typischen Benutzerbereich PGP installiert oder das für alle Nutzer netzweit installierte PGP für die eigene Nutzung konfiguriert werden kann. Hier sind ggf. auch Empfehlungen angebracht, wie Schlüssel im von der CA unterstützten Format erzeugt werden können oder welche Schritte notwendig sind, um Schlüssel in diesem Format mit anderen (neueren) PGP-Versionen nutzen zu können.

Hierzu gehören Hinweise zum einen für die Nutzer der Rechner- und Terminal-Arbeitsplätze im Rechenzentrum der Universität – und, soweit die geleistet werden kann, auch Informationen für die Nutzer in den Fachbereichen –, zum anderen Hinweise für die Universitätsangehörigen, die die Möglichkeit zur Einwahl (*dial-up*) nutzen. Wichtig wären in diesem Zusammenhang auch Hinweise, wie Nutzer, die von beiden Zugangsmöglichkeiten Gebrauch machen, ihren oder ihre PGP-Schlüssel geschickterweise handhaben sollten. Die Adresse, unter der sie E-Mail empfangen können, ist schließlich bei beiden Zugangsmöglichkeiten identisch, im einen Fall dürfte es sich aber bei dem Rechner, an dem sie arbeiten, typischerweise um einen PC unter eigener administrativer Kontrolle handeln (Dialup), im anderen vermutlich eher um eine fremd-administrierte Workstation (Campus-Netz).

Denkbar wäre in diesen Fällen, entweder zwei Schlüsselpaare zu verwenden, von denen das eine – für besonders sensible Kommunikation – nur auf dem heimischen PC benutzt wird, während das andere auf den Rechnern im Campus-Netz verwendet wird, oder aber verschlüsselte Mails grundsätzlich zu Hause zu lesen. Bei der ersten Variante ergibt sich die Schwierigkeit, daß es Sache des Absenders einer vertraulichen Nachricht ist zu entscheiden, mit welchem Empfängerschlüssel die Nachricht gegebenenfalls verschlüsselt wird; es kann also vorkommen, daß ein Kommunikationspartner den „falschen“ Schlüssel benutzt, indem er etwa eine Mail schickt, die vom Campus-Netz aus gelesen werden soll, diese aber mit dem Schlüssel verschlüsselt, dessen geheimes Pendant nur auf dem privaten Rechner des Empfängers eingesetzt wird. Dies führt dazu, daß der Empfänger die Nachricht im Uni-Rechnernetz nicht entschlüsseln kann, so daß es zu Verzögerungen in der Kommunikation kommt oder der Absender die Nachricht erneut und diesmal mit dem zweiten Schlüssel des Empfängers verschlüsseln muß (also u.U. ein erheblicher Aufwand für die Kommunikationspartner). Die zweite Variante, nur auf dem Rechner zu entschlüsseln, der sich unter *eigener* Kontrolle befindet, ist unter Sicherheitsaspekten vorteilhaft, führt aber wieder dazu, daß bei der „normalen“ E-Mail-Kommunikation seltener oder gar nicht verschlüsselt wird, da in diesem Falle der geheime Schlüssel nicht im Uni-Rechnernetz zur Verfügung steht.

4.14.4 Lokalisierung (Sprachanpassung)

Da gerade an einer Universität nicht alle Computernutzer deutsche Muttersprachler sind (und selbst unter denen manche die englischsprachige Variante eines Programms bevorzugen mögen), sollten von Seiten der CA bzw. der Netzwerk-Administratoren alle erforderlichen Vorbereitungen getroffen werden, damit die Nutzer PGP auch mit anderen Sprach-Einstellungen verwenden können. Mindestens sollte dokumentiert sein, wie bei Bedarf Ausgaben in einer anderen Sprache erzeugt werden können und welche Installations- oder Konfigurationsschritte dafür erforderlich sind. Dies erscheint umso wichtiger, als gerade bei Sicherheitssoftware Fehlbedienungen aufgrund von Programm-Ausgaben in einer für den Nutzer schwer verständlichen Sprache gravierende Folgen haben können. (Umgekehrt bedeutet dies aber auch, daß – sofern verfügbar – auch Versionen der Verschlüsselungssoftware mit deutschsprachiger Bedienoberfläche bereitgehalten werden sollten, da viele Benutzer sich damit besser zurechtfinden werden als mit einer englischsprachigen Version!)

4.14.5 Leitfäden für die Zertifizierung

Zur Dokumentation für die Nutzer sollte die CA auch Hinweise parat halten, welche Schritte ein Nutzer unternehmen muß, wenn er eine Schlüsselzertifizierung wünscht. D.h. die Abläufe und Interaktionen Nutzer – CA sollten beschränkt auf die Schritte dargestellt werden, bei denen der Nutzer direkt involviert ist.

4.14.6 Hinweise zum Schutz des Private-Keys

Der Geheimhaltung des – oder der – eigenen Private-Keys kommt eine große Bedeutung zu, denn mit ihr steht oder fällt letztlich die Vertraulichkeit bzw. die Zurechenbarkeit einer Nachricht. Deswegen sollte die CA auch Empfehlungen dazu parat haben, wie jeder Nutzer seinen geheimen Schlüssel wirkungsvoll schützen kann – sowohl auf einem PC, der erfahrungsgemäß viele Sicherheitslücken und damit Angriffsmöglichkeiten aufweisen kann, als auch bei der Verwendung des Schlüssels innerhalb eines Rechnernetzes, in dem andere Personen Superuser-Rechte haben. Hierbei sind beispielsweise Zusammenstellungen wie die von RALF SENDEREK [Sen98] hilfreich. Diese Informationen könnten sinnvollerweise auch mit einem Teil des Zertifizierungsantrages dem Benutzer ausgehändigt werden und bei ihm verbleiben. Auf diesem Teil könnten dann zusätzlich die jeweils aktuellen Schlüsselinformationen zu den Signierschlüsseln der CA sowie ihre Kontaktangaben (Anschrift, Telefonnummer, Mailadressen) angeführt werden, so daß der Zertifikatnehmer sie leicht zur Hand hat, falls er mit der CA in Kontakt treten möchte.

4.14.7 Key-Signing-Parties

Um die Nutzung von Verschlüsselung zu stärken und zugleich die Anwender von Verschlüsselungssoftware in unterhaltsame und öffentlichkeitswirksame Aktivitäten einzubeziehen, bieten sich Key-Signing-Parties an, bei denen die Teilnehmenden untereinander ihre Public-Keys bzw. deren Fingerprint austauschen, ohne daß eine Zertifizierungsstelle involviert sein muß. (Unter Umständen kommt auch von Nutzerseite die Frage, wie so etwas organisiert werden kann.) Daher sollte auch eine Anleitung Teil der Dokumentation sein, wie man ein solches Key-Signing organisieren und durchführen kann. (Eine kurze Beschreibung, wie ein Key-Signing organisiert werden kann, findet sich z.B. als Punkt 6.8 in der Comp.security.pgp-FAQ[10].)

4.15 Software-Pflege (FTP-Server)

Eine Aufgabe, der sich die UNI-CA ebenfalls annehmen sollte, ist die Pflege eines FTP-Servers mit Verschlüsselungssoftware für möglichst alle der an der Universität gängigen Rechnertypen und Betriebssysteme. Mindestens die vom Rechenzentrum selbst angebotenen oder unterstützten Plattformen sollten dabei abgedeckt werden.

Der Zertifizierungsstelle wird aufgrund ihres größeren Know-hows auf diesem Gebiet ziemlich automatisch auch die Aufgabe zukommen, existierende Software mit (Public-Key-)Verschlüsselung auf ihre kryptographische Sicherheit hin zu bewerten oder zumindest die Einschätzungen von Fachleuten auf diesem Gebiet für ihre Nutzer zu dokumentieren (und im Extremfall einer gravierenden Sicherheitslücke diese auch darüber zu informieren). Dies betrifft insbesondere die unterschiedlichen Software-Versionen, die sich aus den US-Exportbestimmungen ergeben, die die Ausfuhr von starker Verschlüsselung außer in Sonderfällen bisher nicht zuließen (vgl. 4.19.6).

Die UNI-CA sollte dazu Empfehlungen geben, welche von möglicherweise verschiedenen verfügbaren Versionen eines Programms unter Sicherheitsaspekten bevorzugt benutzt werden sollte. Zusätzlich könnte die Zertifizierungsstelle auch Patches bereitstellen und dokumentieren, die die starke Verschlüsselung in „Exportversionen“ von US-Software wieder einschalten.[11]

Im Zuge der Software-Bereitstellung durch die UNI-CA könnte es sich bei stärkerer Nachfrage nach kommerziellen Verschlüsselungsprogrammen oder kommerzieller Anwendungssoftware mit Verschlüsselung als sinnvoll erweisen, Bestellungen oder den Verkauf solcher Software an Universitätsangehörige abzuwickeln, wie es die Benutzerbetreuung oder die Rechenzentren mancher Hochschulen bereits mit allgemeiner Software tun (siehe [FUB99]), oder sogar Campus-Lizenzen für die UNI zu beschaffen, falls die Nachfrage nach einem bestimmten Produkt sehr groß ist oder so erheblich günstigere Konditionen möglich werden.

Die Software, die die CA auf ihrem FTP-Server bereitstellen sollte, läßt sich in die drei Kategorien

- Anwendungssoftware
- Server-Software und
- CA-Software

unterteilen, wobei eine eindeutige Zuordnung eines Programms in nur eine dieser Kategorien vermutlich nicht immer möglich sein wird.

[10] `http://www.iks-jena.de/mitarb/lutz/security/pgpfaq.html`

[11] z.B. Farrell McKays „Fortify“ für den Netscape Navigator, `http://www.fortify.net`

4.15.1 Anwendungssoftware

Am stärksten wird Anwendungssoftware wie PGP selber, Web-Browser mit Verschlüsselungsfunktionalität oder Mail-Useragents mit integrierter Public-Key-Verschlüsselung nachgefragt werden, aber auch z.B. Newsreader mit PGP-Unterstützung oder Public-Key-verschlüsselnde Werkzeuge wie *ssh*.

Bei den Web-Browsern sollte den Nutzern deutlich gemacht werden, wie stark oder schwach die Verschlüsselung ist, die von den unterschiedlichen Produkten bzw. deren verschiedenen Export-, Freeware- oder internationalen Versionen unterstützt wird. Hier sind insbesondere die durch die Offenlegung des Netscape Navigator-Quellcodes ermöglichte Open-Source-Version dieses Programms, *Cryptozilla*[12], mit starker Verschlüsselung und der nicht von den US-Exportbestimmungen betroffene Browser *Opera* zu nennen, der ab Version 3.5 ebenfalls starke 128-Bit-Verschlüsselung unterstützt [CZ98i] und demnächst auch für Linux, Solaris und MacOS erhältlich sein soll [CZ98j]. Da inzwischen die US-Exportbestimmungen gelockert wurden, sind nun auch Browser-Versionen mit starker Verschlüsselung von Netscape[13] und von Microsoft[14] international erhältlich.

4.15.2 Server-Software

Für die Absicherung von WWW-Servern durch starke Verschlüsselung werden bestimmte Versionen der jeweiligen Server-Software (oder Patches zu diesen Programmen) benötigt, die die CA ebenfalls zusammenstellen und bereithalten sollte. Administratoren in den einzelnen Fachbereichen werden sich von der Zertifizierungsstelle, die letztlich von vielen Nutzern als eine Art *Competence Center* für alle Verschlüsselungsbelange angesehen werden wird, Hilfe bei der Auswahl geeigneter Server-Software oder bei deren Installation und sicherheitstechnischer Konfiguration erhoffen. Letztlich sollen für solche Server dann ja auch von der UNI-CA die Zertifikate ausgestellt werden.

4.15.3 CA-Software (für Sub-CAs)

Spätestens dann, wenn die UNI-CA eigene nachgeordnete Zertifizierungsstellen zuläßt, wird es auch erforderlich, diesen geeignete CA-Software und vielleicht auch Hinweise zu deren Einsatz unter den spezifischen Bedingungen der UNI zur Verfügung zu stellen. Daher sollte auch ein entsprechender Teil mit CA-Software, -Tools usw. auf dem FTP-Server der UNI-CA eingeplant werden.

4.16 Außendarstellung der CA

> „Wir haben weltweit sechs Millionen User der Verschlüsselungssoftware PGP, aber unsere Antivirussoftware wird schon von 60 Millionen benutzt.“ Bill Larson in [Moe99]

Wie das Zitat von NAI-Chef BILL LARSON belegt, existiert ein großes Potential an durchaus sicherheitsbewußten oder für Sicherheitsgefahren sensibilisierten Computernutzern, das aber bislang

[12] `http://www.cryptozilla.org`
[13] `http://www.netscape.com/newsref/pr/newsrelease794.html`
[14] `http://www.microsoft.com/presspass/press/2000/Jan00/encryptionPR.asp`

hinsichtlich des Einsatzes von Verschlüsselungstechnik noch kaum erschlossen ist. Daher muß es auch die Aufgabe einer Zertifizierungsstelle sein, das Bewußtsein der Anwender für Sicherheitsrisiken bei der elektronischen Kommunikation oder der Internet-Nutzung zu wecken und zu schärfen. Dies kann auf eher humorvolle Weise geschehen, durch Texte wie den von ANDREAS PFITZMANN zum „Schlüsselhinterlegungs- und Aufbewahrungsgesetz“[15], aber auch durch die Vorbildwirkung der Zertifizierungsstelle und anderer Einrichtungen oder Personen.

Aus diesem Grund sollte die Zertifizierungsstelle und, wenn irgend möglich, das ganze Rechenzentrum die Sicherheitsverfahren, deren Einsatz sie propagieren, auch selber *sichtbar* anwenden. (Das ist auch eine Sache der Glaubwürdigkeit.) Ein erster Schritt in dieser Richtung könnte beispielsweise darin bestehen, die wichtigsten WWW-Server des Rechenzentrums bzw. der UNI HTTPS-fähig zu machen, so daß auch verschlüsselte Verbindungen zu ihnen möglich sind. (Eine Anleitung, wie dies mit dem weit verbreiteten HTTP-Server *apache* durchgeführt werden kann, ist in Teil III dieses Handbuches wiedergegeben.) Auf diese Neuerung müßte natürlich auch angemessen hingewiesen werden (durch News-Postings in geeignete UNI-interne Newsgruppen, durch Artikel in der UNI-Hauszeitung oder durch Hinweise auf den Einstiegsseiten der Server), damit möglichst viele Nutzer davon erfahren.

Eine weitere Möglichkeit bestünde darin, daß die RZ-Mitarbeiter sich PGP-Schlüssel erzeugen und in ihren Mail-*Signatures* darauf hinweisen, daß jetzt auch eine verschlüsselte E-Mail-Kommunikation mit ihnen möglich ist oder daß sie ihre News-Postings mit PGP digital signieren. (Bei Klartext-Signaturen, wie sie PGP unterstützt, bleibt der eigentliche Text weiterhin lesbar, auch für alle Anwender, die PGP nicht nutzen, insofern wäre dies problemlos möglich und würde niemanden daran hindern, die Postings lesen zu können.)

Positiv zu vermerken ist, daß die Sensibilität für das Thema Verschlüsselung bzw. Vertraulichkeit bei elektronischer Kommunikation wächst, wie das folgende Zitat aus dem *ötv-magazin* belegt:

> „Ein Einfallstor für Datenspione aller Art ist die E-Mail. Einmal versendet, kann sie von Systemverwaltern, Mitarbeitern des Providers oder den Datenfahndern der internationalen Geheimdienste jederzeit mitgelesen werden. Im neuen Datenschutzgesetz sind das Archivieren der E-Mail-Kommunikation durch die Provider und ein Zugriff der Geheimdienste vorgesehen. Schützen läßt sich die E-Mail-Kommunikation nur durch konsequente Verschlüsselung. Von den vielen Verschlüsselungstechniken hat sich bis vor kurzem »Pretty Good Privacy« (http://www.pgp.com) als besonders wirksam erwiesen.
> (Aus »journalist Medienpraxis« 8/1998)“ [ötv99]

und auch die „Verbraucherschützer raten zur Verschlüsselung“, wie der Berliner *TAGESSPIEGEL* zu berichten weiß [Tsp99b].

Um in der Anlaufphase gezielt Nutzer anzusprechen, die bereits PGP zur Verschlüsselung einsetzen und die daher vermutlich am leichtesten für die Nutzung der Dienste einer Zertifizierungsstelle gewonnen werden können, wäre eine gezielte Mail mit dem Hinweis auf das neue Angebot an alle PGP-Nutzer der UNI denkbar. Dazu könnten von den PGP-Keyservern alle Schlüssel abgefragt werden, die die Zeichenkette ‘UNI’ oder ‘UNI.de’ in einer ihrer Benutzerkennungen enthalten. Die entsprechenden Adressen könnte man anmailen und die betreffenden Personen so auf das neue

[15] `http://www.zerberus.de/texte/ccc/ccc95/div/pfitz/satire.htm`

Angebot hinweisen. (Allerdings gilt es hier sorgsam abzuwägen, ob diese Aktion nicht „nach hinten losgehen“ könnte, und zwar dann, wenn viele der angeschriebenen die Mail eher als *spam*, also als belästigende, unverlangt zugesandte Nachricht, auffassen!)

Eine andere Aktionsform, mit der die Zertifizierungsstelle den Zusammenhalt der UNI-PGP-Nutzer untereinander fördern und sie miteinander persönlicher bekanntmachen könnte, wären Key-Signing-Parties (siehe 4.14.7) im Rahmen anderer geeigneter Anlässe. Damit eine solche Key-Signing-Party ein Erfolg wird, sind einige organisatorische Vorbereitungen zu treffen; würde die UNI-CA diese Aufgabe übernehmen, so würde sie durch gelungene Key-Signing-Parties die Nutzer sicher auch zur Nachahmung anregen.

Weitergehende öffentlichkeitswirksame Aktivitäten der Zertifizierungsstelle könnten z.B. in Kooperationen mit dem Datenschutzbeauftragten der UNI oder dem des betreffenden Bundeslandes oder mit Verbraucherverbänden (s.o.) bestehen. (Siehe dazu auch Abschnitt 4.18.)

Nachstehend werden nun einige spezielle Gesichtspunkte, die Teile der Außendarstellung oder der Öffentlichkeitsarbeit einer Zertifizierungsstelle betreffen, näher erörtert.

4.16.1 Benennung: ‘CA’ vs. ‘Trustcenter’

Der Begriff ‘Trustcenter’ wird unterschiedlich gebraucht: Manche Autoren verwenden ihn pauschal für alle Vertrauensinstanzen, andere verknüpfen den Begriff mit einer bestimmten baulich-organisatorischen Ausprägung:

> „Ist die Vertrauensinstanz räumlich und organisatorisch verselbständigt, quasi als eigene Trutzburg hinter Beton, Außenhautsicherungen und Zutrittskontrollen abgesichert, wird sie als *Trustcenter* bezeichnet.“ [BHK+94, S. 41]

Unter anderem in Dokumenten der Europäischen Union bedeutet der Begriff ‘Trustcenter’ oder auch ‘Trusted Third Party’ (TTP), also ein vertrauenswürdiger Dritter, fast immer zugleich auch ‘eine Stelle zur Hinterlegung geheimer Schlüssel’ (Stichwort *key escrow* bzw. *key recovery*), die gegebenenfalls den Sicherheitsbehörden eines Landes den Zugriff auf die Schlüssel ermöglicht. Insofern ist, zumindest für „Eingeweihte“, der Begriff ‘Trustcenter’ nicht mehr so positiv oder neutral, wie er für einen neutralen Beobachter zunächst erscheinen mag. Die EU-Kommission hält dies selber fest, indem sie in [EUK97, 2., dt. Fassung] bemerkt:

> „TTPs ... werden jedoch häufig mit dem rechtmäßigen Zugriff auf Schlüssel in Verbindung gebracht [...]. Es ist nicht ausgeschlossen, daß TTPs auch als CA, wie in dieser Mitteilung beschrieben, fungieren können. Die Aufgaben beider Einrichtungen werden jedoch als verschieden angesehen.“

An anderer Stelle im selben Dokument wird die Kommission sogar noch deutlicher, wenn sie im Zusammenhang mit dem Schutz vor „Identitätsdiebstahl“ feststellt:

> „CAs muß es deshalb verboten sein, private Schlüssel aufzubewahren. Das wiederum unterscheidet CAs von TTPs, welche gerade das Aufbewahren von Informationen über private Schlüssel zu ihren Aufgaben zählen.“

Das alleine wäre schon Grund genug, mit Blick auf die EU-Richtlinie zu elektronischen Signaturen (vgl. 3.5.3) auf derartige Angebote an die Nutzer zu verzichten. Dieser Verzicht sollte auch aus einem anderen Grund umso leichter fallen: Die sichere Erzeugung bzw. Speicherung oder Archivierung geheimer Nutzerschlüssel ist mit erheblichem Aufwand für die betreffende Stelle verbunden – auch aus diesem Grund sollte die UNI-CA bis auf weiteres keine derartigen Trustcenter-Dienste anbieten.[16]

4.16.2 Vertrauen

Das Vertrauen ist eine zarte Pflanze;
ist es zerstört, so kommt es sobald nicht wieder.
— OTTO von BISMARCK, nach [Ull60, S. 349]

Vertrauen kann nicht „erzeugt“, wohl aber können die Rahmenbedingungen, die sein Entstehen begünstigen, geschaffen werden. Vertrauen kann beschrieben werden als riskante Vorleistung bzw. ein mittlerer Zustand im Spannungsfeld zwischen Wissen und Nichtwissen.

> „Der Wissende braucht nicht zu vertrauen. [...] Das Fehlen vollständiger Informationen ist also eine Hauptbedingung dafür, daß Vertrauen erforderlich ist. Eine Position, die Vertrauen vor allem unter dem Aspekt der Vertrauenswürdigkeit betrachtet und ... davon ausgeht, daß nur vertraut wird, wenn etwas sicher ist, verfehlt diesen wichtigen Punkt ...“ [BBFK98, S. 3–4].

Vertrauen ist „spröde“ wie ein Kristall, sagen Sozialwissenschaftler. Es wächst nur langsam, und es kann durch eine heftige Einwirkung von außen vollständig zerstört werden – umso mehr muß einer Zertifizierungsstelle, die ihre Existenz ja letztlich in dem Vertrauen begründet, das ihr die Zertifikatnutzer entgegenbringen, daran gelegen sein, dieses Vertrauen zu bewahren und Erschütterungen zu vermeiden.

Die UNI-CA wäre allerdings nicht nur auf solche „Werbe-“ oder „vertrauensbildenden Maßnahmen“ angewiesen, sondern könnte selber auch von ihrer anzunehmenden personellen und räumlich-organisatorischen Verzahnung mit dem UNI-Rechenzentrum und von dessen positivem Image (so vorhanden) bei den meisten Internet-Nutzern der UNI profitieren:

> Wenn die Namen von Herstellern und Betreibern bereits bekannt sind und bislang für solide Produkte und Dienstleistungen standen, wird man der neuen Technik u.U. im voraus mehr Vertrauen schenken, als wenn auch sie in der Öffentlichkeit großenteils unbekannt sind. ... Auch neue Unternehmen, die aus bekannten hervorgehen oder mit ihnen in anderer Weise in Verbindung gebracht werden können, profitieren von solchen Analogien. [BBFK98, 15] (siehe auch 4.12)

Ein ganz heikler Punkt ist in diesem Zusammenhang auch die Schlüsselerzeugung:

[16] Zu einer Erörterung der angemessenen Bezeichnung für die noch umfassendere *Public-Key-Infrastruktur* siehe HAMMER [Ham98]

> „Die Vertrauenswürdigkeit eines Trust Centers ist viel leichter (bzw. nur) zu erreichen, wenn es keine geheimen Teilnehmerschlüssel kennt. Selbst wenn die Schlüssel nach der Erzeugung wirklich sofort gelöscht werden, läßt sich dies nicht beweisen. Es bleibt stets Raum für Gerüchte und Unterstellungen. Es sollte also im eigenen Interesse eines Trust Centers sein, keine geheimen Schlüssel [für die Teilnehmer] zu erzeugen. Wer eine Sicherungsinfrastruktur aufbaut, muß dem Teilnehmer Optionen geben, seine Schlüssel sicher zu generieren. Wie aber die obigen Bemerkungen gezeigt haben, ist eine alleinige Generierung in einem Trust Center ungeeignet.“ [Fed97][17]

Daher ist auch die Schlüsselgenerierung für Teilnehmer ausschließlich für die Low-Level UNI-CA vorgesehen. Sie soll (und kann) kein hohes Sicherheitsniveau bei der Zertifizierung gewährleisten – dafür ist die Medium-Level-CA vorgesehen –, sondern ihr Hauptzweck besteht darin, möglichst öffentlichkeitswirksam zu agieren; sie muß sich dabei allerdings davor hüten, durch nachlässiges Verhalten die Vertrauenswürdigkeit der UNI-CA insgesamt zu gefährden.

4.16.3 WWW-Präsenz

Natürlich gehört für eine Institution, die sich ausschließlich mit Dienstleistungen im Zusammenhang mit digitaler Kommunikation und dem Internet befaßt, zur Außendarstellung auch eine entsprechende WWW-Präsenz.

Die UNI-CA sollte diesen Weg nutzen, um ihre Arbeit, die Grundlagen dafür und entsprechende Dokumentation für alle Interessierten (auch für *relying parties*) in leicht zugänglicher Form zur Verfügung zu stellen. An anderer Stelle wurde bereits erwähnt, daß im Interesse ihrer Glaubwürdigkeit die Zertifizierungsstelle selbst auch diejenigen Sicherungsmöglichkeiten (verschlüsselte Kommunikation, digital signierte öffentliche Ankündigungen, SSL-gesicherter Zugriff auf ihre Web-Seiten) nutzen sollte, die sie als Dienste anbietet und deren Nutzung sie fördern will.

Bei der Software-Verteilung haben die WWW-Seiten gegenüber dem FTP-Server den Vorzug, daß mit ihrer Hilfe noch besser die Struktur des Software-Angebotes deutlich gemacht und daß Hinweise zu einzelnen Programm-Versionen unmittelbar mit diesen verknüpft dargestellt werden können.

Eine Studie zu Vertrauen in Anbieter von *electronic commerce* im World Wide Web und zu deren Web-Auftritt [CSA99] ergab einige interessante Aspekte, die man beim Web-Auftritt der neuen „Marke“ UNI-CA beherzigen sollte:

- Auch wenn sich Vertrauen erst langsam im Laufe der Zeit einstellt, muß Vertrauenswürdigkeit dem Besucher einer Web-Site vom ersten Kontakt an vermittelt werden.
- Eine neue Firmenmarke (*brand*) im WWW braucht eine exzellente Navigation und Kundenzufriedenheit, wenn ihr vertraut werden soll.

[17] Die entgegengesetzte Position *pro* Schlüsselgenerierung im Trustcenter wird von ROLAND NEHL (Deutsche Telekom AG/Telesec) in [Neh97] vertreten. Nicht sonderlich überzeugend allerdings sein Argument, das Trustcenter hätte gar kein Interesse daran, Kenntnis des geheimen Schlüssels eines Kunden zu erlangen, da es ja selber ein Schlüsselpaar erzeugen und dann den öffentlichen Schlüssel als vorgeblich den des Kunden zertifizieren könnte – dieser Angriff würde viel schneller auffallen als die Kenntnis des geheimen Nutzerschlüssels.

- Der Bekanntheitsgrad einer Web-Site ist wichtig für das Vertrauen, das ihr die Besucher entgegenbringen.
- Web-basierende Sicherheits-Markennamen-Symbole wie VeriSign vermitteln dem Besucher Vertrauenswürdigkeit.

Die Bedeutung einer guten Gestaltung und Benutzerführung beim WWW-Auftritt also gerade für eine Zertifizierungsstelle, für die es ja wie für kaum eine andere Einrichtung auf das Vertrauen ihrer „Kundschaft“ ankommt, kann daher nicht genug betont werden. Zum Glück läßt sich auch mit einfachen Mitteln, mit kostenlosen Tools und vertretbarem Aufwand ein gelungener Web-Auftritt mit einer *Corporate Identity* erzielen [BP99]. Aufgrund des letzten Punktes bietet es sich darüber hinaus an, den Web-Auftritt der UNI-CA mit einigen geeigneten, bekannten *seals of support* zu versehen, beispielsweise jenen für PGP, SSL, OpenSSL oder „pro Verschlüsselung“ allgemein.

4.16.4 Erreichbarkeit der CA als Einrichtung

Für eine Zertifizierungsinstanz und ihre Registrierungsstellen, die ja auf den persönlichen Kontakt mit den Zertifikatnehmern angewiesen sind, ist eine gute Erreichbarkeit durch die Nutzer unabdingbar. Das betrifft sowohl die geographische Lage ihrer Räumlichkeiten, als auch die Öffnungs- oder Bürozeiten, als auch ihre sonstigen Kommunikationsverbindungen. Alle diese Angaben müssen für die (potentiellen) Nutzer leicht zugänglich und mit wenig Aufwand ermittelbar sein.

4.16.4.1 Büro und Sprechzeiten

Einzelne CA-Mitarbeiter oder Raumnummern der Zertifizierungsstelle sollten nicht in der Policy genannt werden, da diese ansonsten geändert werden müßte, bloß weil eventuell das CA-Büro innerhalb des Gebäudes einige Zimmer weiter zieht oder weil ein Mitarbeiter ausscheidet.

Hinsichtlich der Büro-Sprechzeiten der UNI-CA bieten sich zwei Alternativen an: Entweder die Zertifizierungsanträge können in der allgemeinen Benutzerberatung während deren Öffnungszeiten abgegeben werden und werden dort von einem der RZ-Mitarbeiter entgegengenommen, der dort gerade Dienst tut (dann müßten diese Personen über die vorzunehmenden Kontrollen bei der Identitätsprüfung informiert und in die Arbeitsorganisation der CA einbezogen werden), oder aber es wird eine separate „Zertifizierungs-Sprechstunde“ angeboten, während der Zertifizierungsanträge abgegeben und Fragen zur Zertifizierung gestellt werden können. Bei der zweiten Variante wäre der Umfang der wöchentlichen Sprechzeiten vermutlich geringer, dafür wäre sichergestellt, daß einer der CA-Mitarbeiter für die Fragen zur Verfügung steht, außerdem müßten nicht so viele Mitarbeiter wie bei der ersten Variante koordiniert werden.

4.16.4.2 Hotline für Notfälle

Für eine umgehende Verständigung der CA im Falle einer Schlüsselkompromittierung könnte eine „Notrufnummer“ hilfreich sein. Allerdings bestünde bei ihr vermutlich das Problem, daß sich kaum verhindern ließe, daß sie von den Nutzern auch angerufen würde, wenn sie irgendwelche Fragen zur

Zertifizierung oder zu Verschlüsselungssoftware haben und gar keine Dringlichkeit ihrer Anfrage gegeben ist.

Wenn ein solcher Service nicht angeboten wird, sollte in den Policies der UNI-CA deutlich gemacht werden, innerhalb welcher Zeiträume die CA typischerweise erreicht werden kann, wie lang also schlimmstenfalls die Zeitspanne zwischen der Entdeckung einer Schlüsselkompromittierung und der Sperrung des betreffenden Zertifikates ist.

4.16.4.3 E-Mail

Für die Kommunikation mit der UNI-CA per E-Mail muß zunächst einmal die Mailadresse festgelegt werden. Sie wird auch in der Policy genannt und muß nach den Vorgaben der Low-Level DFN-Policy bei PGP-Schlüsseln in der Benutzerkennung des CA-Signierschlüssels enthalten sein (Abschnitt 8.1). Naheliegend und angenehm kurz wäre `ca@UNI.de`, aber es sind auch andere Varianten denkbar (`zertifizierungsstelle@UNI.de`, `uni-ca@rz.UNI.de`, ...) Es dürfte sinnvoll sein, für die UNI-CA als Einrichtung nur eine einzige E-Mailadresse zu verwenden und nicht für jede der Policies eine eigene Adresse, etwa der Art `low-ca@UNI.de` usw., vorzusehen. Falls die Nachfrage nach Zertifizierungsdiensten später einmal sehr groß werden sollte, kann eine Aufteilung vielleicht hilfreich sein, während der Startphase und der ersten Zeit der CA-Tätigkeit dürften sich aber die Anfragen in Grenzen halten – insbesondere, da es zu den Low-Level-Zertifikaten nicht so viele Fragen an die CA geben dürfte, weil diese ja unmittelbar ausgestellt und dem Zertifikatnehmer ausgehändigt werden.

Ein weiterer wichtiger Punkt, der von Anfang an geklärt sein muß, ist die Handhabung der verschiedenen CA-Schlüssel bzw. von Mails „der CA“ nach außen. Zu klären sind die Punkte

- Unter welchem Absender bzw. von welchem Account aus werden die Antworten der CA verschickt?
- Mit welchem/wessen Schlüssel (von wem) werden sie gegebenenfalls (oder immer?) signiert? Mit einem Schlüssel der CA? – Das dürfte dann nicht der Zertifizierungsschlüssel sein. Oder mit dem Schlüssel des antwortenden Mitarbeiters?
- Wird ein einheitliches *Reply-to:* bei den ausgehenden Mails der Zertifizierungsstelle verwendet (empfehlenswert, damit die Kommunikation weiterhin an die Adresse der CA und nicht die eines einzelnen Mitarbeiters gerichtet bleibt), und wenn ja: Welche Adresse wird angegeben?
- Soll von allen ausgehenden CA-Mails eine Kopie oder Blindkopie an eine CA-Mailadresse geschickt werden? (an welche?)

Vorgeschlagen wird eine Vorgehensweise analog zu der bei DFN-CERT und -PCA. Dort gibt es einen „Team-Key“ genannten Schlüssel für die vertrauliche Kommunikation mit dem CERT, über dessen geheime Komponente alle CERT-Mitarbeiter verfügen, und die Mitarbeiter signieren die Nachrichten, die sie im Rahmen ihrer CERT-Arbeit versenden, mit ihrem persönlichen Schlüssel [CER94]. Der Zertifizierungsschlüssel der CA wird ausschließlich für das Signieren von Zertifikaten, Zertifikatwiderrufen, Cross-Zertifikaten, Sperrlisten und Policies verwendet.

Die Mails, die in der Funktion als CA-Mitarbeiter verschickt werden, sollten alle ein ‘Cc:’ sowie ein ‘Reply-to:’ auf die Mailadresse der UNI-CA enthalten. Darüber hinaus sollte die CA nach außen hin auch in ihren E-Mails ein einheitliches Erscheinungsbild wahren, insbesondere was das PGP-Format der Mails angeht. Hier muß eine Entscheidung zwischen zwei Varianten getroffen und dann in der Praxis auf ihre Tauglichkeit erprobt werden, die dann auch von den CA-Mitarbeitern eingehalten werden sollte – auch wenn dies für sie u.U. heißen kann, daß sie für die CA-Tätigkeit ein anderes als ihr individuell bevorzugtes Mail-Programm benutzen müßten (siehe dazu auch Abschnitt 5.2.9).

4.16.5 Verteilung des authentischen CA-Signierschlüssels

Zertifizierungsstellen reduzieren zwar das Problem der Verteilung authentischer Schlüssel (vgl. 2.3), schaffen es aber nicht völlig aus der Welt: Zumindest der Schlüssel, mit dem die Signaturen der jeweiligen Zertifizierungsstelle überprüft werden können, muß noch in authentischer Form zu den Zertifikat-Nutzern transportiert werden.

Üblicherweise bedient man sich dazu nicht-elektronischer Kommunikationswege wie Kuriere oder Drucksachen. So muß nach § 8 (2) SigV die „zuständige Behörde“, die Wurzel-Zertifizierungsinstanz nach dem Signaturgesetz, ihre Schlüssel im Bundesanzeiger veröffentlichen[18], die pgpCA der *c’t* nutzt das Impressum der eigenen Zeitschrift (alle Ausgaben seit Start der Krypto-Kampagne zur CeBIT 1997), die DFN-PCA verfügen mit den *DFN-Mitteilungen* über ein geeignetes Verbreitungsmedium, und die IN-Root-CA ist mit dem Fingerprint ihres Schlüssels beim *Linux Magazin* [LIN] untergekommen.[19]

Zwei Möglichkeiten zur Schlüsselverteilung für die UNI-CA und zugleich eine gute Gelegenheit, sich mittels eines Artikels einem breiteren Publikum bekannt zu machen, wären die regelmäßigen UNI-Publikationen und das gedruckte Vorlesungsverzeichnis der UNI.

Eine andere Gelegenheit, den öffentlichen UNI-CA-Schlüssel zur Prüfung von CA-Signaturen unter den RZ-Nutzern zu verteilen und sich selbst bekannt zu machen, böte eine CD-ROM der Zertifizierungsstelle – oder aber es wird die nächste Gelegenheit genutzt, die Schlüsseldaten der öffentlichen CA-Schlüssel und einige Zusatzinformationen auf der „Dialup“-Software-CD-ROM mit unterzubringen, die das UNI-Rechenzentrum seinen Nutzern anbietet. Hier wäre bei der Produktion des Masters bzw. der Abnahme der fertiggepreßten CD-ROMs vom Hersteller besonders zu kontrollieren, daß auch die authentischen CA-Schlüssel und keine Fälschungen auf den Medien gespeichert sind. Ein Nachteil besteht bei den nicht überschreibbaren CD-ROMs: Ein eventueller Schlüsselwechsel könnte nicht nachvollzogen werden, falls die CD-ROM in einer hohen Auflage gepreßt würde und sie nicht schnell genug verteilt wird. Oder, noch ungünstiger: eine Kompromittierung des UNI-CA-Schlüssels führte, falls sie denn einträte, dazu, daß sich ein solcher kompromittierter, nicht mehr verwendbarer Schlüssel auf den noch vorhandenen CD-ROMs befände. Bei Disketten wäre es hingegen möglich, entsprechende neue Schlüssel auf die Datenträger zu kopieren. Insofern ist die Lösung mit Disketten, die auch der Schleswig-Holsteinische Datenschutzbeauftragte für seinen PGP-Key gewählt hat, eventuell günstiger.

---

[18] so erstmalig im Bundesanzeiger Nr. 186 vom 6. Oktober 1998

[19] Die im März 2000 gültigen Schlüsselinformationen der DFN-PCA sind auch ganz am Ende dieses Handbuches abgedruckt.

Wichtig ist in jedem Fall der eindringliche Hinweis an die Nutzer, sich immer erst zu vergewissern, daß sie den *authentischen* Schlüssel erhalten haben, bevor sie sich auf die damit signierten Zertifikate verlassen. Ein Vorfall von 1997 mag deutlich machen, daß dies gar nicht oft genug betont werden kann:

Im Sommer 1997 waren einige „exponierte" PGP-Schlüssel wie der Zertifizierungsschlüssel der *c't*-CA und der Signierschlüssel der IN-Root-CA Ziel eines unbekannten Angreifers. Der Unbekannte spielte Fälschungen der entsprechenden Schlüssel auf den internationalen PGP-Keyservern ein, vermutlich mit dem Ziel, andere PGP-Anwender zu verwirren und die betreffenden CAs zu diskreditieren:

```
Type Bits/KeyID    Date       User ID
pub  2048/93478F6B 1997/06/17 in-ca@individual.net SIGN EXPIRE:1998-12-31
 Root CA des Individual Network e.V. <in-ca@individual.net>
 Key fingerprint = F9 B7 D9 68 AC 0D 18 18  96 1C 3B BA 78 47 B3 14

pub  2048/D8759C65 1997/03/10 in-ca@individual.net SIGN EXPIRE:1998-12-31
 Root CA des Individual Network e.V.
 Key fingerprint = 1B E1 4F F7 EC DE 3B 09  6E 5A 9F 6F 13 CB AF C7
```

Die oberen Daten sind die des gefälschten, die unteren die des authentischen 1997er-Signierschlüssels der IN-Root-CA. Entsprechend sind nachstehend zuerst die Schlüsselinformationen des gefälschten und dann die des authentischen Zertifizierungsschlüssels der *c't* pgpCA wiedergegeben:

```
Type Bits/KeyID    Date       User ID
pub  1024/604D8FFB 1997/03/05 ct magazine CERTIFICATE <pgpCA@ct.heise.de>
           Key fingerprint = 17 01 93 C0 06 59 27 6B  49 12 F6 D9 B2 14 D4 A5

pub  1024/BB1D9F6D 1997/03/04 ct magazine CERTIFICATE <pgpCA@ct.heise.de>
           Key fingerprint = 22 09 55 9D 72 60 87 B0  02 C3 71 9C 4E 0E 07 77
```

Original und Fälschung unterscheiden sich jeweils zwar noch in einigen Punkten (KeyID, hinterer Teil der UserID, Fingerprint, Erstellungsdatum), aber ein etwas entschlossenerer Angreifer hätte leicht auch einen gefälschten Schlüssel mit *identischem* Erstellungsdatum fabrizieren können – etwa indem er die Rechneruhr vor der Schlüsselerzeugung entsprechend zurückstellt. Er hätte daneben die User-ID völlig identisch wählen können und vielleicht sogar auch noch versucht, eine Fälschung mit derselben numerischen Key-ID wie der des betreffenden CA-Schlüssels zu generieren. (Daß es grundsätzlich möglich ist, einen Schlüssel mit derselben numerischen Kennung wie der eines anderen Keys zu erzeugen, zeigen der 1998er-Signierschlüssel der IN-Root-CA und der überregionalen IN-CA – siehe 5.5. Inzwischen gibt es sogar schon die ersten PGP-Schlüssel, bei denen alle 64 Bits der KeyID identisch sind, nicht nur die 32 Bits, die normalerweise angezeigt werden![20])

Diese Fälschungen sind möglich, weil man die User-ID bei PGP-Schlüsseln selbst bestimmen und beliebige öffentliche Schlüssel auf den Keyservern ablegen kann, ohne daß zuvor eine Identitätsprüfung stattgefunden hat. Leider kann man bei der Funktionsweise der etablierten PGP-Keyserver nicht verhindern, daß sich ein Spaßvogel oder Angreifer einen Schlüssel mit derselben Benutzerkennung wie der eines existierenden Schlüssels generiert und diese Fälschung auf die Keyserver lädt. Die Fälschung ist dann nur bei genauem Hinsehen anhand der Signaturen für diesen Schlüssel

[20] http://www.rubin.ch/pgp/keyring/

oder anhand des Fingerprints vom Original zu unterscheiden. Und selbst das kann unter Umständen noch nicht ausreichend sein: Es ist ratsam, immer Fingerprint, Schlüssellänge und Erstellungsdatum zu vergleichen, denn der Fingerprint alleine kann unter Umständen durch Variation der Schlüssellänge und/oder des Erstellungsdatums nachgeahmt werden [How97]; insofern müssen zusätzlich dazu auch die Schlüssellänge und das Erstellungsdatum eines Schlüssels verglichen werden, um sicher sein zu können, daß der authentische Key vorliegt. Die CA sollte daher immer diese drei Informationen zusammen weiterverteilen.

4.16.6 Vorträge

Vorträge innerhalb der UNI, in denen das neue Angebot ‘Zertifizierung’ vorgestellt und erläutert wird, bieten der CA ebenfalls eine Möglichkeit, auf sich aufmerksam zu machen – und zugleich die Chance, die Grundlagen und die Bedeutung von Verschlüsselung zu erläutern.

Zusätzlich könnte in anderen Kursen oder Schulungen, die das Rechenzentrum oder die Benutzerbetreuung durchführen, auf Verschlüsselung und auf die neuen Zertifizierungsdienste (oder auf entsprechende Vorträge) hingewiesen werden, wann immer sich dies thematisch anbietet, beispielsweise in Einsteiger- oder Fortgeschrittenen-Kursen zu *electronic mail*, zur Administration von Web-Servern oder bei Schulungsveranstaltungen mit Datenschutz-Bezug (natürlich auch bei dezidierten PGP-Vorträgen...).

In diesem Rahmen wäre es auch wichtig, nicht nur die wissenschaftlichen und studentischen Angehörigen der UNI als Zielgruppe anzupeilen, sondern auch die Mitarbeiter der Universitätsverwaltung einzubeziehen, schließlich wird auch dort ein wachsender Anteil der Kommunikation auf elektronischem Wege abgewickelt. In Absprache mit einzelnen Abteilungen der Verwaltung, die vielleicht schon besonders weit sind bei der Einführung von E-Mail an den Arbeitsplätzen, könnte dort gezielt eine „Sprechstunde“ oder eine Informationsveranstaltung der CA-Mitarbeiter angesetzt werden. Andere potentielle Adressaten wären Abteilungen, die wie die Personalstelle besonders sensible Daten verarbeiten und die daher verpflichtet sind, technische Vorkehrungen gegen unbefugten Einblick in diese Daten zu treffen.

4.16.7 Zertifizierung vor Ort

Die Low-Level UNI-CA wurde gezielt so konzipiert, daß sie weder technisch noch nach ihren Zertifizierungsrichtlinien an das Rechenzentrum gebunden ist. Sie soll vielmehr zu den Nutzern kommen können und vor Ort ihre Zertifizierungsdienste anbieten. Dazu könnte sich die Low-Level CA mit ihrem Zertifizierungsrechner und einigem geeigneten Informationsmaterial (Policy, Zertifizierungsanträge, Schlüsselinformationen auf Papier, Diskette oder CD-ROM, Werbe-Aufkleber für Verschlüsselung – siehe M) mit einer Art Stand z.B. an stark frequentierten Punkten auf dem Campus oder am Rande von Veranstaltungen mit Sicherheitsbezug oder starkem Publikumsverkehr o.ä. postieren. Sie könnte aber auch nach Absprache bzw. auf Anforderung einen Außentermin wahrnehmen. Auf diese Weise ließen sich, insbesondere, wenn ein geeigneter Blickfang (gläserner Rechner, überdimensionaler Schlüssel o.ä.) eingesetzt wird, Nutzer ansprechen, die sonst vielleicht nicht den Weg ins Rechenzentrum finden würden.

Solange die Nachfrage so gering ist, daß sich die Einrichtung von Registrierungsstellen nicht lohnt, besteht mit der Low-Level CA immerhin eine Möglichkeit, einzelne Zertifizierungstermine an anderen Standorten oder in anderen Einrichtungen der UNI abzuhalten, so daß nicht alle zertifizierungswilligen Nutzer den u.U. weiten Weg zum Rechenzentrum auf sich nehmen müssen.

4.16.8 Mails an zertifizierte Nutzer

Eine weitere Möglichkeit zur „Kundenpflege“ bestünde für die CA in regelmäßigen Rund-Mails an alle von ihr Zertifizierten. Darin könnten Hinweise auf aktuelle Sicherheitsprobleme in gängigen Programmen, auf neue Angebote der Zertifizierungsstelle, auf erfolgte Cross-Zertifizierungen u.ä. gegeben werden. Wenn die Policy, nach der ein User zertifiziert ist, die Möglichkeit einer Verlängerung eines Zertifikates vorsieht, so könnte man ihn auch einige Zeit vor Ablauf des alten Zertifikates daran erinnern und ihm so rechtzeitig vor dem Ablauf der Gültigkeitsdauer die Gelegenheit geben, wegen einer Re-Zertifizierung aktiv zu werden. Wenn eine Re-Zertifizierung (innerhalb gewisser Grenzen, was z.B. die Anzahl möglicher Re-Zertifizierungen angeht) automatisch erfolgt, könnte dem Nutzer auf diesem Weg sein neues Zertifikat zugestellt werden.

Die Zertifikatnehmer sollten auf dem Zertifizierungsantrag bei der Einwilligung in die Verarbeitung ihrer persönlichen Daten auch danach gefragt werden, ob sie derartige Mails wie die o.g. bekommen möchten. Wer grundsätzlich *gar keine* E-Mails von der CA bekommen möchte, sollte ausdrücklich bestätigen, daß er zur Kenntnis genommen hat, daß er dann auch bei eventuellen Sicherheitswarnungen nicht informiert werden kann.

Da die Möglichkeit, die Zertifikatnehmer per E-Mail zu informieren, für Schadensfälle wie das Bekanntwerden des CA-Signierschlüssels ohnehin vorgesehen und technisch eingerichtet werden muß (s. 5.4.5), wäre der zusätzlich erforderliche Aufwand für derartige Rund-Mails gering.

4.17 Cross-Zertifizierung

Eine Cross-Zertifizierung zwischen zwei Zertifizierungsinstanzen bedeutet, daß beide beteiligten CAs den Signierschlüssel der jeweils anderen CA bestätigen. Es findet also hier ausnahmsweise eine Zertifizierung „von gleich zu gleich“ statt; die Cross-Zertifizierung drückt keine Über- oder Unterordnung der einen der beiden beteiligten CAs unter die andere CA aus.

Die Cross-Zertifizierung ist nur für die vorgeschlagene Medium-Level UNI-CA vorgesehen; die Low-Level-CA führt keine Cross-Zertifizierungen durch.

Der Ablauf einer solchen CA-Cross-Zertifizierung unterscheidet sich nicht von dem bei der Zertifizierung eines Nutzers; es wird auch hier wieder die Identität des jeweiligen Gegenüber geprüft. Bei der Cross-Zertifizierung von CAs kommt nach der DFN-Policy lediglich noch die Prüfung hinzu, ob der jeweilige Gegenüber überhaupt der Einrichtung angehört, als deren CA-Mitarbeiter er sich ausgibt. Unter Umständen könnte man hier für die UNI-CA die Anforderungen bei einer Cross-Zertifizierung noch verschärfen, indem verlangt wird, daß sich ihre Mitarbeiter vor der Durchführung einer Cross-Zertifizierung vergewissern, daß der Mitarbeiter der anderen CA auch zur Vertretung seiner Einrichtung befugt ist. (Diese Vergewisserung könnte passieren, indem der Mitarbeiter eine Vollmacht seiner Einrichtung vorweist oder indem in der betreffenden Einrichtung rückgefragt wird.)

Wenn im Großraum der UNI bereits einige andere Zertifizierungsstellen existieren (z.B. an anderen Rechenzentren, Forschungseinrichtungen oder Hochschulen, eventuell auch von kommerziellen oder ehrenamtlich betriebenen CAs) oder sich im Aufbau befinden, dann könnte sich die UNI-CA bemühen, das lokale Zertifizierungsnetz durch Cross-Zertifizierungen mit diesen CAs zu verstärken und so die von den anderen CAs ausgestellten Zertifikate auch für ihre Zertifikatnehmer nutzbar zu machen.

Die DFN-Zertifizierungsrichtlinien sehen eine solche Möglichkeit ausdrücklich vor, und auch die DFN-PCA hat bereits mehrere Cross-Zertifizierungen absolviert, so unter anderem mit der pgp-CA der Fachzeitschrift *c't* und der Root-CA des Individual Network [IN97, DFN97] sowie einigen ausländischen CAs.

Eine Crosszertifizierungs-Vereinbarung mit einer anderen CA kann auch aus ganz pragmatischen Gründen zustandekommen, z.B. wenn einer der UNI-CA-Administratoren einen Mitarbeiter einer anderen CA trifft (z.B. weil er ihn persönlich kennt oder am Rande einer Tagung sieht) und beide diesen persönlichen Kontakt zu einer Cross-Zertifizierung nutzen wollen.

4.18 Kooperationsmöglichkeiten

Durch Kooperationen mit anderen Einrichtungen oder Projekten kann die UNI-CA nicht nur Synergieeffekte nutzen oder arbeitsteilig vorgehen, um unterschiedliche Tätigkeitsschwerpunkte auszunutzen und zu neuen Diensten zusammenzuführen, sondern sich auch einem breiteren Personenkreis bekanntmachen. Daher sollten derartige Möglichkeiten, wenn das Projekt ernsthaft und längerfristig verfolgt werden soll, unbedingt gesucht und genutzt werden.

Eine Zusammenarbeit mit der UNI-CA kann unterschiedlich ausfallen, je nachdem, was das Anliegen der kooperierenden Einrichtung ist. Die nachfolgenden Beispiele können Anregungen geben, mit wem und in welcher Form eine Zusammenarbeit der UNI-CA erfolgen könnte.

4.18.1 Registrierungsstellen oder nachgeordnete CAs

Die vielleicht naheliegendste Form der Zusammenarbeit könnte für die UNI-CA darin bestehen, daß sie in einzelnen Fachbereichen der UNI oder auch in einzelnen Abteilungen der Universitätsverwaltung Registrierungsinstanzen oder sogar Sub-CAs einrichtet bzw. dort eingerichtete CAs oder RAs zertifiziert.

4.18.1.1 Fachbereiche

Unter den Fachbereichen der UNI dürften diejenigen die geeignetsten Kandidaten für eine eigene Registrierungs- oder sogar Zertifizierungsstelle sein, die selber ein eigenes Rechenzentrum oder eigene Rechnernetze betreiben und administrieren – z.B. ein Fachbereich wie die Informatik –, da dort das für eine CA oder für deren Unterstützung erforderliche Know-how am ehesten vorhanden sein dürfte. (Die Informatik könnte u.U. auch ein akademisches Interesse am Betrieb einer eigenen CA haben.)

Daneben kämen noch solche Fachbereiche in Frage, die wie z.B. die Sozialwissenschaften oder die Medizin viel mit persönlichen Daten (Umfrageergebnisse, Krankendaten) zu tun haben oder die bereits einen Teil der Kommunikation zwischen den Studenten und den Hochschullehrern per E-Mail abwickeln und nun auf diesem Weg Noten oder Aufgabenstellungen vertraulich mitteilen möchten.

4.18.1.2 Verwaltung

Verfügt die Universitätsverwaltung über eine eigene EDV-Abteilung, so wäre auch dort zumindest die technische Umsetzung einer Registrierungs- oder Zertifizierungsstelle kein Problem. Bei einer Universität mit stark zersplittertem Campus und vielen getrennten Standorten könnte der Einsatz von Verschlüsselung auch die Arbeit der Verwaltung vereinfachen und beschleunigen: Dank digitaler Unterschriften könnte die Kommunikation per E-Mail eindeutig einem Urheber zugeordnet und so für andere Anwendungsbereiche als bisher genutzt werden, z.B. im Rahmen der internen elektronischen Beschaffung oder des elektronischen Bestellwesens.

4.18.2 Uni-Verwaltung

Die Kooperationsmöglichkeiten für die UNI-CA mit der Universitätsverwaltung sind vielfältiger und umfassen mehr Möglichkeiten als nur die Einrichtung von RAs oder Sub-CAs. Wie bereits in Abschnitt 4.16.6 erwähnt, beschränkt sich die Nutzbarkeit der Dienste, die eine Zertifizierungsstelle anbietet, nicht auf die wissenschaftliche Kommunikation – auch die Verwaltung könnte von den neuen Möglichkeiten, die Public-Key-Verfahren eröffnen, profitieren. Als Beispiel für eine denkbare Anwendung aus einem ähnlichen Umfeld mag hier der Zugriff über das Internet auf Großrechner-Daten aus dem firmen-internen Netz unter Verwendung von Zertifikat-basierter Authentifizierung und SSL(eay) dienen [Hub98]. Andere Gebiete, in denen sich ebenfalls Anwendungsmöglichkeiten ergeben, wären solche Bereiche, in denen vertrauliche Informationen gehandhabt werden (Personaldaten, Betriebsarzt, Finanzabteilung, Personalrat) oder Kooperationen mit externen Partnern, in deren Rahmen Daten zur Verfügung gestellt werden, die vertraulich behandelt werden müssen.

4.18.3 Externe Projekte und Einrichtungen

Als potentielle Partner für Cross-Zertifizierungen wurden andere örtliche CAs bereits genannt. Natürlich muß eine Zusammenarbeit mit ihnen nicht unbedingt (nur) in Form einer Cross-Zertifizierung erfolgen; ebenso wären konzertierte Aktionen oder ein Erfahrungsaustausch mit diesen Einrichtungen denkbar.

Es kommen jedoch nicht nur andere Zertifizierungsstellen als Kooperationspartner in Frage. Inhaltliche Anknüpfungspunkte gäbe es beispielsweise mit dem Forschungsprojekt DFN Directory Service (DDS)[21] der sich intensiv mit der Speicherung von Public-Keys – auch von PGP-Schlüsseln – im

[21] `http://www.directory.dfn.de/`

X.500-Verzeichnisdienst befaßt [Spa99], oder mit dem Verbundprojekt UNICORE[22] [AS98], das es zertifizierten Nutzern ermöglichen will, über ein WWW-Interface Aufträge für Berechnungen (*jobs*) an Höchstleistungsrechner absetzen zu können.

Eine weitere Möglichkeit wäre eine Beteiligung an der Initiative des Bundesministeriums für Wirtschaft und Technologie für mehr Sicherheit in der Informationsgesellschaft [BMW99a] (siehe Kap. 1).

Hinsichtlich der grafischen Gestaltung von Aufklebern, Logos und dergleichen könnte man sich am Vorgehen des Schleswig-Holsteinischen Landesbeauftragten für Datenschutz orientierten und beispielsweise auf das Know-how des Fachbereichs Design oder der örtlichen Kunsthochschule o.ä. zurückgreifen und damit zugleich dortigen Studenten praxisnahe und interessante Aufgabenstellungen anbieten.

4.19 Rechtliche Aspekte des CA-Betriebs

4.19.1 Krypto-Regulierung

Die Auseinandersetzung darüber, ob die Verwendung starker Verschlüsselungsverfahren reglementiert oder sogar verboten werden soll, weil sonst die staatlichen Sicherheitsinteressen gefährdet wären, bezeichnet man als „Krypto-Kontroverse“. In ihr stehen sich auf der einen Seite die Interessenvertreter der Sicherheitsorgane und Geheimdienste, auf der anderen die Advokaten eines Datenschutzes durch Technik und der freien Verwendung kryptographischer Verfahren zum Selbstschutz im Sinne des informationellen Selbstbestimmungsrechts gegenüber.

Während 1997 und 1998 vom damaligen Innenminister KANTHER eine Krypto-Regulierung[23] befürwortet wurde [SPI96, SPI98, CZ98g], steht die neue Bundesregierung dem Einsatz von Verschlüsselung erheblich wohlwollender gegenüber [Bir98, Kre99b], siehe Kapitel 1. Exemplarisch wird dieser Umschwung – von ‘Sinneswandel’ kann ja angesichts des Regierungswechsels schlecht gesprochen werden... – an der Aussage von MICHAEL HANGE, BSI-Vizepräsident, deutlich:

> „Wir möchten ... den Ausbau einer interoperablen Public-Key-Infrastruktur (PKI) vorantreiben und auf ein höheres Sicherheitsniveau – Stichwort Signaturgesetz – migrieren.“ [SH99]

Da immer wieder gegenteilige Aussagen zu hören sind, soll hier noch einmal in aller Deutlichkeit klargestellt werden:

> **Es gibt zur Zeit (März 2000) kein Verbot der Nutzung kryptographischer Verfahren in der Bundesrepublik Deutschland und auch keine gesetzliche Beschränkung der maximalen Schlüssellängen; anderslautende Behauptungen sind schlichtweg *falsch*!**

[22] `http://www.fz-juelich.de/zam/RD/coop/unicore/`
[23] `http://www.fitug.de/ulf/krypto/verbot.html`

Doch selbst wenn sich einmal die Verfechter eines Krypto-Verbotes durchsetzen sollten, bliebe fraglich, ob entsprechende gesetzliche Regelungen nicht spätestens vom Bundesverfassungsgericht gekippt würden. Ein Verbot oder eine gesetzliche Einschränkungen der Verwendung starker Verschlüsselung wäre in Deutschland ein Grundrechtseingriff bzw. eine Beschränkung der Grundfreiheiten, wie sie u.a. in der Europäischen Menschenrechtskonvention als Mindeststandards festgeschrieben sind, und – da unverhältnismäßig bzw. nicht zur Erreichung des beabsichtigten Zieles geeignet – unzulässig [BRR97, Dir98].

Auch auf internationaler Ebene gibt die Entwicklung Anlaß zum Optimismus: eine Lockerung entsprechender restriktiver gesetzlicher Regelungen war während der jüngsten Zeit in Großbritannien[CZ98b, Med99], Frankreich[CZ97, Jos99, Sch99] und Finnland[24] zu vermerken. Dem stehen allerdings auf der anderen Seite die Genehmigungspflicht für kryptographische Verfahren in Rußland sowie umstrittene Gesetzgebungspläne[25] in Großbritannien gegenüber, die in der “Regulation of Investigatory Powers Bill”[26] die Preisgabe geheimer Schlüssel vorsehen, wenn die Sicherheitsbehörden dies fordern, und die das Nichtbefolgen – z.B. auch bei gelöschten Schlüsseln o.ä. – mit bis zu zwei Jahren Haft unter Strafe stellen.

4.19.2 Haftung

Die nachfolgenden Ausführungen geben die augenblickliche rechtliche Situation Anfang des Jahres 2000 wieder. Da mittlerweile jedoch die EU-Richtlinie zu elektronischen Signaturen in Kraft ist und die nationalen Gesetzgeber verpflichtet sind, deren Bestimmungen bis zum 19. Juli 2001 in nationales Recht umzusetzen (siehe 3.5.3), dürfen die Haftungsregelungen nicht außer Acht gelassen werden, die die EU-Richtlinie vorsieht:

Der Zertifizierungsdiensteanbieter (= CA) haftet

- für die Richtigkeit der Informationen in einem qualifizierten Zertifikat (und für dessen Vollständigkeit im Sinne der Richtlinie)
- dafür, daß der Zertifikatnehmer zum Zeitpunkt der Zertifikaterstellung im Besitz derjenigen geheimen Signaturerstellungsdaten (= Secret-Key) ist, deren korrespondierende Signaturprüfdaten (= Public-Key) im Zertifikat zertifiziert werden
- die Anwendbarkeit der Signaturerstellungs- und -prüfdaten, falls sie vom Diensteanbieter erzeugt werden (also daß beide Komponenten eines Schlüsselpaares zusammenpassen)
- gegenüber den Zertifikatnutzern, falls er mindestens fahrlässig den Widerruf des Zertifikats nicht registriert hat

Dafür muß der Anbieter über die erforderlichen Finanzmittel verfügen oder entsprechend versichert sein.

Der Zertifizierungsdiensteanbieter haftet *nicht* für Schäden, die bei der Überschreitung einer im Zertifikat angegebenen Obergrenze des Transaktionswertes entstehen.

[24] `http://www.mintc.fi/www/sivut/dokumentit/tiedote/viestinta/ti260499508eng.htm`
[25] `http://www.fipr.org/rip/prfeb10.html`
[26] `http://www.homeoffice.gov.uk/oicd/ripbill.htm`

Noch sind sie in Deutschland zwar nicht geltendes Recht, die Gerichte werden sich aber in entsprechenden Streitfällen – ähnlich wie beim Datenschutzrecht, wo eine Umsetzung der EU-Richtlinie in deutsches Recht (trotz Fristablauf!) ebenfalls noch nicht erfolgt ist (vgl. 3.5.4) – bereits an den Bestimmungen der EU-Richtlinie orientieren, solange diese noch nicht ihren Niederschlag in Form von nationalen Gesetzen gefunden hat.

Haftung für Schäden aus der Zertifizierung

Nach der momentanen Rechtslage spielt es für die Frage der Haftung der Zertifizierungsstellen keine Rolle, ob es sich um eine Stelle im Sinne des § 2 SigG handelt oder nicht. Der Gesetzgeber hat bei Erlaß des Signaturgesetzes auf eine eigene Haftungsregelung sowie auf eine Haftungsbegrenzung – entgegen teilweise anderer Erwägungen im Gesetzgebungsverfahren – letztendlich doch verzichtet (s. 3.5.1). Somit gelten sowohl für Zertifizierungsstellen i.S.d. SigG als auch für sonstige Zertifizierungsstellen die allgemeinen zivilrechtlichen Haftungsregelungen:

Wenn aufgrund von technischen Defekten, betrügerischen Manipulationen oder fahrlässigem Fehlverhalten von Mitarbeitern der Zertifizierungsstelle bei der Ausgabe oder Verwaltung von Zertifikaten Schäden entweder beim Zertifikatnehmer oder bei einem am Zertifizierungsvorgang nicht beteiligten Dritten entstehen, haftet die Zertifizierungsstelle dafür.

Zwischen der Zertifizierungsstelle (bzw. der Institution, die die Stelle betreibt) und dem Inhaber des von dieser Stelle erteilten Zertifikates besteht ein vertragliches Schuldverhältnis, aus dem für jede der beteiligten Parteien entsprechende Haupt-, Schutz- und Nebenpflichten gegenüber der anderen Partei folgen. Seitens der Zertifizierungsstelle zählen hierzu im allgemeinen die ordnungsgemäße Schlüsselzuordnung und -verwaltung, die Ausstellung eines gültigen Zertifikates, in entsprechenden Fällen auch dessen Sperrung, die Einrichtung ausreichender Sicherheitsvorkehrungen, die zutreffende Auskunftserteilung bei Zertifikatsabfragen sowie ggf. auch die geeignete Generierung von Schlüsseln. Verstößt die Zertifizierungsstelle bzw. einer ihrer Mitarbeiter schuldhaft (auch fahrlässig) gegen eine der vorgenannten Pflichten, so hat die Zertifizierungsstelle hierfür gegenüber dem Vertragspartner einzustehen. Die Haftung der Zertifizierungsstelle umfaßt den Ersatz von Vermögensschäden ebenso wie den Gewinn, der dem Vertragspartner durch die Pflichtverletzung entgeht [Tim97].

Ein Haftungsausschluß in den Nutzungs-/Geschäftsbedingungen der Zertifizierungsstelle ist dabei nur möglich für den Fall fahrlässigen Verschuldens. Die Haftung für Schäden bei grob fahrlässigem und vorsätzlichem Verhalten ist nicht ausschließbar. Ebenso ist eine Haftungsbegrenzung auf eine bestimmte Schadenssumme nicht möglich. Diese Regelungen gelten auch dann, wenn die Zertifizierungsstelle ihre Leistungen unentgeltlich anbietet. Die Unentgeltlichkeit einer Leistung führt mithin nicht zu einem generellen Haftungsausschluß [Hil98].

Zwischen einem geschädigten Dritten und der Zertifizierungsstelle klafft hingegen eine Haftungslücke: Hier bestehen keine vertraglichen Beziehungen, das Produkthaftungsgesetz greift nur bei privater Nutzung von Produkten und gilt nicht für Dienstleistungen. Nach § 831 BGB haftet die Zertifizierungsstelle zwar für Schäden, die ihre Mitarbeiter als sog. Verrichtungsgehilfen bei Dritten verursachen. Die Zertifizierungsstelle haftet außerdem nach § 823 BGB, wenn sie ihre inneren Betriebsabläufe nicht so organisiert hat, daß die Anleitung und Überwachung der den Schaden verursachenden Mitarbeiter durch leitende Mitarbeiter gewährleistet ist. In diesen Fällen hat ein Geschädigter jedoch grundsätzlich nur einen Ersatzanspruch bei Beeinträchtigung sogenannter „absoluter Rechte und Rechtsgüter" (Leib, Leben, Gesundheit, Persönlichkeitsrecht, Eigentum) – solche Arten von Schäden dürften bei der typischen Tätigkeit einer Zertifizierungsstelle relativ selten sein [Tim97] –, jedoch nicht bei Vermögensschäden [Roß97, S. 79].

Sollte der Dritte jedoch durch Mitarbeiter der Zertifizierungsstelle auf betrügerische Weise oder sittenwidrig geschädigt worden sein, so haftet die Zertifizierungsstelle auch für Vermögensschäden (§§ 823 Abs. 2, 826 BGB). Da in beiden Fällen jedoch eine *vorsätzliche* Schädigung Voraussetzung ist, folgt aus lediglich fahrlässigem Verhalten keine Haftung der Zertifizierungsstelle für Vermögensschäden [Hil98].

Haftung für Schäden durch fehlerhafte Software

Bietet eine Zertifizierungsstelle nicht nur die direkten Zertifizierungsdienste an, sondern hält darüber hinaus, wie beispielsweise die DFN-PCA, auch Software auf ihrem WWW- oder FTP-Server zum Download bereit, so kann sie unter Umständen auch für Fehler oder Viren in diesen Programmen haftbar gemacht werden [Hoe97, S. 81–91].

Bei kostenlos verteilter Software ist die Sicherheitserwartung der Nutzer gering, entsprechend niedrig sind die Sorgfaltspflichten des Anbieters hinsichtlich Stichprobenkontrollen anzusetzen (insbesondere dann, wenn beispielsweise auf das Risiko des Virenbefalls hingewiesen wird oder Anti-Viren-Programme auf dem Server angeboten werden). Werden allerdings spezielle Programmpakete für den Anwender bereitgestellt, so kann dies gehobene Sicherheitserwartungen beim Nutzer wecken – insbesondere, wenn eine Zertifizierungsstelle dies tut –, die wiederum zu höheren Sorgfaltspflichten seitens des Anbieters führen. Er hat Produktbeobachtungspflichten und für Software, die über die Einrichtung lizensiert wird und für die Support o.ä. von ihm angeboten wird, eine Instruktionspflicht, d.h. eine Pflicht zur Einweisung in den Umgang mit dem Programm, wozu auch der Hinweis auf mögliche Gefahren zählen kann [Hoe97, S. 86 f.].

4.19.3 Versicherungsschutz

Wie im vorigen Abschnitt dargelegt, haftet eine Zertifizierungsstelle, auch eine, die unentgeltlich tätig ist, in bestimmten Situationen für Schäden, die aus ihrer Arbeit resultieren [Tim97]. Das Fazit in einer entsprechenden Stellungnahme für den Individual Network e.V. [Hil98] lautete daher:

> „Es ist ratsam, wenn die Zertifizierungsstellen entsprechende Haftpflichtversicherungen abschließen. Ansonsten haftet allein die Organisation, die die Zertifizierungsstelle trägt, mit ihrem gesamten Vermögen."

Ein solcher Versicherungsschutz ist allerdings nicht so einfach zu bekommen: Der Gerling-Versicherungskonzern, an neuen Geschäftsfeldern sonst durchaus interessiert und eher aufgeschlossen gegenüber neuen Ideen, lehnte es beispielsweise bislang aus grundsätzlichen Erwägungen ab, eine entsprechende Police abzuschließen. Nach Auskunft von LUTZ DONNERHACKE, Mitarbeiter der bei Gerling anfragenden und auf dem Gebiet der Zertifizierung kommerziell tätigen IKS GmbH:

„Gerling kann sich nicht entscheiden. Deutsches Recht sieht keine Haftung für CAs vor, EU-Recht sieht durchaus eine Haftung vor.“ Hinzu kommt, daß viele Versicherungen vorhersagbare Risiken im normalen Geschäftsablauf grundsätzlich nicht versichern [Kar99] und aus diesem Grund auch nicht bereit sind, die Tätigkeit einer Zertifizierungsstelle entsprechend abzusichern. Darüber hinaus mag noch ein ganz anderer Grund ursächlich sein für die Zurückhaltung der Versicherungsunternehmen: Es ist im Vorhinein in keiner Weise absehbar, gegenüber wem und für welche Zwecke ein Zertifikatnehmer seinen Schlüssel einsetzen wird, daher ist eine Risikokalkulation und Versicherung des so ermittelten Risikos kaum möglich [Tim97].

Es ist zur Zeit also noch schwierig, die Tätigkeit von Zertifizierungsstellen zu versichern, zumal bislang auch kaum Erfahrungen mit deren Arbeit vorliegen, die es den Versicherungen vielleicht eher ermöglichen würden, das Schadensrisiko einzuschätzen und die Höhe der Prämien zu kalkulieren. Eventuell bringen hier die Haftungsregelungen, wie sie im Entwurf zu einer EU-Richtlinie zu elektronischen Signaturen vorgesehen sind (s. 3.5.3), Bewegung in den Markt. Diese wäre umso wünschenswerter, als die möglichen Kosten aufgrund von Rechtsstreitigkeiten, die aus Internet-Angeboten resultieren, schnell ein erhebliches Ausmaß annehmen können: Die angebotenen Informationen (hier: Zertifikate) können jederzeit weltweit abgerufen und genutzt werden, häufig ist dann der Ort des Abrufs oder des Schadens der Ort der rechtlichen Auseinandersetzung [Rec98].

4.19.4 Möglichkeit zu SigG-konformer Arbeit

Wenn schon eine Zertifizierungsstelle an der UNI eingerichtet werden soll, stellt sich angesichts des 1997 verabschiedeten Signaturgesetzes die Frage, ob es nicht sinnvoll und möglich wäre, die UNI-CA als Zertifizierungsstelle im Sinne des SigG zu betreiben. Dies dürfte sich aber nur schwer und mit hohem finanziellen, personellen, baulichen und organisatorischen Aufwand umsetzen lassen, denn SigG und SigV sowie die zugehörigen Maßnahmenkataloge für Zertifizierungsstellen bzw. für technische Komponenten stellen sehr hohe Sicherheitsanforderungen.[27]

Die Zertifizierung selbst durch die zuständige Behörde soll zwar nach der Schätzung der Bundesregierung nur etwa 3 000 bis 5 000 DM kosten [BRD97], die Kosten, die vorher und im laufenden Betrieb für Umbauten, Sicherungsmaßnahmen und Personal anfallen, dürften jedoch um Größenordnungen darüber liegen.

So sieht der Maßnahmenkatalog nach § 12 Abs. 2 SigV [RTP98, MZ 5.4] vor, daß ein Notfallkonzept gewährleisten muß, daß auch im Falle beispielsweise eines Brandes mögliche Ausfallzeiten des von der Zertifizierungsstelle obligatorisch anzubietenden Verzeichnisdienstes „auf ein Minimum beschränkt bleiben“. Was darunter genau zu verstehen ist, wurde aus dem sehr viel ausführlicheren, 300seitigen *ENTWURF Maßnahmenkatalog für digitale Signaturen* [RTP97, S. 59] deutlich, in dem es dazu heißt, daß folgende „Obergrenzen einzuhalten sind: 1. Die maximale Ausfallzeit muß kleiner als drei Stunden sein. [...]“ Auch wenn dieser detaillierte Katalog in dieser Form nicht mehr als endgültige Empfehlung verabschiedet wurde, kann man doch davon ausgehen, daß letztlich die darin genannten Empfehlungen der Maßstab für die Sicherheitsvorkehrungen in der Zertifizierungsstel-

[27] Nicht alle Maßnahmen, die in den Katalogen genannt werden, sind obligatorisch; einige Punkte stellen bloß Empfehlungen dar, bei deren Einhaltung die Erfüllung der Sicherheitsanforderungen nach SigG/SigV als gewährleistet angesehen wird. Wird auf diese Maßnahmen verzichtet oder von den Empfehlungen abgewichen, so ist die Gleichwertigkeit der stattdessen ergriffenen Maßnahmen oder Vorkehrungen nachzuweisen.

le sind, den die Prüfstellen anlegen werden, wenn sie die Umsetzung eines SigG/SigV-konformen Sicherheitskonzeptes in der Zertifizierungsstelle bestätigen sollen.

Dem ausführlichen Katalog-Entwurf zufolge ist weiterhin für die besonders sicherheitsrelevanten Schritte bei der Schlüsselerzeugung für die CA (und gegebenenfalls für die Zertifikatnehmer) sowie bei der Erstellung oder der Sperrung von Zertifikaten das Vier-Augen-Prinzip anzuwenden, so daß also für die betreffenden Arbeitsschritte jeweils zwei CA-Mitarbeiter vorgesehen werden müßten. Die Zertifizierungsstelle wird mindestens drei oder vier Mitarbeiter haben müssen, um eine Rollentrennung realisieren zu können (diejenigen Mitarbeiter, die für die Registrierung und Identifizierung der Zertifizierungswilligen bzw. für das Ausstellen der Zertifikate zuständig sind, dürfen zum Beispiel nicht zugleich die Rolle des internen Revisors innehaben).

Nach § 13 Abs. 2 hat die Zertifizierungsstelle über die Sicherheitsmaßnahmen und über die konkreten, im laufenden Betrieb anfallenden Daten eine Dokumentation zu führen und diese mindestens 35 Jahre lang aufzubewahren und sicherzustellen, daß die erforderlichen Geräte zur Verfügung stehen, um digitale Daten aus der Dokumentation während dieser Zeitspanne abrufen zu können. (Immerhin muß die Dokumentation von Auskünften an *Sicherheitsbehörden* im Vergleich dazu nur 12 Monate aufbewahrt werden und kann dann gelöscht werden.) Es muß nicht nur die Verfügbarkeit der Dokumentation, also auch die Lesbarkeit digitaler Dokumente, während dieses Zeitraumes sichergestellt werden, sondern diese Unterlagen müssen auch vor dem Zugriff Unbefugter geschützt aufbewahrt werden. Es wären also voraussichtlich auch gesicherte Lagerräume für das Archiv der Zertifizierungsstelle notwendig. Ferner wären die Ausgabe und vorherige Personalisierung von technischen Komponenten, die nach dem SigG die Signaturschlüssel der Anwender speichern müssen (z.B. Chipkarten), nebst entsprechender gesicherter Lager- und Verarbeitungsräume vorzusehen.

Insgesamt müßten für die verschiedenen Aufgabenbereiche der Zertifizierungsstelle (Registrierung, Schlüsselgenerierung/Personalisierung des Schlüsselträgers, Zertifizierung, Verzeichnisdienst) Sicherheitsbereiche gebildet werden, die räumlich und baulich voneinander abgegrenzt sein müßten. Der Bereich der Schlüsselgenerierung bzw. -zertifizierung müßte durch Personenschleusen abgetrennt, zutrittsgeschützt und darüber hinaus gegen kompromittierende elektromagnetische Abstrahlung geschützt sein, für Publikumsverkehr dürfte nur der Registrierungs- und Ausgabebereich zugänglich sein [Rei97b, S. 38]. Zutritts- und Sabotageschutz für die Räume der Zertifizierungsstelle würden auch eine entsprechend widerstandsfähige Außenhaut des Gebäudes nebst Überwachungs- und Alarmanlagen einschließen [RTP97, S. 52 f.][SW].

Auch für die Anwender SigG-konformer Zertifikate dürften erhebliche Kosten entstehen: Die einzusetzenden technischen Komponenten zur Speicherung der Schlüssel, zum Erstellen und Prüfen von Signaturen nach dem SigG müssen nach den *Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik* eine hohe Sicherheitseinstufung aufweisen, sie werden folglich wegen des damit verbundenen Aufwandes bei der Herstellung der Geräte und für die entsprechende Sicherheitsprüfung nicht unerhebliche Kosten verursachen. Es ist, wenigstens zur Zeit, fraglich, ob die potentiellen Nutzer einer SigG-konformen Zertifizierungsstelle bereit wären, diese Kosten sowohl für die auf ihrer Seite erforderliche Hardware als auch für die Zertifizierung selbst zu tragen. Für die Universität müßte ggf. eine umfassende Kosten-Nutzen-Abschätzung vorgenommen werden, um zu prüfen, ob der mögliche Vorteil und das Einsparungspotential durch den Einsatz von digitalen Signaturen nach dem SigG so groß sind, daß sie die Kosten für die Einrichtung und den Betrieb einer gesetzeskonformen Zertifizierungsstelle überwiegen.

Eventuell ist aber auch die Etablierung einer Alternative zu SigG-CAs wichtig und man ist eines Tages froh darüber, daß es sie gibt. Bei einem Ausfall der einen Infrastruktur, z.B. wegen eines Durchbruchs in der numerischen Mathematik oder in der Kryptanalyse, stünde dann immer noch die andere Infrastruktur zur Verfügung [Zie97, S. 343]. Das würde aber voraussetzen, daß in beiden separaten Infrastrukturen auch unterschiedliche Algorithmen verwendet werden, was bislang aber nicht der Fall ist.

Wenn die EU-Signatur-Richtlinie in deutsches Recht umgesetzt sein wird – fristgerecht müßte dies bis spätestens Mitte 2001 geschehen –, wird sich auch eine andere Möglichkeit zur SigG-konformen Arbeit eröffnen: Wenn dann eine entsprechende Versicherung abgeschlossen werden könnte (eventuell DFN-weit für alle DFN-CAs?), die etwaige Haftungsrisiken abdecken würde, dann ließe sich eine DFN-(P)CA nach der EU-Signatur-Richtlinie betreiben. Sie würde dann zwar vielleicht nur einfache (nicht die 'qualifizierten') Zertifikate gemäß der Richtlinie bzw. gemäß der dann geltenden Neufassung des SigG ausstellen, solchen Zertifikaten bzw. den mit ihnen verbundenen elektronischen Signaturen dürfte aber nicht von vornherein die Wirksamkeit und Zulässigkeit als Beweismittel in Gerichtsverfahren abgesprochen werden.

Eine weitere Möglichkeit zu einer SigG-konformen Arbeit könnte dann gegeben sein, wenn Angebote wie das eines „virtuellen Trustcenter“, das die Deutsche Post SignTrust anbietet[28], nach dem SigG anerkannt werden.

4.19.5 Beweiswert nicht SigG-konformer Signaturen

Das SigG *zwingt* nicht zu SigG-Zertifizierungsstelle und -Signaturen, sondern ist nur ein Angebot, das eine gewisse rechtliche Bevorzugung genießt – Stichwort „Vertrauensvorschuß“. Vor Gericht sind SigG-konforme Signaturen daher ebenso wie Signaturen, die mit anderen Verfahren erstellt wurden, der freien Beweiswürdigung des Richters unterworfen; die SigG-Signaturen haben dann lediglich den Vorteil für sich, daß bei ihnen eine hohe Sicherheit vermutet werden dürfte, während dies bei Signaturen nach anderen Verfahren erst glaubhaft gemacht werden muß (bei dieser komplexen und spezifischen Materie vermutlich durch einen Gutachter). Da es aber noch keine Streitfälle, geschweige denn Rechtssprechung, zum Thema Signaturgesetz gibt, läßt sich noch nicht absehen, wie die Richter gegebenenfalls urteilen werden und ob beispielsweise SigG-Signaturen tatsächlich einen Vorteil gegenüber sonstigen Signaturen bieten, falls es zu einem Rechtsstreit kommt, oder ob nicht andere Verfahren bei entsprechend gutachterlich bestätigter Tauglichkeit eventuell ebenso anerkannt werden wie (möglicherweise) die Signaturen nach dem SigG.

Anders sieht die Situation aus, wenn – wie in § 1 Abs. 2 SigG vorgesehen – in zukünftigen Rechtsnormen ausdrücklich die Form der digitalen Signatur nach dem SigG verlangt oder sie als Alternative zur Schriftform zugelassen wird.[29] In diesen Fällen würden nicht SigG-konforme Verfahren von vornherein nicht in Frage kommen.

Davon abgesehen werden manche der DFN-CA-Zertifikate sowieso nie SigG-konform sein können, da sie nicht für Personen, sondern für Server – also für Maschinen bzw. für Computer und darauf laufende (SSL-Server-)Programme – ausgestellt werden. Von daher kann zumindest die DFN-

[28] Faltblatt *Virtual eTrust – „Werden Sie Zertifizierungsstelle für digitale Signaturen.“*

[29] Ein erster entsprechender Gesetzentwurf zur „Elektronischen Form“ existiert bereits.

Server-CA nie SigG-konform sein, und die DFN-PCA folglich auch nicht, wenn sie diese Server-CA zertifiziert!

Aber auch andere arbeiten nicht unbedingt nach den Vorgaben des Signaturgesetzes, wenn es um Schlüsselzertifizierung geht, sondern verfolgen eigene Ideen und Ziele und arbeiten nach eigenen Regeln. So hat z.B. ein internationales Konsortium unter Beteiligung der Deutschen, der Dresdner Bank und der HypoVereinsbank mit *Identrus*[30] (vormals *Global Trust Enterprise*) ein weltweites Unternehmen gegründet, das mit den Finanzinstituten seiner Gründungsfirmen als CAs Identitätszertifikate auf der Basis der Root-CA von CertCo[31] ausstellen will [Cer98, CZ98a]. Und es ist doch mehr als fraglich, ob bei dieser weltumspannenden Kooperation ausgerechnet die Bestimmungen des *deutschen* Signaturgesetzes angewandt werden...

4.19.6 Exportkontrolle

Nach der Neuregelung der Exportkontrollen gemäß dem Wassenaar-Abkommen ist jetzt Verschlüsselungssoftware bis 56 Bit symmetrischer Schlüssellänge und bei Massenprodukten bis 64 Bit Schlüssellänge von der Exportkontrolle ausgenommen und darf beliebig exportiert werden [CZ98c]. Bisher war Massenmarkt- und Public-Domain-Software nach der EG-Dual-Use-Verordnung von der Exportkontrolle ausgenommen; das neue Abkommen ist also in diesem Punkt restriktiver, da nun auch die Schlüssellänge entscheidet, ob eine Ausfuhrgenehmigung erforderlich ist [SH98c].

Software für größere Schlüssellängen darf nicht ohne weiteres exportiert werden (sie steht unter Exportkontrolle); das bedeutet aber auch nicht, daß ihre Ausfuhr grundsätzlich verboten wäre. Es ist Sache der einzelnen Mitgliedsländer des Wassenaar-Abkommens, wie sie die vereinbarten Regeln national gesetzgeberisch umsetzen bzw. ob ein Produkt, das unter Exportkontrolle steht, trotzdem ausgeführt werden darf. Außerdem scheint es dabei auch einigen Interpretationsspielraum zu geben. So hat Dänemark bereits einen Journalisten dazu angehalten, die Download-Möglichkeit von PGP von seiner Web-Site zu deaktivieren. Die Bundesrepublik will hingegen erst in sechs Monaten mit der nationalen Umsetzung der Übereinkunft beginnen. Im Wirtschaftsministerium teilt man die dänische Auffassung, PGP sei “Mass Market Software” und damit exportkontrollpflichtig, nicht [CZ99b, Moe98]. Die Internet Engineering Steering Group (IESG) und das Internet Architecture Board (IAB) haben hingegen mit einem scharfen Protest auf die neuen Vereinbarungen reagiert [BC98], vermutlich weil die Mitgliedsstaaten *verpflichtet* sind, die – meist einstimmig getroffenen – Vereinbarungen in einem bestimmten Zeitrahmen in nationales Recht zu übertragen und anzuwenden [Rot98a, S. 9]. – Im Zweifelsfall hilft vielleicht eine Anfrage an das Bundesausfuhramt (siehe Anhang M) weiter.

> „Die (käuflichen) Web-Browser (z.B. Netscape Navigator Gold) sind als mass market software für jedermann frei erhältlich und dazu entwickelt, vom Benutzer selbst installiert zu werden. ... Die im Internet kostenlos erhältlichen Web-Browser sind als Public Domain Software genehmigungsfrei.“ [Rot98a, S. 9]

Doch richtig aufatmen kann man z.B. als Administrator eines FTP-Servers trotzdem nicht – es gibt etliche andere Software mit kryptographischer Funktionalität, deren Export aus der Bundesrepu-

[30] `http://www.identrus.com`
[31] `http://www.certco.com`

blik zwar nicht genehmigungspflichtig oder gar verboten sein mag, die aber sehr wohl unter die *US-amerikanischen Re-Exportkontrollen* fällt. Diese extraterritorialen Kontrollen sind zwar völkerrechtswidrig, das hindert die Vereinigten Staaten aber nicht daran, sie bei sich bietender Gelegenheit durchzusetzen (z.B. bei Urlaubsreisen der Betroffenen in die USA). Da die meisten Software-Bibliotheken von US-Firmen produziert und angeboten werden, außerhalb der USA entwickelte Software meistens mindestens eine dieser Standard-Bibliotheken benutzt, fällt die so entwickelte Software nach US-Rechtsauffassung unter die amerikanischen Re-Exportkontrollen [Rot98b].[32]

4.20 Entwicklungsperspektiven

Eine Weiterentwicklung der UNI-CA könnte in mehrere Richtungen – durchaus auch zeitlich parallel – und aus unterschiedlichen Gründen geschehen:

- wachsende Nachfrage bis hin zur vollen Ausschöpfung des existierenden Potentials an Nutzern
- bei höheren Sicherheitsanforderungen ($\Longrightarrow$ technische, organisatorische und bauliche Maßnahmen)
- neue Anwendungen und Dienste

4.20.1 Fortschreibung und Anpassung der Policies

Bei den Policies dürfte eine kontinuierliche Fortschreibung sinnvoll sein. Das bedeutet nicht, daß ständig Änderungen daran vorgenommen werden sollten, man sollte sie in einer einmal geltenden Form aber andererseits auch nicht als „für die Ewigkeit“ formuliert betrachten. Die DFN-PCA entwickelt ihre Policies weiter (die nächste Änderung wird vermutlich, wenn nichts Unvorhergesehenes passiert, zum 1. Januar 2001 vorgenommen werden, denn dann laufen die geltenden Policies und das aktuelle PCA-Projekt aus), insofern könnten Anpassungen der UNI-Policies erforderlich werden, sofern weiterhin eine Zertifizierung durch die DFN-PCA gewünscht wird. In diesem Zusammenhang muß sich auch zeigen, ob besser ein monolithisches Policy-Dokument (mit internen Fallunterscheidungen) oder besser mehrere format- oder anwendungsspezifische Policies zum Einsatz kommen sollten. Die DFN-PCA hat bisher versucht, mit ihren Policies anwendungs-, z.T. auch format-übergreifend zu bleiben, während in diesem Konzept für die UNI-CA eine Trennung zumindest nach den Zertifikatformaten vorgeschlagen wird. Die Policies werden auch vor dem Hintergrund der individuellen Gegebenheiten und Erfahrungen fortgeschrieben werden müssen, die die jeweilige CA mit ihnen in der Praxis gemacht hat. Mit der enger werdenden europäischen Integration dürfte beispielsweise die Frage häufiger auftreten, wie ausländische Ausweisdokumente bei der Identitätsprüfung gehandhabt werden sollten. (Kein CA-Mitarbeiter wird alle Pässe oder Ausweispapiere auch nur der EU-Staaten so gut kennen, daß er Fälschungen sofort als solche erkennen könnte.)

[32] Interessant wäre es in diesem Zusammenhang zu erfahren, ob auch solche Programme unter die US-Re-Exportkontrollen fallen, die nicht statisch gelinkt sind, folglich auch keine Bestandteile von US-Software enthalten, aber u.U. *dynamisch* gebundene US-Software-Bibliotheken nutzen.

Bei wachsenden Nutzerzahlen wird eine möglichst effiziente Handhabung und Durchführung der einzelnen Arbeitsschritte einer Zertifizierung immer wichtiger.[33] Der Automatisierung des Zertifizierungsprozesses (sei es durch selbstgeschriebene Skripte, sei es durch den Einsatz neuer CA-Software wie beispielsweise der, die von SIMON FRISCHEISEN im Rahmen einer Diplomarbeit an der TU München [Fri99] entwickelt wird) kommt daher mit zunehmender Nachfrage nach den Zertifizierungsdiensten eine immer größere Bedeutung zu. Ein Punkt, an dem die CA-Arbeit sich voraussichtlich konzentrieren dürfte, ist der hohe Arbeitsaufwand bei der Verwaltung der zertifizierten Keys und der Re-Zertifizierung bei einem Schlüsselwechsel der CA. Überhaupt sind die Funktionen und Protokolle für ein effizientes und für den Anwender möglichst transparentes Key-Management gerade erst im Entstehen und noch lange nicht umfassend in der Praxis erprobt. Hier darf man auch auf die Erfahrungen der SigG-Zertifizierungsstellen gespannt sein, so sie denn hoffentlich einen Teil ihrer Erfahrungen auch publizieren werden.

Ein weiterer Punkt, der sicher erst in einiger Zeit relevant werden wird, wenn die gesicherte Individualkommunikation etabliert ist, dürfte die Zertifizierung von Gruppenschlüsseln, z.B. für verschlüsselte Mailinglisten sein. Dabei werden dann möglicherweise auch Performance-Fragen eine größere Rolle spielen (z.B. bei großen Mailinglisten mit mehreren tausend Empfängern). Auch eine stärkere Unterstützung von Anonymitäts- und/oder Pseudonymitätsdiensten – gerade durch eine Forschungseinrichtung! – wäre denkbar.

Die weiteren Fortschritte der Kryptographie können schließlich ebenfalls eine Entwicklung anstoßen oder ein nicht eingeplantes Vorgehen erforderlich machen. So ist nicht auszuschließen, daß die bislang entdeckten Schwachpunkte im Hash-Algorithmus MD5 in Zukunft so weit ausgenutzt werden könnten, daß er als *Message Digest* nicht mehr weiter eingesetzt werden kann. Im Fall eines kryptographischen oder mathematischen Durchbruches könnte sogar das RSA-Verfahren gebrochen werden; dann müßte überall, wo es bisher eingesetzt wird, auf andere Verfahren oder Verfahrensklassen ausgewichen werden. In beiden Fällen würde der Kompatibilitätsaspekt, der in diesem Konzept noch als Begründung für das Festhalten an PGP 2.6 mit seinem MD5-Message Digest genannt wird (s. 4.8.1), gegenüber den Sicherheitsbelangen in den Hintergrund treten. Um für solch einen Fall besser gewappnet zu sein, ist es auch wichtig, daß neue Verschlüsselungsstandards bzw. -formate so flexibel formuliert werden, daß ein Wechsel des Verschlüsselungs- oder des Message-Digest-Verfahrens möglich ist, ohne deswegen den Standard oder den restlichen Teil der Software ändern zu müssen.[34]

Die beiden letzten Punkte, Zertifizierung von Gruppenschlüsseln und Weiterentwicklung der Kryptographie, könnten u.U. ebenfalls eine Änderung der Policies nach sich ziehen. Gleiches gilt für die Nachbereitung oder Aufarbeitung von sicherheitsrelevanten Vorfällen in einer CA. Gegebenenfalls müssen aus derartigen Geschehnissen auch entsprechende Konsequenzen gezogen werden, z.B. für die Arbeitsorganisation oder die erforderlichen Schutzmaßnahmen, so daß eine Policy-Anpassung unumgänglich wird.

[33] Siehe dazu auch [FSV98]; über Praxiserfahrung im Umgang mit einer großen Zahl von Zertifizierungsanträgen kann heute bereits die pgpCA der *c't* Auskunft geben (Anschrift s. Anhang M)

[34] Im zukünftigen Internet-Protokoll IPv6 haben die Autoren diesem Umstand beispielsweise schon Rechnung getragen.

4.20.2 Verlagerung des Schwerpunktes der Arbeit in der Zertifizierungsstelle

Je nach der Phase, in der sich die UNI-CA befindet, wird sich der voraussichtliche Schwerpunkt des Arbeitsaufwandes für den CA-Betrieb im Laufe der Zeit verlagern:

Phase	Aktivitätsschwerpunkt
Einrichtung	Beschaffung, Installation
Inbetriebnahme	Einspielen der Abläufe und Verantwortlichkeiten, Bekanntmachen der Dienste
Wachstum	Dokumentation, Aufklärung und Sensibilisierung der (potentiellen) Anwender
Fortbestehen	Aspekte des Schlüsselmanagements (Re-Zertifizierung, Key-Rollover)
Massenbetrieb	Automatisierung, Pflege der Dokumentation, Überarbeitung des Sicherheitskonzeptes

4.20.3 Ausbaustufen

Die nachfolgende Tabelle zeigt eine mögliche Abfolge von Ausbaustufen der UNI-CA nebst einer Schätzung, wie lange die jeweilige Phase etwa dauern könnten. Die zeitliche Abfolge ist dabei in keiner Weise zwingend oder zwangsläufig, naheliegend ist nur, daß die ersten beiden genannten Punkte den übrigen vorangehen dürften.

Dauer	Aktivität
3 Mann-Monate (MM)	Vorbereitung, Aufbau und Inbetriebnahme der Low-Level CA
1 MM	Inbetriebnahme der Medium-Level CA
1 – 2 MM	Unterstützung von X.509-Zertifizierungen
$\frac{1}{2}$ MM	Zertifizierung von PGP 5.x-/6.x-/OpenPGP-Schlüsseln
1 MM	Eröffnung von Registrierungsstellen
2 MM	Einrichtung von nachgeordneten CAs

4.20.4 Zertifizierungs-Hardware

Nicht nur die Policy kann oder muß im Laufe der Zeit weiterentwickelt werden; auch die Hardware-Ausstattung der Zertifizierungsstelle kann ausgebaut oder verfeinert werden. Dabei können verschiedene Aspekte betont werden, die in den folgenden Abschnitten genauer beschrieben werden. Zur Zeit ist die Auswahl an geeigneten Produkten, die auch unter freien Unix-Versionen unterstützt werden, noch nicht sehr groß, daher werden im Anhang B.3 ff. Geräte beispielhaft genannt, die diese Anforderung erfüllen.

4.20.4.1 Schutz vor elektromagnetischer Abstrahlung (TEMPEST)

[Luc96, Kuh98] beschreiben Angriffe, die die elektromagnetische Abstrahlung (EMA) eines Rechners ausnutzen, um ohne direkten physikalischen Zugang oder Zugriff auf den Rechner Informationen „abzuhören“. Bezogen auf den Zertifizierungsrechner könnte das beispielsweise bedeuten, daß ein Angreifer die Tastendrücke aufzeichnet, die bei der Eingabe der Passphrase für den Signierschlüssel passieren. Schafft er es dann noch, Zugriff auf den Schlüsseldatenträger zu bekommen, kann er den geheimen Signierschlüssel der CA kopieren und dann mißbräuchlich einsetzen.

Eine Entwicklungsmöglichkeit oder Ausbaustufe der UNI-CA könnte daher darauf abzielen, diese Strahlung beim CA-Rechner stärker abzuschirmen und so mögliche TEMPEST-Angriffe zu erschweren. Dies könnte entweder durch einen dezidierten abgeschirmten Rechnerraum (Faradayscher Käfig; leitende, metalldurchwirkte Tapeten u.ä.) oder aber durch den Einsatz eines speziell gegen solche Strahlung abgeschirmten CA-Rechners geschehen.[35] Der Preis für einen PC in TEMPEST-Ausführung liegt laut WOLFF [Wol98] bei rund 30 000 DM; ein dazu passender entsprechend abgeschirmter Drucker kostet zusätzlich noch einmal ca. 15 000 DM. LUCKHARDT [Luc96] schreibt, man müsse bei TEMPEST-Hardware etwa mit dem Vierfachen des Normalpreises rechnen.

Nachteilig ist, daß der TEMPEST-Schutz völlig neu ausgemessen werden muß, wenn auch nur ein Teil (z.B. eine Controller-Karte) am Gesamtsystem ausgetauscht wird. Ein weiteres Problem bei TEMPEST-Rechnern ist der hohe zeitliche Aufwand bei der Herstellung und die geringe Nachfrage, die zusammen dazu führen, daß TEMPEST-Geräte der schnellebigen Informationstechnik meist um etwa drei Jahre hinterherhinken. Man kann also heutzutage einen TEMPEST-486er-PC oder gerade einmal einen Pentium-PC in TEMPEST-Ausführung bekommen, jedoch keinen *State-of-the-art*-Rechner [Wol98].

Eine andere Schutzvorkehrung gegen EMA kann im Einsatz eines „Störsenders“ bestehen, der in einem Umkreis um den Einsatzort (nahe am Zertifizierungsrechner) die EMA auf allen Frequenzen mit anderen Signalen dergestalt überlagert, daß ein Lauscher mit seinen Empfangsgeräten die für ihn interessanten Signale nicht mehr empfangen kann. Ein Beispiel für ein solches im Handel erhältliches Gerät ist das *Data Safety Device* (siehe Anhang B.3).

4.20.4.2 Chipkarten

Eine Erhöhung der Sicherheit ließe sich u.U. auch durch den Einsatz von Chipkarten erreichen, auf denen die geheimen CA-Schlüssel gespeichert werden können und die diese nie preisgeben. Dafür müssen diese Karten zusätzlich in der Lage sein, entsprechende Ver- oder Entschlüsselungs- und Signier-Operationen selbst und ausschließlich im Mikroprozessor der Chipkarte durchzuführen, ohne daß dabei die Zentraleinheit des Zertifizierungsrechners gebraucht wird. Derartige Krypto-Chipkarten werden durch das Signaturgesetz und dessen Umsetzung einen Nachfrage-Schub bekommen und dann hoffentlich durch die großen Stückzahlen auch erheblich preiswerter werden.

Aber auch Chipkarten sind nicht 100prozentig sicher [AK96, Zie98], man sollte sich daher nicht der trügerischen Annahme hingeben, sobald (Krypto-)Chipkarten eingesetzt werden, wären keine Angriffe auf die CA oder ihre geheimen Schlüssel mehr möglich.

[35] Solche Rechner sind mittlerweile am freien Markt erhältlich, u.a. bei Siemens/Fürth (siehe Anhang B.3); allerdings wird die Käuferadresse an das Bundesamt für Sicherheit in der Informationstechnik (BSI) übermittelt [Luc96]. Weitere Anbieter finden sich auf der TEMPEST-Web-Seite von JOEL MCNAMARA (siehe Anhang A.1)

4.20.4.3 Dezidierte Krypto-Hardware

Die Langzahl-Rechenoperationen, die für die gängigen Public-Key-Operationen ausgeführt werden müssen, sind aufwendig und verbrauchen, gerade wenn viele solcher Operationen schnell hintereinander ausgeführt werden sollen, viel Rechenzeit. Wenn bei einer hohen Anzahl von Zugriffen beispielsweise auf einen HTTPS-Server die CPU-Leistung für die aufwendigen Public-Key-Berechnungen nicht mehr ausreichend ist, um erträglich kurze Antwortzeiten sicherzustellen, ist der Einsatz von speziellen Krypto-Coprozessor-Boards im betroffenen Server (oder auch in einem CA-Rechner, falls der überlastet ist) möglich. Ein Beispiel für solche Geräte sind die Produkte aus der „nFast“-Krypto-Beschleuniger-Familie der Firma nCipher (siehe Anhang B.3). Sie können beispielsweise auch mit dem gängigen Web-Server *apache* eingesetzt werden und werden über eine SCSI-Schnittstelle angeschlossen.

Solche spezielle Krypto-Hardware hat einen Vorteil, der zugleich ein Nachteil sein kann: Sie ist meist *tamper resistant* aufgebaut, d.h. so versiegelt, daß eine Analyse ihres Inneren nicht möglich ist. Man ist daher gezwungen, sich auf die Zusicherungen des Herstellers über die kryptographischen Eigenschaften z.B. des in so einem Modul enthaltenen Schlüssel- und Zufallszahlengenerators zu verlassen. Bestenfalls können seine Ergebnisse ausgelesen und extern weiterverarbeitet werden, eine Gelegenheit zu einer gründlichen Analyse, wie sie z.B. bei reinen Software-Lösungen bei Vorliegen des Quellcodes möglich ist, hat man bei Krypto-Hardware in der Regel nicht.

4.20.4.4 Hardware-Zufallszahlengenerator

Die Erzeugung von Schlüsseln ist ein besonders sensibler Punkt bei allen kryptographischen Verfahren. Die Schlüssel sollen für einen Angreifer nicht vorhersag- oder -bestimmbar sein, müssen also möglichst echte Zufallszahlen sein. Echter Zufall ist nun aber etwas, das sich mittels Software mit ihren genau festgelegten Arbeitsschritten kaum erzeugen läßt.

Sicherheitsfachleute schlugen schon vor Jahren vor, einen physikalischen Zufallszahlengenerator, also Hardware, für diese Aufgabe zu verwenden:

> “Is there any hope for strong portable randomness in the future? There might be. All that’s needed is a physical source of unpredictable numbers.
>
> A thermal noise or radioactive decay source and a fast, free-running oscillator would do the trick directly [...]. This is a trivial amount of hardware, and could easily be included as a standard part of a computer system’s architecture. [...] All that’s needed is the common perception among computer vendors that this small additional hardware and the software to access it is necessary and useful.” [CES94]

Aber erst jetzt setzt mit Intel ein Hersteller von Massenmarkt-Computer-Hardware dies in die Tat um und realisiert einen solchen Zufallszahlengenerator in einem Produkt: Der neue Pentium-III-Prozessor enthält (neben der unter Datenschutzgesichtspunkten eher bedenklichen Seriennummer) auch einen Hardware-Zufallszahlengenerator [IW99, JK99].

Alternativ könnten aber auch Ansätze wie der in [HJJS98] erfolgreich sein, die auf üblicherweise vorhandener Hardware – hier: Festplatten – aufbauen und diese zur Gewinnung echter Zufallszahlen nutzen.

Ein kryptographisch starker Zufallszahlengenerator ist für eine CA vor allem bei der *Erzeugung* von Schlüsseln wichtig. Wenn die CA nicht gleichzeitig auch als Trustcenter arbeitet und für ihre Nutzer die Schlüsselpaare generiert, sondern nur als Zertifizierungsstelle auftritt, dann ist der Nutzen eines Hardware-Zufallszahlengenerators auf die seltenen Momente beschränkt, in denen die CA selbst ihre neuen Schlüssel generiert. Beim *Zertifizieren* von Schlüsseln werden keine Zufallszahlen benötigt.

Zufallszahlenerzeugung als „Werbegag“ für die CA bei Veranstaltungen Die Computerzeitschrift *c’t* bot auf der diesjährigen CeBIT nicht nur die Zertifizierung von PGP-Schlüsseln, sondern für alle neugierig Gewordenen, die ihren Schlüssel nicht dabei hatten oder noch gar nicht über einen solchen verfügen, gleichzeitig auch noch als besonderen Service die Schlüsselgenerierung an:

> „Allen, die noch keinen PGP-Schlüssel besitzen, bieten wir auf der Messe eine Möglichkeit zur Schlüsselgenerierung an. Der dabei eingesetzte DOS-Rechner bootet von CD-ROM und besitzt als einziges Speichermedium eine Diskette, auf der die Schlüsseldaten landen. Diese Diskette nehmen Sie mit nach Hause – auf dem System können keine Spuren Ihres geheimen PGP-Keys verbleiben.“ [Luc99a]

Eine werbewirksame Idee, die sich mit der Zufallszahlenerzeugung als Zutat für Schlüsselgenerierung zu einer PR-Aktion für die Low-Level UNI-CA kombinieren ließe: radioaktiver Zerfall, Höhenstrahlung o.ä. könnte als Rauschquelle mit entsprechendem abenteuerlich-auffallendem Geräte-Aufbau dienen. (Eventuell ließe sich die jeweilige Zufalls-Quelle danach wählen, in welchem Umfeld die CA gerade dargestellt werden soll; also z.B. eine physikalische Rauschquelle wie eine Diode, wenn Physiker angesprochen werden sollen, oder Radioaktivität, wenn es darum geht, Chemiker zu interessieren usw.) Die von diesem Generator gelieferte Bitfolge könnte dann auf Diskette gespeichert und anschließend dem Interessenten ausgehändigt werden. Oder sie würde, als Verfeinerung der o.g. *c’t*-Aktion, in einen „gläsernen Rechner“ geleitet, der vom CD-ROM bootet und unter Verwendung der Zufallszahlenfolge ein Schlüsselpaar generiert.

Der *gläserne* Rechner dient dabei mehr als Blickfang und dazu, den Leuten zeigen zu können, daß wirklich keine Festplatte o.ä. eingebaut ist, die den gerade erzeugten geheimen Schlüssel des Nutzers eventuell doch speichern könnte. – Das schützt allerdings trotzdem nicht vor einer schlechten oder gezielt manipulierten Implementierung der Schlüsselgenerierungs-Software, die dann vielleicht gar keine wirklich zufälligen Schlüssel erzeugt oder die den generierten geheimen Schlüssel versteckt in den Public-Key einbettet, so daß ihn der wissende Angreifer daraus später extrahieren kann [How97]. Insofern ist dieses Verfahren eher eine Attraktion, um einige Leute auf Verschlüsselung aufmerksam zu machen und sie dafür zu gewinnen. Höheren Sicherheitsanforderungen wird diese Variante der Schlüsselerzeugung nicht gerecht, noch zumal wo sie ja in aller Öffentlichkeit passiert und insofern einem Angreifer zusätzliche Möglichkeiten bietet.

Zu einem solchen Verfahren, bei dem Zufallszahlen verschiedener Qualität aus verschiedenen Quellen einfließen, schreibt FEDERRATH [Fed97]:

> „Das generierte Schlüsselpaar ist dabei mindestens so sicher wie der sicherste Zufallszahlenanteil. Es könnten also sowohl vom Teilnehmer als auch vom Trust Center Zufallszahlen in das

> Gerät eingegeben werden, wobei der beste Zufallsanteil die minimale Güte der Schlüsselerzeugung bestimmt.“

Das stärkste Glied in der Kette bestimmt also, wie stark das Ganze ist – ein sehr erfreulicher Umstand!

Das alles gilt natürlich unter dem Vorbehalt, daß die verwendete Software, die die Zufallszahlen-Eingaben als Ausgangsbasis für die Schlüsselerzeugung nutzt, zuverlässig arbeitet und diese Zufallszahlen nach dem „Stand der Kunst“ zu Schlüsseln weiterverarbeitet. Es wäre allerdings wünschenswert, daß Software, die Schlüssel generiert, zukünftig auch das „Zumischen“ von Zufallszahlen z.B. aus einer oder mehreren anderen Quellen, beispielsweise aus einer Datei auf einer Diskette, ermöglicht. Dann könnte eine CA statt der Schlüsselgenerierung die bloße Erzeugung von echten Zufallszahlen „zum Abfüllen“ und Mitnehmen anbieten. Diese könnten dann vom Anwender auf einem Datenträger mit nach Hause genommen und dort bei der nächsten Schlüsselerzeugung „beigemischt“ werden.

4.20.4.5 Biometrische Verfahren

Biometrische Verfahren zur Zugriffssicherung können auch in einer CA eingesetzt werden, z.B. um den Zugriff auf den geheimen Signierschlüssel zu kontrollieren. Das wäre eine andere denkbare Erweiterung und Verfeinerung der CA-Hardware-Ausstattung, die zugleich einen zusätzlichen Schutz vor unbefugtem Zugriff auf den Private-Key böte.

Der *Biomouse* Fingerabdruck-Scanner (siehe Anhang B.3) ist eines der wenigen biometrischen Geräte, die auch mit Treibern bzw. einem Developer Toolkit für Unix/Linux erhältlich sind, insofern könnte er für die UNI-CA interessant sein, wenn dort auch biometrische Verfahren zum Einsatz kommen sollen.

4.20.5 Neue Anwendungen im Zusammenhang mit Public-Key-Verfahren

Die Tätigkeit der UNI-CA muß nicht auf die Zertifizierung von Schlüsseln beschränkt bleiben. Es bietet sich vielmehr an, das dabei gewonnene Know-how auch im Rahmen der übrigen Aufgaben des Rechenzentrums einzusetzen. Eine konkrete Anwendung soll das verdeutlichen. Mit ihr werden sich Einrichtungen, die wie die das UNI-RZ mit Netzwerkverwaltung befaßt sind, ohnehin in nicht allzu ferner Zeit zu befassen haben:

DNSsec, die Sicherheits-Erweiterungen für den Domain Name Service (DNS) [Gil97, Eas99] schützen gegen bisher mögliche Angriffe auf den DNS wie z.B. DNS-Spoofing [MW97, Mar99], indem DNS-Einträgen ein neuer Eintragstyp hinzugefügt wird, der eine digitale Unterschrift zu den übrigen Einträgen enthält. Anhand dieser digitalen Signatur läßt sich dann leicht feststellen, ob der DNS-Eintrag in unverfälschter, authentischer Form übermittelt wurde und wirklich von dem für die betreffende Zone maßgeblichen Server stammt.

Zugleich bietet DNSsec auch ein alternatives Verteilmedium für Zertifikate in Form des entsprechend erweiterten DNS an, so daß dieser Standard auch aus diesem Grund für eine Zertifizierungsstelle interessant ist.

Bis vor kurzem fehlte es noch an der Anwendung von DNSsec in großem Umfang, weil die einzige verfügbare Version der verbreiteten DNS-Serversoftware BIND *mit DNSsec-Unterstützung* auf einer veralteten BIND-Version 4.x beruhte. Da mittlerweile BIND 8.x-Versionen mit DNSsec-Funktionalität verfügbar sind, wird die Zahl der DNS-Server mit DNSsec-Unterstützung nunmehr bald steigen.

Im September 1998 hat die US-Regierung einen entsprechenden Auftrag für den Schutz des DNS („kryptographische Authentifizierung“) an die Firma Network Associates vergeben [CZ98l]. NAI veranschlagt für die Entwicklung einen Zeitraum von etwa 18 Monaten. Wenn dieser Zeitplan halbwegs einhalten wird, dürfte man spätestens ab Ende 2000 mit einem signifikant steigenden Einsatz von DNSsec im Internet rechnen.

Kapitel 5

Praktische Umsetzung des CA-Konzeptes

Dieses Kapitel ist als Handreichung für die Vorbereitung und Umsetzung des im vorigen Kapitel beschriebenen Konzeptes für die UNI-CA gedacht. Es beschreibt zum Teil sehr konkret, dabei teilweise auch in stichpunktartiger Form oder in Form von Checklisten, wie dabei vorgegangen werden sollte.

5.1 Zertifizierungsplattform

5.1.1 Der Zertifizierungsrechner

Als Zertifizierungsrechner ist ein mobiler, tragbarer Rechner vorgesehen. Damit wird es der Low-Level UNI-CA möglich sein, flexibel vor Ort bei verschiedenen Gelegenheiten ihre Zertifizierungsdienste anbieten und demonstrieren zu können. Die Möglichkeit, eine physikalische Diebstahlssicherung anzubringen, mit der das Gerät vor Wegnahme geschützt werden kann, ist unbedingt vorzusehen, vor allem für externe Termine mit Publikumskontakt, da tragbare Computer einem erheblichen Diebstahlsrisiko ausgesetzt sind. 1996 wurden 265 000 tragbare Computer gestohlen (Zahlen von Safeware Insurance, nach [Per98]).

5.1.2 Betriebssystem

Der Zertifizierungsrechner der UNI-CA sollte unter dem Betriebssystem Linux betrieben werden und zwar aus folgenden Gründen, die für Linux sprechen:

- Linux läuft auf gängigen tragbaren Computern (wenn auch nicht auf allen)
- Linux ist „Open-Source“, eine Eigenschaft, die gerade bei sicherheitsrelevanten Anwendungen von Vorteil ist [Neu99]
- MICHAEL HANGE, Vizepräsident des Bundesamtes für Sicherheit in der Informationstechnik:

> „Vorteile [von Linux] sehen wir im höheren Stabilitätsgrad im Server-Bereich sowie im Bereich höherschutzbedürftiger Anwendungen. ... Generell sollte man die Transparenz, die das System zur Zeit hat, in Zukunft erhalten. Die freie Verfügbarkeit ist ein enormer Vorteil.“ [SH99, S. 218]

- Schwächen von Microsoft-Betriebssystemen bei Sicherheitsaspekten [CZ98h] und mangelnde Reaktion von Microsoft auf Hinweise auf Fehler in Sicherheitsfunktionen ihrer Produkte, wie von PETER GUTMANN in [Gut98] geschildert

Zusätzlich könnte eventuell ein verschlüsselndes Filesystem eingesetzt werden. CFS [Bla93] verschlüsselt nur mit DES, das mag besser sein als ein unverschlüsseltes Dateisystem, ist aber nicht mehr zeitgemäß, da DES inzwischen innerhalb weniger Tage oder sogar Stunden geknackt werden kann [EFF98].

TCFS[1] [CP, Mau97] bietet eine breitere Palette von Verschlüsselungsalgorithmen zur Auswahl. Je nach Leistungsfähigkeit der verwendeten CPU und Verfügbarkeit eines entsprechenden Krypto-Moduls für TCFS sollte IDEA als Verschlüsselungsverfahren der Vorzug vor 3DES gegeben werden. DES oder RC5 sollten nicht verwendet werden. Da TCFS über NFS angesprochen wird, müßte auf dem CA-Rechner auch ein NFS-Server laufen. Beide, CFS und TCFS, werden auch in [Sch98] dargestellt.

Die Integrität der Software und der Daten auf dem Zertifizierungsrechner sollte regelmäßig überprüft werden. Um dies zu gewährleisten, kann beispielsweise das bewährte Sicherheits-Tool *tripwire* [KS94] eingesetzt werden. Es bildet Prüfsummen nach verschiedenen kryptographischen Verfahren über den einzelnen Dateien und speichert diese für spätere Vergleichsläufe ab.

5.1.3 Zertifizierungssoftware

Als Zertifizierungssoftware für PGP-Schlüssel wird PGP 2.6.3**in** vorgeschlagen, eine PGP-Version, die speziell auf den Einsatz in Zertifizierungsumgebungen zugeschnitten ist und einige der – fehlerträchtigen und mühsamen – Plausibilitätsprüfungen automatisch vornimmt, die bei einer Zertfizierung durchgeführt werden sollten. Außerdem unterstützt diese Version bei Verwendung separater RSA-Schlüssel für das Signieren und Entschlüsseln die *automatische* Benutzung des jeweils vorgesehenen Schlüssels, so daß bei der verlangten Funktionstrennung (siehe 4.8.2) die Gefahr von Verwechslungen und Fehlbenutzungen reduziert wird.

PGP 2.6.3in ist außerdem die einzige PGP 2.x-Version, die Zertifikatwiderrufe unterstützt, insofern bleibt eigentlich kaum eine Wahl, zumal die Unix-Variante der neueren PGP-Versionen (PGP 5.0i, PGP 6.5.1i) aus den in 4.8.1 genannten Gründen (noch) nicht eingesetzt werden soll. Im Anhang I.2 werden weitere Gründe dafür aufgeführt, warum gerade *PGP2.6.3in* in der UNI-CA verwendet werden sollte.

[1] `http://mikonos.dia.unisa.it/tcfs`

5.1.4 Speichermedium für den geheimen Signierschlüssel

Die Policy sieht eine Speicherung des geheimen CA-Signierschlüssels auf einem Wechselmedium vor. Als Trägermedium für die Schlüsseldaten, insbesondere die Private-Keys der CA, ist eine Floppy-Disk bzw. ZIP-Disk vorgesehen.

Ein Speichern der geheimen Schlüssel auf einer CD-ROM hätte zwar den Vorteil, daß diese nicht mehr von einem Angreifer im Betrieb verändert werden könnten, dem stünden aber zwei Nachteile gegenüber: Zum einen wäre dann eventuell ein Akku-Betrieb des CA-Rechners nicht mehr möglich – manche CD-Laufwerke verbrauchen (zu)viel Strom –, zum anderen könnten die öffentlichen Schlüssel der CA nicht mehr auf demselben Medium gespeichert werden, da zumindest bei PGP auch neu zertifizierte Schlüssel dem Public-Keyring hinzugefügt werden.

Eventuell ist es sogar möglich, einen Boot-Loader bzw. ein „Mini-Filesystem“ auf dem Schlüsselmedium unterzubringen und dann ein *verschlüsseltes* Filesystem von der Festplatte selbst zu booten. Die Festplatte enthielte dann ausschließlich ein verschlüsseltes Filesystem, das ohne das zugehörige Paßwort und ohne die entsprechende Boot-Diskette nicht sinnvoll angesprochen werden könnte.

Sofern es Treiberunterstützung für PCMCIA-Memory-Cards unter Linux gibt, die die entsprechenden Schnittstellen des Laptop ansteuern können, könnte auch eine solche Memory-Card als Speichermedium für den oder die geheimen Schlüssel in Erwägung gezogen werden. Der Vorteil dieser Lösung bestünde darin, daß ein potentieller Angreifer schon ein PCMCIA-Karten-taugliches Gerät mit sich führen müßte, um schnell und unbemerkt den Schlüssel von der Memory-Card zu kopieren. Außerdem bliebe das Floppy-Laufwerk des Rechners für den Datenträgeraustausch frei, und es müßte nicht zusätzlich ein ZIP-Laufwerk vorhanden sein.

Die Diskette mit den geheimen Schlüsseln sollte mit einem langen Farbstreifen, der im Betrieb aus dem Floppy-Laufwerk heraushängt, auffällig markiert werden, ähnlich etwa einem Werkstatt-Schlüssel, damit sie sofort auffällt, wenn sie sich im Laufwerk befindet, und dort nicht vergessen wird, und damit immer deutlich ist, daß dies kein x-beliebiger Datenträger ist.

5.2 Vorgehen bei Einrichtung der CA (*“build”*)

5.2.1 Auswahl und Beschaffung der Hardware

Da der CA-Rechner unter dem Betriebssystem Linux betrieben werden soll, ist bei der Auswahl der Hardware darauf zu achten, daß ein System angeschafft wird, dessen Komponenten von Linux unterstützt werden. Der Grafik-Chipsatz des Laptops muß von Linux unterstützt werden bzw. es sollte ein Rechner mit einem Chipsatz ausgewählt werden, für den es im Sourcecode verfügbare Grafiktreiber (X-Server) unter Linux gibt. Ebenfalls kritisch bei Linux auf tragbaren Rechnern sind der Treiber-Support für den Soundchip und für den CardBus-Controller [DK98].

Sowohl Intel-basierte Laptops als auch Apple PowerBooks sind grundsätzlich geeignet, denn für beide ist Linux verfügbar.[2] Ein Vergleich von Apple-PowerBooks und Notebook-PCs findet sich in [MR98].

[2] Linux für PowerPC: `http://www.linuxppc.org`, `http://www.mklinux.org`, beide besprochen in [Wil98]

Vor dem Kauf sollten verfügbare Quellen über den Einsatz von Linux auf tragbaren Computern wie z.B. die *Linux on Laptops*-Web-Seite von KENNETH E. HARKER (siehe Anhang A.2) konsultiert werden, da die Installation nicht trivial ist und nur manche der Geräte und Chipsätze unterstützt werden.[3]

Da das Arbeitskonzept zwei separate Festplatten vorsieht, damit die Low-Level- und die Medium-Level UNI-CA physikalisch getrennt werden können, darf beim Rechnerkauf ein **zweiter Festplatteneinschub nebst Festplatte** nicht vergessen werden. Beide Festplatten sind dabei ausreichend groß zu dimensionieren – der PGP Public-Keyring des internationalen PGP-Keyserver-Verbundes ist bereits heute größer als 500 MB (die Datei *pubring.pgp* auf ftp.pgp.net war am 21. Februar 1999 542 MB groß), und die entsprechende Datenbank der Keyserver-Software [Hor96] nimmt nach Aussagen der DFN-PCA sogar bereits rund 2 GB Plattenplatz ein. Bei einer angenommenen durchschnittlichen Größe eines PGP-Public-Keys von 20 KB wäre der angenommene Schlüsselring mit den öffentlichen Schlüsseln von 25 000 Rechenzentrums-Nutzern schon 500 MB groß (und PGP legt während der Arbeit eine Zwischenkopie des gesamten Schlüsselbundes an), daher sollte hier der Festplattenplatz besser nicht zu knapp bemessen werden.

Der **Festplatten-Einschub** des Rechners muß leicht zugänglich sein; die Festplatte soll ohne Schraub-Arbeiten am Rechner ausgewechselt werden können (damit die Festplatte für den Low-Security-CA-Betrieb leicht statt derjenigen für die Medium-Security-CA eingesetzt werden kann, wenn der CA-Rechner zu einem Termin außerhalb des Rechenzentrums mitgenommen werden soll). Andererseits sollte der Ausbau nicht in wenigen Sekunden durchgeführt werden können, da sonst bei einem CA-Einsatz außer Haus die Gefahr zu groß ist, daß ein Angreifer in einem unbeobachteten Moment „zugreift“ – eventuell schützt hier auch die Diebstahlsicherung (s.u.).

Der Laptop sollte über ein ***internes* ZIP-Laufwerk** verfügen – alternativ kommt stattdessen auch ein LS-120- oder HiFD-Laufwerk in Frage, sofern es dafür ebenfalls Treiberunterstützung unter Linux gibt – das ggf. anstelle eines internen CD-ROMs installiert werden könnte und sollte. Auf diesem Medium sollen die geheimen Schlüssel und die Public-Keyrings abgelegt werden.

Das Gerät *muß* über ein **internes Floppy-Laufwerk** verfügen, damit ein Wechselmedium zur Verfügung steht, um Schlüssel entgegennehmen bzw. Zertifikate auf die Disketten von Nutzern schreiben zu können. Ein externes Floppy-Laufwerk ist dafür nur begrenzt geeignet, weil damit die Portabilität erheblich reduziert und insofern der Vorteil des tragbaren Rechners (fast) wieder aufgehoben würde. ZIP- und Diskettenlaufwerk sollten auch bei Akku-Betrieb gleichzeitig benutzt werden können (bei manchen Modellen ist eine gleichzeitige Benutzung nur bei Netzbetrieb möglich), damit nicht ständig im einzigen Laufwerk ein Medium zum Datenaustausch mit gegen das Schlüsselträger-Medium (u.U.) ausgetauscht werden muß.

Als **CD-ROM-Laufwerk** kann auch ein *externes* Laufwerk mit passendem Interface eingesetzt werden; eventuell braucht auch kein CD-Laufwerk extra für den CA-Rechner angeschafft zu werden und zwar dann, wenn ein solches Laufwerk bereits im Rechenzentrum vorhanden ist und am SCSI-Anschluß des CA-Rechners – PCMCIA-Adapter oder Schnittstelle an einer eventuellen Docking-Station – betrieben werden kann.

[3] Einen ersten Eindruck von den Hürden, die es bei der Installation von Linux auf einem Notebook zu überwinden gilt, gibt [DK98].

Im Interesse der Zukunftssicherheit und des Investitionsschutzes sollte der Laptop auch über einen USB-Anschluß verfügen, über den sich ggf. weitere Peripheriegeräte (Drucker, weiteres CD- oder Diskettenlaufwerk, Biometrie-Hardware, Kartenleser etc.) anschließen läßt.

Zubehör

Eine **SCSI-Adapterkarte** zum Anschluß von SCSI-Bandlaufwerken für die Datensicherung wird benötigt, es sei denn, es wird auch eine Docking-Station mit SCSI-Anschluß gekauft; dann kann der separate SCSI-Adapter entfallen.

Die Anschaffung eines **tragbaren Druckers** *mit passender Schnittstelle zum Anschluß an den Laptop* – ein Netzwerkdrucker kann bzw. darf ja nicht benutzt werden! – könnte lohnenswert sein. Es sollte dabei *kein* Thermo-Drucker gewählt werden, denn deren Ausdrucke sind nicht dokumentenecht, was nachteilig wäre, wenn im „Außeneinsatz“ der CA Zertifizierungsanträge damit ausgedruckt würden und diese nach einiger Zeit verblaßten. Auch hier gilt es wieder, auf die erforderliche Treiberunterstützung unter Linux sowohl für die Anschlußschnittstelle als auch für das Druckerfabrikat zu achten. (Ein *USB*-Drucker hätte ggf. den Vorteil, daß der Druckerport am Laptop beispielsweise für einen externen Hardware-Zufallszahlengenerator frei bliebe, denn manche dieser Geräte werden am Parallelport betrieben.) Ein möglichst leichter (portabler) Drucker wäre für die Außentermine der CA natürlich vorteilhaft.

Lithium-Ionen-Akkus (Li-Ionen) sind vorzuziehen, weil sie leichter sind, bei gleicher Größe eine höhere Energiedichte aufweisen als herkömmliche NiMH-Akkumulatoren und darüber hinaus nicht zum bei NiCd-Akkus gefürchteten „Memory-Effekt“ neigen. Nach Möglichkeit sollten daher Li-Ionen-Akkus beschafft werden, und zwar zwei Stück, da die Laufzeit eines Laptop-Rechners mit einer Akku-Ladung meist nur rund drei Stunden beträgt. Das könnte dann „außer Haus“ zu Problemen führen, wenn die CA auf einer Veranstaltung präsent ist und währenddessen, womöglich in einem Moment besonders großer Nachfrage, der Akku leer wird. Wenn auf einem solchen *Event* die Möglichkeit besteht eine Steckdose zu benutzen – umso besser, aber das wird vielleicht nicht immer der Fall sein, zum Beispiel dann nicht, wenn die Stände im Freien aufgebaut sind (Sommerfest).

Die Anschaffung von gleich **zwei Akkus** erscheint auch aus einem anderen Grund sinnvoll: Die Innovationszyklen und Produkt-Lebensdauern gerade bei Laptops sind so kurz, daß u.U. etwas längere Zeit nach dem Kauf bereits keine passenden Ersatzteile oder Ersatz-Akkus mehr für das „veraltete“ Gerät erhältlich sind. Diese Gefahr droht insbesondere bei No-Name-Fabrikaten, weshalb Markengeräten der Vorzug gegeben werden sollte. (Alternativ kann man sich auch vom Händler den Mindestzeitraum für die Ersatzteilversorgung schriftlich garantieren lassen.)

Es sollte gleich beim Rechnerkauf eine **Diebstahlsicherung** passend zum Laptop erworben werden (Stahlseil mit Schloß und Befestigungsmöglichkeit am Rechner), damit der CA-Rechner beim Außer-Haus-Einsatz gegen Wegnahme gesichert werden kann.

Eventuell könnte eine sog. **Docking-Station** für den Laptop eine sinnvolle Anschaffung sein (sofern für das gewählte Rechnermodell verfügbar). Sie würde es z.B. ermöglichen, einen externen Monitor anzuschließen, falls im Rechenzentrum am CA-Rechner gearbeitet werden soll und das kleine LC-Display als störend empfunden wird. Außerdem bieten manche Docking-Stations einen eingebauten

SCSI-Adapter, so daß auf diese Weise die Datensicherung einfacher durchgeführt werden könnte bzw. keine zusätzliche SCSI-Controller-Karte mehr benötigt würde.

Von einigen Herstellern werden inzwischen besonders **robuste, stoß- und spritzwasserunempfindliche Laptops** angeboten (siehe Anhang B.2) – wenn also ein Gerät, das die übrigen Anforderungen erfüllt, zusätzlich auch noch besonders *stabil* ist, so daß es den harten Einsatz an einem von Menschen umlagerten CA-Stand und den Transport zwischen Veranstaltungsort und Rechenzentrum sowie im Rechenzentrum selbst besser überstehen kann, wäre das ein zusätzlicher Pluspunkt.

Zuzusichernde Eigenschaften

Beim Kauf sollte der Verkäufer die Linux-Tauglichkeit des Laptops und des Zubehörs schriftlich garantieren oder ein Rückgaberecht für den Fall der Nicht-Tauglichkeit einräumen.

5.2.2 Beschaffung der Software

Das Betriebssystem wird auf einer CD-ROM oder ZIP-Disk als Installationsmedium benötigt. Da der CA-Rechner nicht (auch nicht bei der Erst-Installation) vernetzt betrieben werden darf, müssen alle erforderlichen Programme und Daten per Wechselmedium oder CD-ROM auf den Laptop gebracht werden.[4] Es wird also bei der Installation zusätzlich ein zweiter Rechner mit Internet-Zugang und ZIP-Laufwerk (LS-120-/HiFD-Laufwerk) benötigt, sofern nicht alle benötigte Software bereits auf CD-ROM(s) vorliegt. Dieser Rechner muß ZIP-Medien in dem Dateisystem-Format beschreiben können, das der CA-Rechner bzw. dessen Betriebssystem lesen kann.

Es werden zusätzlich mindestens folgende Programme benötigt (die eventuell nicht auf der Installations-CD-ROM oder darauf nicht in der betreffenden Version enthalten sind):

Tripwire zur Integritätskontrolle für den CA-Rechner

PGP PGP2.6.3in für die PGP-Zertifizierung

OpenSSL für die Ausstellung von X.509-Zertifikaten (Nachfolger von *SSLeay*, siehe Teil II dieses Handbuches)

Bezugsquellen für alle drei Pakete finden sich im Anhang B.1.

Alle drei Programme sind von ihren Autoren PGP-signiert worden (bei Tripwire versteckt sich die PGP-Signatur in dem .tar-Archiv). Auf dem vernetzten Rechner sollte als erstes geprüft werden, ob diese Signaturen sich verifizieren lassen, d.h. ob die Software-Archive in unverfälschter Form vorliegen. Dies geschieht durch Aufruf von ‘pgp *FILENAME*.asc’. (Diese Prüfung ist sinnvoll nur möglich, wenn der öffentliche Schlüssel des jeweiligen Autors bereits in authentischer Form vorliegt.)

[4] Das mag im ersten Moment sehr streng erscheinen, dient aber letztlich dazu, eine genau definierte, reproduzierbare Software-Umgebung auf dem CA-Rechner zu schaffen. Bei einer Installation über eine Netzwerkverbindung könnte man nie sicher sein, *was* letztlich genau an Daten auf den CA-Rechner gelangt.

5.2.3 Hardware-Vorbereitungen

Viele aktuelle Laptop-Rechner verfügen heute über eine Infrarot-Schnittstelle. Da der CA-Rechner nicht auf diesem Weg vernetzt werden soll und mittels dieses Kanals nicht angreifbar sein darf, ist die IR-Schnittstelle lichtundurchlässig, aber reversibel abzukleben (reversibel für den Fall, daß der Rechner später einmal für andere Zwecke genutzt werden soll).

Die Empfehlungen z.B. des Berliner Datenschutzbeauftragten zum Einsatz von Laptops [DSB92] sollten ebenfalls, soweit sinnvoll anwendbar, berücksichtigt werden, gleiches gilt für die entsprechenden bzw. in 4.7 genannten Bausteine des BSI-Grundschutzhandbuches.

Insbesondere sollte

- das BIOS-Paßwort gesetzt werden
- ein eventuell vorhandener BIOS-Schutz von Master- und System-Bootsektor aktiviert werden (so verfügbar)
- ein Screen-Locker aktiviert werden (wenn vom BIOS unterstützt)
- der Bootvorgang bzw. Single-user-Boot Paßwort-geschützt werden
- das Booten vom Floppy-Laufwerk sollte abgeschaltet werden (ggf. auch erst direkt nach der Installation, falls nicht von CD-ROM gebootet werden kann und eine Boot-Disk den Installationsvorgang starten muß)

5.2.4 Installation des Betriebssystems

Bei der Installation von Linux wird eine Partitionierung der Festplatte erforderlich, dabei muß eingeplant werden, daß Keyserver-Datenbank-Dateien mit mehr als 2 GB Größe auftreten können. Netzwerkunterstützung muß deaktiviert werden (NFS ggf. aktiviert lassen, falls mit TCFS gearbeitet werden soll – siehe 5.1.2), Treiber für Netzwerkkarten und Infrarot-Schnittstelle sind ganz zu entfernen bzw. gar nicht erst einzubinden. Es sollte mit einem „minimalen System“ gearbeitet werden. (Es werden allerdings die Entwickler-Tools wie *gcc* usw. benötigt, da die CA-Software – pgp, openssl – im Quellcode vorliegt und erst auf dem CA-Rechner übersetzt werden muß. Diese Übersetzung darf auf keinem anderen Rechner vorgenommen werden.)

Die erforderlichen Treiber für ZIP-Laufwerk, CD-ROM, SCSI-Adapter, Bandlaufwerke (Datensicherung) dürfen nicht vergessen werden und sind mitzuinstallieren, darüber hinaus auch eventuell existierende Jahr-2000-Patches.

Der Rechner sollte so konfiguriert werden, daß nach dem Hochfahren deutlich zu erkennen ist, ob die Festplatte der Low-Level- oder die der Medium-Level UNI-CA eingebaut ist, der Rechner also gerade der Low-Level- oder der Medium-Level-Zertifizierungsrechner ist.

5.2.5 Tests vor der Inbetriebnahme

Vor der richtigen Inbetriebnahme des Zertifizierungsrechners sind Datensicherungs-Tests durchführen:

- Läßt sich der Streamer ansprechen?
- Lassen sich Dateien aus Backups wiederherstellen?
- Wie kann/muß gegebenenfalls nach einem Totalausfall der Festplatte das komplette System aus den Backups neu installiert werden?

5.2.6 Einrichten der Benutzerbereiche

Es sollten separate Benutzerbereiche für die PGP- und die X.509-Zertifizierung vorgesehen werden, damit diese Aufgaben ggf. von verschiedenen Mitarbeitern wahrgenommen werden können und ihre jeweiligen Zugriffsmöglichkeiten dann auf den ihnen zugewiesenen Aufgabenbereich beschränkt sind. Für alle Benutzerbereiche sollten ein Auto-Logout bei Leerlauf (*idled* oder ein entsprechendes Feature der jeweiligen Shell) und ein ebenfalls zeitgesteuerter Bildschirmschoner und *Screen-Locker* konfiguriert werden.

5.2.7 Installation der CA-Tools

Als erstes der drei oben unter 5.2.2 genannten Programme sollte Tripwire auf dem Laptop installiert werden – *vor* PGP und SSLeay, damit eventuelle Veränderungen an der Ausgangsinstallation, die durch diese Programme hervorgerufen werden könnten, bereits detektiert werden. Vor der Installation von PGP und SSLeay/OpenSSL auf dem CA-Rechner sollte also nicht nur die Installation, sondern auch die Konfiguration von *tripwire* und die Initialisierung von dessen Datenbank abgeschlossen sein. Die Installation von OpenSSL selbst ist ausführlich in Teil II dieses Handbuches ab S. 133 beschrieben, die von PGP kann z.B. dem DFN-CERT Informations-Bulletin DIB-94:05[5] entnommen werden (einschließlich eines wichtigen Hinweises zur Installation von PGP 6.5.1i).

5.2.8 Schlüsselerzeugung für die CA

Bereits vor der Erzeugung des oder der geheimen CA-Schlüssel sollte der dafür vorgesehene Datenträger (ZIP-Disk, Floppy) mit einem langen, signalfarbenen Papierstreifen, ähnlich einem Anhänger an einem Werkstattschlüssel, markiert worden sein, damit dieses Speichermedium überall sofort erkannt wird und auffällt, falls es doch einmal versehentlich unbenutzt herumliegen sollte und nicht ordnungsgemäß weggeschlossen verwahrt wird.

Bei der Schlüsselerzeugung muß die Mailadresse der CA als Teil der Benutzerkennung (bzw. des Distinguished Names bei X.509-Zertifikaten) angegeben werden, d.h. sie muß zu diesem Zeitpunkt feststehen und sollte auch schon benutzt werden können. An diese Benutzerkennung werden in der Low-Level DFN-Policy einige Anforderungen gestellt, die von der gewählten Kennung für die CA-Schlüssel erfüllt werden müssen. Hinweise dazu gibt der *PGP: Leitfaden zur CA-Zertifizierung*[6] der DFN-PCA; im Zweifelsfalle sollte vor einer Schlüsselerzeugung Rücksprache mit der DFN-PCA gehalten werden, da diese letztendlich die CA und ihre Schlüssel „absegnen“ muß.

[5] `http://www.cert.dfn.de/infoserv/dib/dib-9405.html`
[6] `http://www.pca.dfn.de/dfnpca/certify/pgp/instca.html`

Wenn dann die CA-Schlüssel generiert werden und z.B. PGP um Tastendrücke bittet, um seinen Zufallszahlen-Pool für die Schlüsselerzeugung zu füllen, dann sollte besonders darauf geachtet werden, *zufällige* Eingaben zu machen, denn die sind nach [Lin98] unter Unix die einzige wirkliche Zufallszahlen-Quelle bei der Schlüsselerzeugung. Alle anderen vermeintlichen Zufallsquellen – *system calls* von **times()** und **gettimeofday()** – liefern recht gut vorhersagbare oder im Nachhinein für einen Angreifer eingrenzbare Werte.

Die dann erforderliche Festlegung der *Passphrase* sollte unter Berücksichtigung des *PGP passphrase FAQs* [Wil97] bzw. der Tips von RALF SENDEREK [Sen98] zur Wahl einer guten Passphrase erfolgen. Falls das vorgeschlagene Vier-Augen-Prinzip für den Zugriff auf den Medium-Level UNI-CA-Signierschlüssel realisiert werden soll, müssen beide involvierten CA-Mitarbeiter getrennt voneinander ihren Teil der Passphrase festlegen, ohne daß der jeweils andere sehen kann, was eingeben wird.

Wie wichtig eine gute Passphrase ist, belegt der Vorfall mit dem Caligula-Word-Makrovirus, das auf befallenen Rechnern nach einem eventuell vorhandenen PGP-Secret-Keyring sucht und diesen per FTP an einen anderen Rechner übermittelt (wo vermutlich versucht wurde bzw. wird, die Passphrase zu knacken, die den Private-Key schützt) [CZ99a].

Der geheime Kommunikationsschlüssel der CA ist sodann an alle CA-Mitarbeiter unter Mitteilung der ihn schützenden Passphrase zu verteilen. Falls dafür eine initiale, einfach zu merkende/ratende Passphrase benutzt wurde, sind alle Betroffenen nochmals darauf hinzuweisen, daß sie umgehend eine eigene, nicht-triviale Passphrase für den Schlüssel setzen müssen.

5.2.9 Vorbereitung der CA-Mitarbeiter

5.2.9.1 Zu verwendendes PGP-Nachrichtenformat

Diejenigen Rechenzentrumsmitarbeiter, die CA-Aufgaben wahrnehmen sollen, müssen darauf hingewiesen werden, welche Form PGP-Mails der UNI-CA haben sollen. Es muß eine Entscheidung getroffen werden zwischen dem „klassischen“ Format [ASZ96], bei dem die Nachrichten üblicherweise als MIME-Typ *text/plain* verschickt werden, und einem etwas neueren Format [Elk96], das den Vorteil bietet, daß die Content-Types der MIME-Bodyparts der E-Mail als *multipart/encrypted; protocol="application/pgp-encrypted"* respektive als *multipart/signed; protocol="application/pgp-signature"* gekennzeichnet sind (was die automatische Erkennung und Verarbeitung durch das Mailprogramm erleichtert), das aber nicht von allen Mail-Useragents erzeugt und erkannt wird.

Die Entscheidung für das eine oder andere Format dürfte auch stark von den Vorlieben der einzelnen Beteiligten hinsichtlich ihres bevorzugten Mail-UAs abhängen: *mutt* unterstützt beispielsweise ausschließlich das neuere MIME-PGP-Format nach RFC 2015, andere Mail-Programme nur das alte (so beispielsweise manche *elm*-Versionen oder Skripte für *pine*).

Was bei dieser Entscheidung nicht unberücksichtigt bleiben sollte, sind die Kommunikationspartner, die mit dem gewählten Format klarkommen müssen. Da Prognosen darüber, ob das MIME-PGP-Format mehr oder weniger Schwierigkeiten und Nachfragen nach sich ziehen würde, nur schwer möglich sind, kommt es hier wohl wirklich auf die Erfahrungen in der Praxis an – falls die meisten der Zertifizierungsinteressierten *mutt* benutzen, ist das MIME-Format möglicherweise unproblematisch; bei überwiegend anderen Vorlieben könnte es damit hingegen zu vermehrten Nachfragen von seiten der Nutzer kommen, die mit der Zertifizierungsstelle per E-Mail korrespondieren. Hier droht also u.U. vermeidbare Mehrarbeit durch eine ungeschickte Wahl des Formates.

5.2.9.2 Zu verwendende Mailadressen

Damit die gesamte Mail-Kommunikation der Zertifizierungsstelle für alle Mitarbeiter nachvollziehund lesbar bleibt, sollte jede E-Mail, die ein CA-Mitarbeiter in dieser Funktion verschickt, als Kopie an die Mailadresse der Zertifizierungsstelle gesandt und bei vertraulichen Nachrichten diese auch an die CA mit-verschlüsselt werden. So ist zum einen sichergestellt, daß dort alle ausgehenden „CA-Mails“ dokumentiert sind, zum anderen, daß diese E-Mails – auch die verschlüsselt verschickten – für alle CA-Mitarbeiter lesbar bleiben, da diese ja alle über den geheimen Kommunikationsschlüssel der CA verfügen.

5.2.10 Vorbereitung der Rechenzentrums-Mitarbeiter (Schulung)

Damit die Rechenzentrumsmitarbeiter mit gutem Beispiel vorangehen können (vgl. 4.16), sollte ihnen ein kleiner zeitlicher „Vorsprung“, eine Einarbeitungszeit gegenüber den normalen Nutzern verschafft werden, indem sie bereits einige Zeit vor der offiziellen Ankündigung der UNI-CA eine Einweisung in die Benutzung von PGP und ggf. anderer Public-Key-Software erhalten. Das gibt ihnen die Chance, sich bereits mit dem Programm vertraut zu machen, bevor eventuell Anfragen dazu an sie gerichtet werden. Außerdem können die Betreffenden dann in der Testphase (siehe 5.2.12) besser als „Versuchskaninchen“ fungieren.

Es sollten vorsorglich *alle* Rechenzentrumsmitarbeiter und nicht nur die CA-Administratoren auf die Bedeutung des CA-Rechners und des markierten Datenträgers mit den geheimen CA-Schlüsseln hingewiesen werden.

5.2.11 Einbindung in die Rechenzentrums-Infrastruktur

5.2.11.1 Technische Infrastruktur

Damit mit der Betriebsaufnahme der Zertifizierungsstelle die Nutzer der Rechnerarbeitsplätze im Rechenzentrum und der Dialup-Zugänge die auf diese Weise propagierte Verschlüsselung auch anwenden können, sollten die entsprechenden Vorkehrungen hinsichtlich der vom Rechenzentrum zur Verfügung gestellten Software getroffen werden. Gängige Mail-*Useragents* wie *mutt*, *elm*, *pine* oder *exmh* sollten in einer aktuellen Version vorliegen, die eine möglichst einfache Integration von PGP (bzw. anderer Verschlüsselungssoftware) zuläßt. Insbesondere sollte das Mailprogramm, das in der Standard-Arbeitsumgebung vorgesehen oder vorgegeben ist, möglichst über entsprechende Funktionalität verfügen bzw. dahingehend erweitert oder aktualisiert werden. Es sind aber nicht nur Mail-Programme, die verschlüsselungstauglich gemacht werden sollten; ebenfalls einbezogen werden sollten Newsreader, da auch im Usenet PGP eingesetzt wird – dort vornehmlich, um Beiträge zu signieren und dadurch den Absender nachprüfbar zu machen und sich vor Fälschungen zu schützen. Gegebenenfalls sind entsprechend neue Versionen dieser Software und der zugehörigen

Plugins oder Skripte zu installieren, die eventuell für die Integration der Verschlüsselungsfunktionalität notwendig sind.

5.2.11.2 Organisatorische Vorbereitungen

Es muß eine Möglichkeit geschaffen werden, die Zertifizierungsanträge vor unbefugtem Zugriff und unbefugter Einsicht geschützt zu verwahren. Sie sind vertraulich zu handhaben, da sie personenbezogene Daten enthalten, und sie müssen vor unbefugten Änderungen, vor der Wegnahme und vor dem unbefugten Hinzufügen von Anträgen geschützt sein. Anderenfalls könnte ein Angreifer sich eine unzulässige Zertifizierung erschleichen, indem er einen Antrag mit falschen Angaben unter die richtigen Anträge mischt.

Nach erfolgter Zertifizierung müssen die Anträge weiterhin gesichert aufbewahrt werden, da ein Teil der darin enthaltenen Daten weiterhin vertraulich bleiben muß (Personalausweisnummer) und die Anträge zu Revisionszwecken gebraucht werden könnten, wenn es Zweifel an der Richtigkeit einer Zertifizierung gibt. Außerdem werden sie benötigt, wenn im Falle einer CA-Key-Kompromittierung alle mit dem kompromittierten Schlüssel signierten Zertifikate durch neue ersetzt werden müssen.

Neben den Arbeitsabläufen ist auch sicherzustellen, daß regelmäßig die Mailingliste der DFN-Zertifizierungshierarchie (s. Anhang A.3) verfolgt wird. Diese Aufgabe muß einem oder mehreren CA-Mitarbeitern übertragen werden, und die Liste ist entsprechend zu subskribieren.

5.2.11.3 Materialien

Zertifizierungsanträge Es müssen Zertifizierungsanträge in ausreichender Zahl vervielfältigt werden, so daß sie für die Zertifizierungssprechstunde, externe Auftritte der Low-Level UNI-CA oder andere Gelegenheiten, bei denen Anträge auf Schlüsselzertifizierung gestellt werden können, vorrätig sind.

Die Antragsformulare sollten auch auf den Web-Seiten der UNI-CA angeboten werden, damit Zertifizierungswillige sich diese vorab ausdrucken und ausfüllen können. Auch auf dem Low-Level-CA-Rechner sollten sie in elektronischer Form vorgehalten werden, damit vor Ort Anträge ausgedruckt werden können, falls die Vorräte schneller als erwartet aufgebraucht sind.

Überlegenswert ist, die Antragsformulare nur als PDF-Datei anzubieten, damit es nicht ganz so einfach ist, einzelne Passagen darin vor dem Ausdrucken zu ändern oder wegzulassen; anderenfalls müßten die CA-Mitarbeiter, die die Anträge entgegennehmen, jeden einzelnen dahingehend prüfen, ob nicht etwas Wichtiges darauf weggelassen wurde.

Da von der Zertifizierungsstelle personenbezogene Daten erhoben und verarbeitet werden, ist es empfehlenswert, den betrieblichen Datenschutzbeauftragten zu informieren und eventuell auch vorab zu konsultieren, so daß dieser z.B. einen prüfenden Blick auf den Vordruck für den Zertifizierungsantrag werfen kann.

Schlüsselinformationen Die Zertifizierungsstelle muß für die Zertifikatnehmer auch die Angaben zu ihrem Signierschlüssel und dem der DFN-PCA (oder Kopien der Schlüssel selbst) bereit halten, um sie ihnen aushändigen zu können. Diese Informationen bilden die Grundlage für die spätere Nutzung der von der CA ausgestellten Zertifikate. Ihre Authentizität ist daher von höchster Wichtigkeit; es muß unter allen Umständen vermieden werden, daß hier ein Angreifer seinen Schlüssel oder seine Schlüsselinformationen gegen die der UNI-CA austauscht!

Die Zettel oder Datenträger mit den Schlüsseln oder Schlüsselinformationen müssen daher vor unbefugtem Zugriff sicher verwahrt werden. Andererseits müssen sie zugänglich sein, wenn ein Zertifizierungswilliger erscheint und einen Zertifizierungsantrag stellt, damit sie ihm dann ausgehändigt werden können.

5.2.12 Interne Probeläufe

Zur Einübung, zur Optimierung der Abläufe und damit eventuelle Fehler noch keine dramatischen Folgen für das Ansehen der CA haben, sollten die Arbeitsabläufe bei allen Phasen der Zertifizierung und der sonstigen CA-Arbeit realitätsnah durchgespielt werden.

Hierzu könnten die Schlüssel der RZ-Mitarbeiter bzw. die Schlüssel einiger PGP-Interessierter probezertifiziert werden, mit einem ausdrücklich als „Test-Key“ gekennzeichneten CA-Schlüssel (oder einem Schlüssel mit einem Decknamen) und einer kurzen Gültigkeitsdauer. Die Gültigkeitsdauer sollte so bemessen sein, daß auch Aspekte wie ein Schlüsselwechsel der CA im Rahmen dieser Tests simuliert werden können.

Erprobt werden muß auch die Vorgehensweise bei der Sperrung eines Zertifikates, damit hier später im Ernstfall keine Fehler unterlaufen. Ebenfalls sollte auch die Re-Zertifizierung von Keys durchexerziert werden.

Eventuell in dieser Phase zutage getretene Schwächen oder Fehler in der Policy können zu dieser Zeit noch relativ leicht durch Korrektur derselben behoben werden.

5.2.13 Policy-Verabschiedung und -Bekanntgabe

Es ist eine eindeutige Regelung zu treffen, wer die UNI-CA-Policy letztlich abzusegnen hat oder zu Änderungen befugt ist. Diese Regelung sollte, eventuell auch als Information in der Policy selbst, bekanntgemacht werden.

Die Policy sollte vor ihrer Verabschiedung unbedingt auch mit der DFN-PCA abgestimmt sein, damit es nicht im Nachhinein zu Problemen mit der Einbindung der UNI-CA in die DFN-Zertifizierungshierarchie kommt! Auch bei Änderungen bzw. vor der Verabschiedung einer geänderten Fassung der Policy sollte vorsorglich immer die DFN-PCA beteiligt werden.

Zu Beginn der Zertifizierungstätigkeit der UNI-CA könnte die Policy erst einmal als „vorläufig“ und in der Erprobung deklariert und relativ kurz befristet werden. Wenn sich dann nach den ersten Wochen oder Monaten der Arbeit nach dieser Policy gezeigt hat, ob sie noch Schwachpunkte enthält, die korrigiert werden müßten, oder ob sie so als endgültige Policy verwendet werden kann, dann kann die endgültige Fassung in Kraft gesetzt werden.

5.2.14 Zertifizierung durch die DFN-PCA

Nachdem es nun eine – hoffentlich mit der DFN-PCA abgestimmte – UNI-CA Policy und Schlüsselpaare zum Signieren und für die Kommunikation mit der UNI-CA gibt, ist die Zeit für die Zertifizierung durch die DFN-PCA gekommen. Dafür ist der persönliche Kontakt zwischen dem von seiner Einrichtung authorisierten Ansprechpartner der UNI-CA mit einem DFN-PCA-Mitarbeiter erforderlich. Es muß also entweder jemand nach Hamburg zur DFN-PCA reisen, oder der persönliche Kontakt erfolgt bei am Rande einer Tagung oder Konferenz, auf der Vertreter beider Einrichtungen zugegen sind (geeignet wäre z.B. die DFN-Betriebstagung, die üblicherweise in Berlin stattfindet und auf der meistens ein Vertreter der DFN-PCA anwesend ist).

Zuvor muß allerdings noch ein Schlüsselwiderruf (Key Revocation Certificate) wie unter 5.4.4.1 beschrieben erzeugt werden.

Bei dem Treffen hat dann der Vertreter der UNI-CA folgendes zu übergeben bzw. vorzulegen:

- eine „Ernennungsurkunde“, d.h. ein Bestätigungsschreiben z.B. des Rechenzentrumsleiters auf offiziellem Briefpapier der UNI, in dem die Zertifizierung durch die DFN-PCA beantragt und der UNI-CA-Mitarbeiter namentlich genannt und für den CA-Betrieb bevollmächtigt wird
- eine Diskette mit dem Public-Key der UNI-CA und dem zugehörigen Key Revocation Certificate
- den CA-Zertifizierungsantrag [PCAb]
- seinen *gültigen* Ausweis

(Weitere Einzelheiten beschreibt [PCAa].)

Der DFN-PCA-Mitarbeiter übergibt bei dieser Gelegenheit die authentischen Schlüsselinformationen zu den Signierschlüsseln der DFN-PCA.

Etwas später erhält die CA dann von der DFN-PCA per Mail das Zertifikat für ihren Signierschlüssel.

5.3 PR-Anlässe

Es bieten sich einzelne der bisher beschriebenen Wegmarken während der Einrichtungsphase der UNI-CA dazu an, vorab ein wenig Werbung in eigener Sache für die UNI-CA zu betreiben, indem entsprechende Ankündigungen in die passenden (internen) UNI-Newsgruppen (so vorhanden) gepostet werden:

Ankündigung/Vorstellung der UNI-CA Dies kann beispielsweise so etwas wie eine Absichtserklärung oder eine Vorankündigung sein

Freiwillige für Testläufe gesucht Hiermit könnten Nutzer angesprochen werden, die sich bereits mit PGP auskennen und die Interesse daran haben, am Aufbau der UNI-CA als Test-Zertifikatnehmer mitzuwirken (falls die Rechenzentrumsmitarbeiter selber dafür aus irgendeinem Grund nicht in Frage kommen)

UNI-CA Root-Schlüssel generiert

Zertifizierung durch die DFN-PCA Es darf gefeiert werden, die UNI-CA hat ihren offiziellen „Segen“ von der höchsten Zertifizierungsinstanz im DFN bekommen

Erste Nutzerschlüssel / erster Server zertifiziert

Cross-Zertifizierungen mit anderen Einrichtungen sind jedes Mal eine Meldung wert, zumal sie vermutlich nicht allzu häufig vorkommen

Zertifizierungstermine Das können Ankündigungen sein, daß die UNI-CA am Termin xxx im Gebäude $yyyy$ wieder die Vor-Ort-Zertifizierung gemäß ihrer Low-Level-Policy anbietet

Sicherheitswarnungen mit direktem Bezug zu oder Auswirkungen auf gängige Verschlüsselungsprogramme sollten ebenfalls vermeldet werden

Neue Services, Angebotserweiterungen Hier könnte auf neue, zusätzliche Angebote der UNI-CA hingewiesen werden, wenn sie ihr Tätigkeitsgebiet ausdehnt

5.4 Betrieb der CA (*“run”*)

5.4.1 Beispiel-Szenarien

In diesem Abschnitt wird der Ablauf einer idealtypischen Zertifizierung sowohl nach der Medium- als auch nach der Low-Level Policy dargestellt.

5.4.1.1 Medium-Level-Zertifizierung (im Rechenzentrum)

Die Medium-Level-Zertifizierung verläuft in zwei zeitlich und räumlich getrennten Teilschritten: Antragstellung und Identitätsprüfung auf der einen und eigentlicher Zertifizierungsvorgang auf der anderen Seite. Akteure sind der Zertifikatnehmer, ein CA-Mitarbeiter bei der Registrierung und ein CA-Mitarbeiter bei der Zertifizierung.

Antragstellung Ein Zertifikatnehmer hat vom Angebot der UNI-CA gehört und will seinen PGP-Schlüssel nach der Medium-Level-CA-Policy zertifizieren lassen. Er sucht dazu das Rechenzentrum auf und betritt das Zertifizierungsbüro, wo er von einem Mitarbeiter der UNI-CA empfangen wird. Der Zertifikatnehmer erhält vom CA-Mitarbeiter einen Medium-Level-Zertifizierungsantrag und füllt ihn aus. Anschließend unterschreibt er den Antrag vor den Augen des CA-Mitarbeiters. Dieser nimmt den Antrag entgegen, läßt sich den Personalausweis des Zertifikatnehmers zeigen, zeichnet den Antrag gegen und verschließt ihn in einem Schubfach. Dann gibt er dem Zertifikatnehmer eine

Diskette, die ausweislich des Aufklebers die Zertifizierungsschlüssel der UNI-CA und der DFN-PCA enthält. Der Zertifikatnehmer verläßt das Rechenzentrum, während der CA-Mitarbeiter die eingegangenen Anträge zum Zertifizierer bringt.

Zertifizierung Der Zertifizierer nimmt sich einen der neuen Anträge und prüft, ob der öffentliche Schlüssel des Antragstellers bereits vorliegt [ja, der Antragsteller hat ihn vorab per Mail eingesandt] und ob er alle Anforderungen der Medium-Level-Policy erfüllt. [Dies ist der Fall.] Daraufhin schickt der Zertifizierer eine verschlüsselte E-Mail mit einer Zufallszahl an die UNI-Mailadresse des Antragstellers. Er erhält kurze Zeit später diese Zufallszahl in einer vom Antragsteller signierten Mail zurück und kann die Signatur erfolgreich verifizieren. Der Zertifizierer weiß nun, daß der Antragsteller auch über den korrespondierenden geheimen Schlüssel verfügt. Damit sind alle Voraussetzungen erfüllt, der Zertifizierer schaltet gemeinsam mit einem CA-Kollegen den geheimen Signierschlüssel der CA durch Eingabe des Paßwortes frei und signiert damit den Public-Key des Antragstellers. Das so erzeugte Zertifikat schickt er per E-Mail an den Antragsteller und spielt es online in einen Verzeichnisdienst ein.

5.4.1.2 Low-Level-Zertifizierung (außer Haus)

Bei der Low-Level-Zertifizierung finden Antragstellung, Identitätsprüfung und Schlüsselzertifizierung an einem Ort und zu einer Zeit direkt hintereinander statt. Akteure sind der Zertifikatnehmer und der CA-Mitarbeiter.

Die UNI-CA ist z.B. auf dem Sommerfest der UNI mit einem Stand vertreten. Ein Zertifikatnehmer, der seinen Public-Key immer auf einer Diskette bei sich hat, will sich von der UNI-CA zertifizieren lassen. Er nimmt sich am UNI-CA-Stand einen der dort bereitliegenden Zertifizierungsanträge und füllt ihn aus. Dann gibt er ihn der UNI-CA-Mitarbeiterin, die den Stand betreut und dort mit einem Laptop steht. Die CA-Mitarbeiterin verlangt den Ausweis zu sehen, schaut mehrmals prüfend in den Antrag, auf den Ausweis und ins Gesicht des Zertifikatnehmers und schreibt dann „OK Müller 17.7.99“ auf das Feld mit der Überschrift „für Bearbeitungsvermerke freilassen“ auf dem Zertifizierungsantrag. Sie fragt sodann den Zertifikatnehmer, ob dieser seinen Public-Key bei sich habe, dieser bejaht und reicht ihr die Diskette mit dem Schlüssel. Die CA-Mitarbeiterin liest den Key von der Diskette in den Zertifizierungsrechner ein, prüft, ob der Schlüssel alle Anforderungen der Low-Level-Policy erfüllt, vergleicht die Schlüsselinformationen auf dem Antrag mit denen, die ihr der Zertifizierungsrechner anzeigt und zertifiziert dann den öffentlichen Schlüssel des Zertifikatnehmers. Anschließend speichert sie das Zertifikat auf der Diskette des Zertifikatnehmers, wirft sie aus und händigt sie dem Zertifikatnehmer zusammen mit einem Zettel mit der Aufschrift „Schlüssel-Infos Low-Level UNI-CA“ (und den entsprechenden Daten) aus. Später zurück im Rechenzentrum wird die CA-Admininistratorin das so erzeugte Zertifikat an die Keyserver schicken.

5.4.2 Objekt-Lebenszyklen

Der folgende Abschnitt betrachtet die Abläufe rund um die UNI-CA unter dem Gesichtspunkt, was mit den Daten-Objekten passiert, die dabei bewegt und verändert werden. Dargestellt wird der Lebenszyklus des Signier- und des Kommunikationsschlüsselspaares der UNI-CA unter normalen Bedingungen – d.h. es findet keine Kompromittierung während der Schlüssellebensdauer statt – sowie eines prototypischen von der UNI-CA ausgestellten Zertifikates. Ausgangsbasis dieses Szenarios ist die Annahme, daß die Gültigkeit eines Zertifikates mittels der Gültigkeitsdauer des Zertifizierungsschlüssels implizit ausgedrückt und nicht direkt (wie mit der neuesten PGP-Version möglich) im Zertifikat selber angegeben wird (vgl. dazu 4.8.7.2).

5.4.2.1 Lebenszyklus des CA-Zertifizierungsschlüssels

Zwei Wochen vor dem Ende des Wintersemesters wird der Zertifizierungsschlüssel der Medium-Level UNI-CA für die kommenden zwölf Monate zusammen mit seinem zugehörigen öffentlichen Schlüssel erzeugt. Mit ihm wird ein Zertifikat für den Public-Key ausgestellt. Der Public-Key wird zusätzlich mit dem noch gültigen alten Zertfizierungsschlüssel signiert, während der geheime Zertifizierungsschlüssel (Private-Key) auf eine grellbunte Diskette kopiert wird. Ein Widerruf seiner selbst wird zusammen mit dem zugehörigen Public-Key an die DFN-PCA übermittelt, die den Public-Key ebenfalls zertifiziert und den Schlüssel-Widerruf sicher verwahrt. Dann bekommt der Signierschlüssel viel zu tun: Lauter andere Schlüssel, deren Zertifikat gerade abgelaufen ist, werden mit ihm neu zertifiziert. Anschließend wird die Diskette mit dem Zertifizierungsschlüssel in einem Schrank eingeschlossen, aus dem sie nur selten wieder herausgenommen wird. Wenn dies geschieht, werden mit dem Private-Key andere Schlüssel, Schlüsselwiderrufe oder Sperrlisten signiert. Nach rund sechs Monaten bekommt der Zertifizierungsschlüssel wieder mehr zu tun: Mit ihm werden nun vor allem Schlüsselwiderrufe signiert, weil viele andere Zertifikatinhaber im laufenden Semester aus der UNI ausgeschieden sind, was zum Semesterwechsel festgestellt wird. Da die Medium-Level UNI-CA nur Zertifikate für Personen ausstellt, die der Universität angehören, werden die Zertifikate für die Schlüssel aller anderen, die keine UNI-Angehörigen mehr sind, gelöscht. Danach wird es wieder ruhiger. Kurz vor seinem ersten „Geburtstag“ wird mit dem Zertifizierungsschlüssel dann die öffentliche Komponente seines Nachfolgers signiert. Ganz kurz nach dem Erreichen eines Alters von zwölf Monaten wird der Zertifizierungsschlüssel dann gelöscht; der zugehörige Public-Key hingegen existiert in hunderten von Kopien weiter, die weiterhin zum Prüfen der zuvor mit dem Private-Key ausgestellten Signaturen und Zertifikate verwendet werden.

> “A digital signature private key must be securely destroyed when its active lifetime terminates. If its value is disclosed, even a long time after it is no longer actively used, it may still be used to forge signatures on old documents.” [For94]

5.4.2.2 Lebenszyklus des CA-Kommunikationsschlüssels

Genau zum Beginn des Sommersemesters wird das Kommunikationsschlüsselpaar erzeugt. Es soll sechs Monate lang benutzt werden. Als erstes wird sein öffentlicher Teil mit dem geheimen Signierschlüssel signiert und anschließend in alle Welt verteilt. Der geheime Teil wird auf sicherem Wege zusammen mit der Passphrase in Kopien an die Mitarbeiter der UNI-CA ausgehändigt. Sie benutzen ihn anschließend, um eingehende verschlüsselte Mails für die UNI-CA zu entschlüsseln. Mit Ablauf des Semesters wird mit dem geheimen Kommunikationsschlüssel ein Schlüsselwiderruf erzeugt und in alle Welt verteilt (z.B. über die Keyserver). Damit ist nach außen hin deutlich, daß seine öffentliche Komponente nicht mehr benutzt werden sollte, und die geheime Komponente wandert ins Archiv.

5.4.2.3 Lebenszyklus eines Zertifikates

Mitten im Semester wird ein Zertifikat für den Schlüssel des Studenten Mustermann erzeugt. Er war persönlich bei der UNI-CA, hat seinen Studentenstatus belegt, woraufhin sein öffentlicher Schlüssel mit dem geheimen Zertifizierungsschlüssel der Medium-Level UNI-CA unterzeichnet wurde, der noch bis zum Ende des nächsten Semesters (also noch rund neun Monate) gilt. Drei Monate später ereilt viele andere Zertifikate ihr planmäßiges oder vorfristiges Ende: Der Low-Level-Zertifizierungsschlüssel der UNI-CA läuft kurz nach dem Ende des Semesters ab, und alle mit ihm ausgestellten Zertifikate werden dadurch ungültig. Außerdem gibt es einige Medium-Level-Zertifikate, die vorzeitig gesperrt werden, weil die Inhaber ihrer Schlüssel nicht mehr an der UNI immatrikuliert oder beschäftigt sind. Kurz vor Ablauf des nächsten Semesters wird das Zertifikat dann noch einmal von der UNI-CA selbst verwendet: Sie prüft damit die Unterschrift unter einer E-Mail des Studenten Mustermann an die UNI-CA, in der er um eine Re-Zertifizierung seines Schlüssels bittet. Da die Unterschrift sich erfolgreich mit dem öffentlichen Schlüssel Mustermanns im Zertifikat verifizieren läßt, erstellt die UNI-CA mit ihrem neuen Zertifizierungsschlüssel für das kommende Semester ein neues Zertifikat, das diesmal ein Jahr lang gilt (wenn es nicht widerrufen wird ...). Das alte Zertifikat läuft dann mit dem Ende der Gültigkeitsdauer des alten Signierschlüssels ab.

5.4.3 Terminsachen

Aus einer anderen Perspektive sind Zeiträume und Termine, an denen etwas geschieht oder geschehen muß, wichtiger als die Objekte, mit denen etwas passiert. Diese Sichtweise für verschiedene regelmäßig wiederkehrende oder einmalige Stichtags-Aktivitäten stellen die nachfolgenden Abschnitte dar, indem sie die Tätigkeiten der CA nennen, die zum jeweiligen Termin oder im angegebenen Rhythmus ausgeführt werden sollten oder müssen.

5.4.3.1 Wöchentlich zu erledigen

- Korrektheit der Systemzeit auf dem CA-Rechner direkt beim Hochfahren prüfen – der Rechner kann sich nicht über eine Netzwerkverbindung mit den NTP-Servern synchronisieren, eventuell geht daher seine Uhr stärker falsch als die der übrigen, vernetzten UNI-Rechner (nötigenfalls korrigieren)
- kleiner Integritätscheck mit *tripwire*
- anstehende Zertifizierungen und Zertifikatwiderrufe durchführen
- inkrementelles Backup anfertigen
- DFN-PCA-Mailingliste verfolgen
- Mail für die CA beantworten/News lesen

- Laptop-Akkus aufladen

5.4.3.2 Monatlich wiederkehrende Arbeiten

- Kontrolle des CA-Rechners (Kompletter Integritätscheck mit tripwire, Prüfen der Logfiles)
- Komplett-Datensicherung, Wiederherstellbarkeit des Systems aus den Datensicherungen testen
- CRL-Erneuerung im in der Policy festgelegten Rhythmus (laut DFN-Policy mindestens monatlich), falls mit Sperrlisten gearbeitet wird
- eventuell: Präsenz auf regelmäßig monatlich stattfindenden Veranstaltungen

5.4.3.3 Halbjährlich wiederkehrende Arbeiten (Semesterende)

- Key-Rollover der Low-Level UNI-CA: Zertifizierungsschlüssel wird ungültig
- Benachrichtigung der Zertifikatnehmer, deren Zertifikat wegen des ablaufenden Zertifizierungsschlüssels ungültig wird
- Key-Rollover der UNI-CA: Kommunikationsschlüssel wird ungültig
- Archivierung alter Kommunikationsschlüssel
- Löschung abgelaufener Signierschlüssel
- Re-Zertifizierung von Nutzern
- Veröffentlichung der Liste der Sub-CAs (sofern zertifiziert), der RAs und der RA-Vereinbarung (Wortlaut)

5.4.3.4 Jährlich

- Key-Rollover der Medium-Level UNI-CA: Zertifizierungsschlüssel wird ungültig
- Benachrichtigung der Zertifikatnehmer, deren Zertifikat wegen des ablaufenden Zertifizierungsschlüssels ungültig wird
- Expire des Medium-Level Signierschlüssels
- User anschreiben und auf Möglichkeit der Re-Zertifizierung hinweisen
- eventuell: Präsenz auf jährlich stattfindenden Veranstaltungen

5.4.3.5 Stichtags-Aktivitäten (ca. einmalig)

29.2.2000 Datum prüfen: 2000 ist ein Schaltjahr, obwohl Vielfaches von 100 (weil auch Vielfaches von 400)

31.12.2000 Auslaufen der DFN-Policies, Ende des DFN-PCA-Projektes

Ablauf der UNI-CA-Policies
Bis zu diesem Zeitpunkt sollte feststehen, in welcher Form bzw. nach welcher Policy die Low-Level- bzw. die Medium-Level-CA über den letzten Gültigkeitstag der bisherigen Policies hinaus arbeiten sollen. Eventuell wird mit dem Inkrafttreten einer neuen oder geänderten Policy ein (außerplanmäßiger) Wechsel der CA-Schlüssel erforderlich.

5.4.4 Step-by-Step-Anleitungen

Hinter vielen der bislang mit einem Stichwort genannten Procederes wie „Schlüssel prüfen“ verbergen sich in Wirklichkeit etliche Einzelschritte. Die folgenden Anleitungen bzw. Checklisten sollen helfen, keinen der Punkte zu vergessen. Ein Teil der Punkte könnte bei entsprechender Software-Unterstützung durch spezielle CA-Software auch automatisiert erfolgen (z.B. bestimmte Abgleiche oder die Eintragung in eine Liste bereits zertifizierter Namen oder Schlüssel).

5.4.4.1 Erzeugung eines Schlüsselwiderrufs (Key Revocation Certificate)

1. Backup von Public- und Private-Key-Ring machen
2. Funktionsfähigkeit der Backups testen
3. Schlüsselwiderruf erzeugen
4. Schlüsselwiderruf in Datei extrahieren
5. Originalschlüssel aus Backup einlesen (Achtung: das Backup wird noch gebraucht!)
6. testen, ob das Einlesen funktioniert hat (Key muß jetzt wieder unwiderrufen sein)
7. Widerruf testen
8. Originalschlüssel wiederherstellen
9. Schlüsselwiderruf auf Diskette sichern (ggf. herkömmlich paßwort-verschlüsseln) und an DFN-PCA weitergeben
10. eigene Kopie des Schlüsselwiderrufs sicher verwahren

5.4.4.2 Registrierung und Identitätsprüfung

- Entgegennahme des Zertifizierungsantrages sowie des Personaldokumentes (und ggf. des Public-Keys auf Diskette)
- Ist der Antrag so leserlich, daß auch eine Person, die nicht zugleich das Personaldokument sieht, in der Lage ist, den Namen eindeutig zu entziffern?
- Ist das Personaldokument noch gültig?
- Sichtprüfung: Stimmen Antragsteller und Abbildung im Personalausweis überein? (Ansonsten könnte jeder Taschendieb mit einem gestohlenen Personaldokument ein Zertifikat unter falschem Namen beantragen!)
- stimmen Vorname(n) und Name im Personalausweis mit denen auf dem Antrag überein? (ansonsten könnte jeder Taschendieb mit einem gestohlenen Personaldokument ein Zertifikat unter falschem Namen beantragen)
- Vergleichen der geleisteten Unterschrift mit der im Personaldokument (wichtig: Die Unterschrift darf erst im Beisein des CA-Mitarbeiters geleistet werden, weil sonst äußerliche Ähnlichkeit z.B. von Geschwistern Betrugsmöglichkeiten eröffnen würde!)
- Bei Medium-Zertifizierung: Studentenausweis/Gehaltszettel/Projekt-Zettel zeigen lassen und prüfen, ob der dort genannte/eingedruckte Name mit dem im Personaldokument übereinstimmt
- Diskette mit den Public-Keys bzw. Zettel mit den Schlüsselinfos der UNI-CA aushändigen
- Antrag namentlich kennzeichnen und sicher verwahren bzw. gesichert an die CA weiterleiten

5.4.4.3 Schlüsselprüfung

- Liegt der Schlüssel bereits vor, oder muß er angefordert werden?
- Stimmen der Name und ggf. die Mailadresse aus der UserID (bzw. dem Subject-DN bei X.509-Zertifizierungen) mit dem Angaben auf dem Antrag überein?
- Sind Schlüssellänge und Erstellungsdatum des Schlüssels und die Angaben im Antrag identisch?
- Liegt das Erstellungsdatum des vorliegenden öffentlichen Schlüssels zwischen dem Erstellungsdatum von PGP 2.x und dem aktuellen Datum?
- Ggf. wenn eine Gültigkeitsdauer im Schlüssel enthalten ist: Ist der Schlüssel noch nicht abgelaufen?
- Ggf. wenn eine Gültigkeitsdauer sowohl in der Benutzerkennung als auch im PGP Key-Paket enthalten ist: stimmen beide überein?
- Erfüllt die Benutzerkennung alle Policy-Anforderungen (wie Schreibweise, UNI-Mailadresse)?

- Ist die Benutzerkennung selbst-signiert *mit diesem Key*?
- Hat der Schlüssel die erforderliche Mindestlänge?
- Ggf.: Hat der Schlüssel höchstens die maximal zulässige Länge?

5.4.4.4 Ablauf bei Medium-Level-Zertifizierung

- neue Zertifizierungsanträge entgegennehmen
- Prüfungen nach Checkliste
- Challenge mailen und auf Antrag vermerken
- Antwort auf Challenge archivieren
- Prüfung: stammt die Signatur unter der Challenge wirklich vom zu zertifizierenden Key?
- Public-Key auf Wechselmedium speichern
- CA-Rechner und Schlüsselmedium aus Schutzschrank nehmen
- möglichst *kein* Netz-, sondern Akku-Betrieb bei der Zertifizierung
- möglichst *keine* Zuschauer oder Besucher bei der Zertifizierung (außer dem zweiten CA-Mitarbeiter, der wegen des Vier-Augen-Prinzips erforderlich ist)
- CA-Rechner hochfahren, als Medium-Level-CA-Mitarbeiter anmelden
- zu zertifizierende Schlüssel vom Wechselmedium einlesen
- Schlüsselträger mounten/aktivieren
- (*) Zertifizierung starten, Name, Schlüssellänge, Key-ID und Erstellungsdatum prüfen
- prüfen, ob die UserID schon für einen anderen Key vergeben ist und falls ja, neuen Schlüssel *nicht* zertifizieren (und Abbruch bzw. nächster Key)
- *richtige* Benutzerkennung zusammen mit dem anderen CA-Admin signieren
- Zertifizierung auf dem Antragsformular und auch in eventuell geführten Logfiles vermerken
- Zertifikat extrahieren
- . . . [ggf. weitere Schlüssel auf diese Weise zertifizieren]
- Schlüsselmedium unmounten
- neue Zertifikate auf Wechselmedium sichern und sie damit auf einen vernetzten Rechner bringen
- CA-Rechner und Schlüsselmedium sicher verwahren
- Zertifikat publizieren (an Keyserver mailen) und an User mailen

- User(-Mailadresse) ggf. in Liste oder Datenbank als ‘zertifiziert’ vermerken (für Rundmails usw.)
- Antrag sicher archivieren/ablegen

5.4.4.5 Ablauf bei Low-Level-Zertifizierung

- CA-Rechner und ggf. Low-Level-CA-Schlüsselmedium aus Schutzschrank entnehmen
- aktuelle Version des globalen Public-Keyrings von einem der PGP-Keyserver auf den CA-Rechner kopieren (via Datenträgeraustausch – Achtung, das File ist groß!) als „Schlüsselvorrat“ für den Betrieb unterwegs als Service für alle, die ihren Public-Key nicht ständig bei sich tragen, ihn aber trotzdem gerne vor Ort zertifiziert haben möchten
- Integritätsprüfung mit tripwire, weil defekte Software oder Keys zu irreversiblen Folgen führen würden, da dadurch verursachte unbrauchbare Zertifikate dann schon ausgegeben wären
- sonstige Peripherie (Drucker, Ersatz-Akku, Anträge, Poster) mitnehmen
- Stand aufbauen:
  - Material auslegen
  - CA-Rechner aufbauen und gegen Wegnahme mit Seilschloß sichern
  - Anmelden auf CA-Rechner als Low-CA-Mitarbeiter und ggf. Mounten des Schlüsselmediums
- Schlüsselprüfung gemäß 5.4.4.3 durchführen
- keine Challenge erforderlich
- Public-Key aus Keyserver-Datei entnehmen bzw. von Diskette des Antragstellers einlesen
- Zertifizierung nach 5.4.4.4 ab (*) durchlaufen
- Zertifikat sichern
- Zertifizierung auf Antrag vermerken und Antrag sicher verwahren
- ggf. Zertifikat auf Diskette des Antragstellers speichern
- UNI-CA- und DFN-PCA-Public-Keys aushändigen
- ...
- bei Pausen o.ä.: Schlüsselmedium deaktivieren und sicher verwahren, Rechner nicht unbeaufsichtigt lassen
- ...
- Abbau und Rücktransport
- neue Zertifikate mit Wechselmedium von CA-Rechner auf vernetzten Rechner transportieren

- Schlüsselmedium wegschließen
- neue Zertifikate an User und Keyserver mailen und ggf. in Logfile eintragen
- Anträge ablegen bzw. archivieren
- Backup machen
- CA-Rechner wegschließen
- Akkus aufladen!!

5.4.5 Notfallpläne

5.4.5.1 Hardware-Ausfall Zertifizierungsrechner

Wird eine Reparatur des Zertifizierungsrechners erforderlich, so sind zwei Anforderungen gefährdet: Die Verfügbarkeit des Rechners ist nicht mehr gewährleistet (bzw. die Wieder-Verfügbarkeit unter ungünstigen Umständen sogar überhaupt nicht absehbar), und die Integrität der Software und der Daten auf dem Rechner ist gefährdet, sei es durch unbeabsichtigte Änderungen oder Löschungen während der Reparaturarbeiten, sei es durch mutwillige Manipulationen in dieser Zeit. Ein möglicher Ausweg könnte darin bestehen, die Festplatte aus dem Gerät zu entfernen, bevor es zur Reparatur gegeben wird. Eventuell erschwert das dann aber die Fehlersuche bzw. die Reparatur. Gegebenenfalls müßte die Platte gelöscht werden, bevor sie die Hände der CA-Mitarbeiter verläßt. Auch für diese Situation wäre also ein aktuelles Backup umso wichtiger, besonders im Hinblick auf die Low-Level-CA-Festplatte, da sie beim mobilen Einsatz einer höheren physikalischen Belastung und Gefährdung ausgesetzt ist. Sollte der CA-Rechner bei einem Defekt so ausfallen, daß es nicht mehr möglich ist, die Festplatte zu löschen, während sie in ihm betrieben wird, so muß sie an einen anderen Rechner angeschlossen und dort gelöscht und anschließend neu formatiert werden.

Ein anderer Ausweg könnte darin bestehen, die Medium-Level-CA-Festplatte in das Gerät einzubauen, bevor es zur Reparatur gegeben wird. Diese Harddisk enthält keine vertraulichen Daten (der geheime Schlüssel ist auf einem separaten Datenträger gespeichert, und die Nutzerdaten in den Zertifikaten sind zur Veröffentlichung bestimmt), und wenn der Rechner aus der Werkstatt zurückkommt, könnte anhand der Integritätsprüfung mit (beispielsweise) *tripwire* festgestellt werden, ob und wenn ja welche Dateien modifiziert, gelöscht oder hinzugefügt worden sind. Diese müßten dann ggf. von der letzten Sicherungskopie wieder eingespielt werden.

5.4.5.2 „Tag X“ (CA-Schlüssel-Kompromittierung)

Um in diesem Fall alle erforderlichen Maßnahmen ergreifen zu können, ist es wichtig, daß bereits vorher während der normalen Zertifizierung entsprechende Vorbereitungen getroffen werden. Dazu gehört u.a., eine Möglichkeit vorzusehen, wie alle Zertifikatnehmer per E-Mail benachrichtigt werden können, falls es zu einer Kompromittierung eines CA-Schlüssels und in deren Folge zu einer Sperrung aller mit dem Key unterzeichneten Zertifikate kommt. Andere Fälle, in denen diese

Kommunikationsmöglichkeit genutzt werden sollte, um Informationen an die CA-Nutzer weiterzugeben, wären beispielsweise eine Adreßänderung der Zertifizierungsstelle oder das Bekanntwerden grundsätzlicher Sicherheitslücken in oder neuer Angriffe auf gängige Verschlüsselungsverfahren.

Wird der geheime Signierschlüssel der CA entwendet oder kann diese Möglichkeit bei seinem Verschwinden zumindest nicht ausgeschlossen werden, so ergibt sich bei PGP ein Problem: Der Schlüssel und alle mit ihm ausgestellten Zertifikate müßten so schnell wie möglich für ungültig erklärt werden, indem für ihn ein Schlüsselwiderruf (*Key Revocation Certificate*, KRC) ausgestellt wird. Dies ist aber nur unter Verwendung des Schlüssels selbst möglich, der ja in diesem Falle nicht mehr vorliegt. Für genau diesen Fall ist das *vorsorgliche* Erzeugen eines Schlüsselwiderrufes gedacht. Da die DFN-Policy für jede von ihr zertifizierte CA die Hinterlegung eines solchen KRC bei der DFN-PCA vorschreibt, kann nun darauf zurückgegriffen werden. Die DFN-PCA müßte also auf geeignete Weise und möglichst umgehend davon unterrichtet werden, daß eine Kompromittierung vermutet wird oder zumindest nicht ausgeschlossen werden kann und daher der Widerruf für den betreffenden Schlüssel veröffentlicht werden sollte. (Da die DFN-PCA pro Mitgliedseinrichtung nur eine CA zertifiziert, sollte es hier keine Mehrdeutigkeiten geben, welcher Schlüssel gemeint ist.)

5.5 Stolpersteine

Zum Schluß noch einige Beispiele für kuriose Dinge, die besser vermieden werden sollten oder die unerwartet Schwierigkeiten bereiten können:

- alle zertifizierten Schlüssel werden (ausschließlich) als selbst-extrahierendes `.exe`-File auf dem WWW-Server der CA vorgehalten
- Widerrufslisten (für PGP-Schlüssel) sind nicht unterschrieben
- Widerrufslisten (für PGP-Schlüssel) sind zwar unterschrieben, jedoch nicht mit dem Signierschlüssel der CA
- In der Policy werden viele neue/ungebräuchliche Abkürzungen eingeführt
- KeyIDs unterschiedlicher Schlüssel sind nicht zwangsläufig verschieden:

```
$ pgp -kvc 0x19980101
[...]
Type Bits/KeyID    Date       User ID
pub  2048/19980101 1998/01/12 in-ca@individual.net SIGN EXPIRE:1999-12-31
                              Ueberregionale CA des Individual Network e.V.
           Expire: 1999/12/31 SIGNature only
            Key fingerprint = C4 0E 2C 31 1D DB 5F DB  AF 5F 2C AF 9D 44 13 10

pub  2048/19980101 1998/01/12 in-ca@individual.net SIGN EXPIRE:1999-12-31
                              Root CA des Individual Network e.V.
           Expire: 1999/12/31 SIGNature only
            Key fingerprint = B3 06 9A 8D 38 04 3C 75  41 32 EE DC 8B 7D 61 0D
2 matching keys found.
```

- Es können PGP-Keys mit einem Erstellungsdatum in der Zukunft auftreten:

```
pub 2048/FEECC7A5 2018/06/14 Michael [...] <...@....in-berlin.de>
```

- Es können Tippfehler in den UserIDs vorliegen:

```
Type Bits/KeyID    Date       User ID
pub  1024/0XXXXXXX 2000/02/xx xxxxxxxxxxxxxx <xxxxx@t-online de>
                                                             =
```

- Wie verfährt man bei ungewöhnlichen Darstellungen von Sonderzeichen im Namen in der UserID, z.B. bei Umlauten, die als 8bit-ISO-Latin1-Zeichen in der UserID vorkommen und auf Rechner mit anderer Zeichensatz-Codierung als Sonderzeichen dargestellt werden?

- Sind Schlüssel zertifizierbar, bei denen der Name des Inhabers einen Umlaut enthält, dieser in der UserID aber mit dem nicht-umgelauteten Vokal geschrieben ist (also z.B. Name ‘Müller’, UserID ‘`... Muller ...`’)?

- Sollen Schlüssel zertifiziert werden, die den Namen in der UserID nicht in der Standardform ‘*Vorname Nachname E-Mailadresse*’ enthalten, sondern nur als Bestandteil der Mailadresse, etwa `hans.mustermann@UNI.de`?

Probleme bei der X.509-Zertifizierung bereitet vor allem die eindeutige Zuordnung zwischen Zertifizierungsantrag und einem Zertifizierungs-Request (Certificate Signing Request, CSR), wenn dieser nur per E-Mail übermittelt wird. Viele Programme, die X.509-Zertifikate verarbeiten, bieten keinerlei Möglichkeit, sich den erzeugten CSR oder eine kryptographische Prüfsumme (Hash-Wert, Message Digest) davon anzeigen zu lassen, so daß – anders als bei PGP – auch kein „Fingerprint“ o.ä. im Zertifizierungsantrag angegeben werden kann. Es fehlt dann an der Verknüpfung zwischen schriftlichem Zertifizierungsantrag und dem elektronischen CSR, was z.B. Verwechslungen begünstigt. (Möglicher *workaround*: siehe Anhang H.1.1)

Kapitel 6

Ausblick

The challenge is ... to provide the infrastructure
for automatable Trust Management for everyday life.
We need to protect our own identities
and be sure of our correspondents'.
— ROHIT KHARE, *The World Wide Web Journal* [Kha97]

Zum Abschluß soll noch der Versuch unternommen werden, absehbare Entwicklungen und Trends im Zusammenhang mit der Public-Key-Zertifizierung zu skizzieren und zukünftige Herausforderungen für eine Zertifizierungsstelle auszuloten.

Die rasche Entwicklung und der hohe Innovationsdruck auf dem Gebiet der Informations- und Kommunikationstechnik lassen einen Stillstand oder das Ausruhen auf dem Erreichten kaum zu. Auch und gerade das Feld der Sicherheit ist davon betroffen. Daher soll in diesem Abschnitt versucht werden, Standardisierungsbestrebungen und Konzepte von Bedeutung für Public-Key-Infrastrukturen sowie mögliche zusätzliche Aufgabengebiete für eine UNI-CA ebenso zu umreißen wie solche Aufgaben, die sich für sie daraus ergeben könnten, daß ihre Dienste in nicht allzu ferner Zukunft etabliert sein werden und sie den Status eines (Pilot-)Projektes hinter sich lassen wird.

6.1 Absehbare technische Entwicklung

Die technische Entwicklung im Zertifizierungsumfeld betrifft vor allem zwei Felder: neue Standards und daraus resultierend zumeist auch neue Programme oder Software-Versionen und gänzlich neue Konzepte, die mittel- bis langfristig auch Änderungen an der gerade erst entstehenden Zertifizierungsinfrastruktur erforderlich machen könnten.

6.1.1 Software, Standards

Hinsichtlich der Standardisierung mit Bezug zu Public-Key-Verfahren sind drei wichtige Entwicklungen absehbar:

Die *Internet Public Key Infrastructure*-Arbeitsgruppe (PKIX)[1] der Internet Engineering Task Force hat inzwischen die ersten **Standards für eine Internet-PKI auf Basis von X.509-Zertifikaten** vorgelegt,[2] der Wichtigste darunter sicher [FHPS99]. Darin werden u.a. die Abläufe zur Überprüfung eines Zertifizierungspfades und Formate für Zertifikate und Widerrufslisten beschrieben, die in der zukünftigen Internet-X.509-Public-Key-Infrastruktur verwendet werden sollen. Da diese Standards (und teilweise noch Entwürfe) gerade erst verabschiedet wurden, existieren noch kaum Implementierungen, die diese Verfahren umsetzen. Da so gut wie alle großen Anbieter von Zertifizierungssoftware oder entsprechenden Dienstleistungen am Entwurf dieser Standards beteiligt waren und sind, kann man aber davon ausgehen, daß die meisten diesen neuen Internet-Standard auch in ihren Produkten umsetzen werden. Für eine Zertifizierungsstelle bedeuten diese Internet-Standards, daß entsprechende standard-konforme Software eingesetzt werden sollte, sobald sie verfügbar ist. Damit einher gehen einige neue Dienste für Zertifikat-Überprüfungen, wie z.B. das Online Certificate Status Protocol, das ebenfalls von der PKIX-Arbeitsgruppe entwickelt wurde [AAG+99].

Die Anhänger von PGP versuchen, mit dem **OpenPGP**-Standard [CDFT98] ein Gegengewicht und eine Alternative zu dem von vielen großen Firmen, insbesondere Netscape und Microsoft, unterstützten S/MIME zu etablieren. Da PGP mit seinen bis zu 20 Millionen Anwendern weltweit zur Zeit noch den De-facto-Standard für vertrauliche E-Mail-Kommunikation im Internet darstellt [SW97], dürfte viel davon abhängen, ob die PGP-Gemeinde mit den unterschiedlichen, teilweise untereinander inkompatiblen kommerziellen und nicht-kommerziellen PGP-Versionen weiter zersplittert und so an Bedeutung verliert, oder ob freie und kommerzielle Implementierungen dank des OpenPGP-Standards interoperabel sein werden. Viel wird in dieser Hinsicht wohl davon abhängen, ob Network Associates, der Anbieter der kommerziellen PGP-Versionen, sich selbst an die unter eigener Federführung entwickelte OpenPGP-Spezifikation halten oder aber eigene Wege gehen wird. Die Entwickler freier PGP-Versionen wie z.B. *Gnu Privacy Guard* [Roe98, Lei99] werden sich aller Voraussicht nach an den Standard halten (schon um angesichts der geringeren Verbreitung ihrer jeweiligen Software deren Chancen nicht leichtfertig weiter zu schmälern). Der OpenPGP-RFC selbst wird aktiv weiterentwickelt, zur Zeit liegt ein erster Internet-Draft für sein Nachfolger-Dokument vor [CDFT99].

Weitergehende Interoperabilität könnte eine **Integration von PGP und X.509** bringen, die dann vermutlich am wahrscheinlichsten in Form von X.509-Unterstützung durch PGP/Network Associates realisiert würde, da X.509 sich außerhalb des PGP-Umfeldes als Standard-Zertifikatformat etabliert hat. Network Associates selber plant eine entsprechende Unterstützung von X.509 für zukünftige Versionen ihrer PGP-Software-Suite, wie die folgende Aussage von SIMON LEECH, Prime Support Account Manager, Network Associates International B.V., vom 8. Dezember 1998 an JOHN HORTON, weitergeleitet an die PGP-Directory-Mailingliste[3], belegt:

> Thanks for your enquiry regarding PGP and x509 compatibility. According to our internal PGP roadmap, full support for x.509 will be included in PGP v7.0, which will include an integrated PGP/x509 CA server and will be available in the second half of 1999.

[1] `http://www.imc.org/ietf-pkix/`
[2] Einen Überblick gibt die Web-Seite der Arbeitsgruppe (s. vorige Fußnote 1) oder die PKIX-Roadmap [AT99]
[3] Archiv der Liste unter `http://www.dante.net/np/listarchives/pgp.html`

> However, v6.5 will include integration of our PGP VPN client, which will include support of both PGP and RSA based x509v3 certificates. This will hopefully be available in Q1/Q2 next year.

Damit gäbe es dann mit X.509 ein einheitliches Zertifikatformat, was die Arbeit von Zertifizierungsstellen erheblich erleichtern könnte, da dann nicht mehr PGP- und X.509-Zertifizierung separat betrieben werden müßten.

PGP 6.5 ist mittlerweile verfügbar und unterstützt in seiner VPN-Funktionalität tatsächlich X.509. Auch wenn sich die Version 7.0 um etwa ein Jahr verzögern dürfte, bleibt doch zu hoffen, daß NAI seine PGP-Roadmap einhält und in PGP 7.0 volle X.509-Unterstützung realisiert.

Ende 2000 läuft ferner das US-Patent auf den RSA-Algorithmus[4] aus. Damit wird er in wenigen Monaten lizenzfrei nutzbar sein, was es erleichtern dürfte, ihn als (einen) Standard- oder sogar Default-Algorithmus in IETF-Standards oder in kommerziellen oder freien Software-Produkten einzusetzen. – Dies wäre u.U. auch unter dem Aspekt der Absicherung gegen den Ausfall einzelner Verfahren hilfreich, auf die am Ende des folgenden Abschnittes näher eingegangen wird.

6.1.2 Infrastrukturen, Konzepte

Am einfachsten realisierbar und insofern auch am wahrscheinlichsten dürfte die **Kombination und Integration** von Zertifizierungsdiensten **mit anderen bereits existierenden Diensten** sein. So hilft z.B. der PathServer von MICHAEL REITER und STUART STUBBLEBINE [RS97b] dabei, Zertifizierungspfade zum Schlüssel eines Kommunikationspartners im ansonsten eher unstrukturierten PGP Web-of-Trust zu finden. Diese Suchmöglichkeit ist bisher noch kaum bekannt, und sie wird auch noch nicht systematisch von anderen Programmen genutzt. Hier liegt noch einiges Entwicklungspotential brach, insbesondere in Zusammenhang mit komplexeren Verfahren zur Vertrauensbewertung als den bisher üblichen (siehe auch die Ausführungen zu *Trust Metrics* im nächsten Absatz). Ein weiteres Beispiel sind Zeitstempeldienste, wie sie beispielsweise die Zertifizierungsstelle der Humboldt-Universität zu Berlin bereits für interne Zwecke oder auch der *PGP Digital Timestamping Service* von MATTHEW RICHARDSON [Ric97] anbieten und wie sie die PKIX-Arbeitsgruppe der IETF standardisiert [ACPZ00]. An einer Integration solcher Dienste, die die Zertifizierung von öffentlichen Schlüsseln gut ergänzen würden, fehlt es bislang; sie könnte die Nutzung aller dieser Angebote erheblich steigern.

Die heute existierenden oder im Aufbau befindlichen Zertifizierungsinfrastrukturen haben so gut wie alle einen wesentlichen Punkt gemeinsam: Sie sind in der Regel hierarchisch aufgebaut, und ein Nutzer ihrer Dienste muß allen übergeordneten oder sogar allen Zertifizierungsstellen vertrauen, wenn er die von ihnen ausgestellten Zertifikate nutzen will. Bestenfalls hat er die Möglichkeit, eine Entweder-Oder-Entscheidung zu treffen zwischen „Ja, ich vertraue dieser CA (bedingungslos und uneingeschränkt)“ und „Nein, in diese CA und ihre Arbeit habe ich kein Vertrauen“. Abstufungen wie „erscheint mir ziemlich vertrauenswürdig“ oder „scheint meist zuverlässig zu arbeiten“ lassen sich in dem vorherrschenden Zertifizierungsmodell nicht ausdrücken. Zumindest einen kleinen Schritt weiter geht hier PGP mit den Vertrauensbewertungen, die der Nutzer für andere Personen

[4] `http://www.patents.ibm.com/details?&pn=US04405829__&s_all=1`

vergeben kann, und mit der Vorgabe der Zahl an PGP-Zertifikaten „einigermaßen“ vertrauenswürdiger Personen, die benötigt werden, damit ein fremder PGP-Schlüssel als authentisch eingestuft wird (PGP-Konfigurationsparameter `Marginals_Needed`).

Im „klassischen“ hierarchischen Zertifizierungsmodell sind darüber hinaus – abgesehen von Cross-Zertifizierungen – auch keine *redundanten* Zertifizierungspfade vorgesehen. Fällt eine Zertifizierungsstelle z.B. aufgrund eines *key compromise* aus, so sind damit alle ihr in der Zertifizierungshierarchie untergeordneten CAs und Nutzer vom restlichen Teil der Hierarchie isoliert. – Auch hier schneidet das Web-of-Trust-Modell der PGP-Zertifizierung etwas besser ab: Hier können beliebige, auch mehrere und indirekte Pfade zwischen zwei Knoten im Zertifizierungsgraph existieren.

Das hierarchische Zertifizierungsmodell ist aber nicht zwingend. Es gibt andere Ansätze, die flexiblere und mehr Zertifizierungspfade zulassen und dann anhand mathematischer Modelle die Aussagekraft und Vertrauenswürdigkeit einzelner Pfade in dem resultierenden Zertifizierungsnetz zu ermitteln suchen [RS97a, ARH97]. Anhand von sog. ***trust metrics***, also einer zahlenmäßigen (und bei Bedarf änderbaren) Bewertung für jeden Zertifikat-Aussteller oder sogar für jedes Zertifikat, läßt sich bei diesen Modellen das Vertrauen in die Zertifizierung, die Dritte vorgenommen haben, feiner abgestuft bewerten als im bool’sch bewerteten hierarchischen Ansatz. Mehrere voneinander unabhängige Zertifizierungspfade, die zum selben Schlüssel führen, können bei solchen Vertrauensmetriken die Wahrscheinlichkeit oder das Vertrauen erhöhen, daß der betreffende Schlüssel authentisch ist.

Ein zusätzliches Betätigungsfeld könnte sich aus **anderen Sicherheits- oder Zertifizierungs-Architekturen** ergeben, in denen man nicht darauf angewiesen ist, einer „Root-CA“ o.ä. zu vertrauen.[5] Hintergrund dieser alternativen Ansätze ist die Überlegung, daß man normalerweise niemandem mehr Vertrauen entgegenbringt als sich selbst – oder wie es Khare und Rifkin formulieren:

> “Relying on hierarchical CAs weakens the principle of *trusting yourself*, since it requires blanket trust in very large scale CAs, with corresponding conflicts of interest.” [KR97, S. 39]

Schließlich kann man hier auch noch einen Punkt nennen, der sich pointiert mit „Katastrophen-Vorsorge“ beschreiben ließe: die Einführung alternativer Verschlüsselungs-, Authentifizierungs- und Zertifizierungsverfahren, um eine unter Sicherheitsgesichtspunkten unerfreuliche Abhängigkeit von einem einzigen Verfahren oder einer einzigen Infrastruktur (*single point of failure*) zu vermeiden.[6] Dies wird umso wichtiger werden, je stärker Public-Key-Verfahren und elektronische Kommunikation herkömmliche Wege des Nachrichtenaustausches verdrängen (und diese in der Folge immer stärker reduziert werden, so daß keine Rückfall-Alternativen mehr zur Verfügung stehen), zumal die Stärke der kryptographischen Verfahren, auf denen bislang die verbreitetsten der Public-Key-Verfahren aufbauen, nicht mathematisch *bewiesen* worden ist, sondern nur vermutet wird.

[5] Beispiele für solche Ansätze sind die *Simple Distributed Security Infrastructure*, SDSI [LR96] und die *Simple Public-Key Infrastructure*, SPKI [Ell99, EFL^{+}99].

[6] siehe dazu ZIESCHANG [Zie97, S. 343]

6.2 Erweiterung des Aufgabenfeldes der UNI-CA

Eine Zertifizierungsstelle könnte, wenn Interesse besteht, ihr Tätigkeitsfeld nach und nach, je nach Nachfrage und ihren eigenen personellen, technischen usw. Möglichkeiten, erweitern. Sie könnte beispielsweise selbst einen Zeitstempeldienst anbieten oder den einer anderen Einrichtung zugänglich machen, unter Umständen würde sie zunächst auch erst einmal intern Zeitstempeldienste wie das „Ewige Logfile“ [Don96] zu Protokollierungszwecken nutzen (auch bzw. eventuell zuerst für interne Audit-Zwecke der CA). Für interne Nachweiszwecke wären auch Konzepte wie das eines sicheren Logfiles auf unsicheren Maschinen [KS98] geeignet.

Darüber hinaus sind weiterhin Aktivitäten denkbar wie

- die Übernahme der Aufgabe einer Registrierungs- und Ausgabestelle für SigG-Zertifikate bzw. -Chipkarten
- Unterstützung bei der Verwendung von SigG-Software (damit ist mit großer Wahrscheinlichkeit zu rechnen – die UNI-Nutzer werden das dem Kompetenzgebiet der CA-Mitarbeiter zuordnen und mit Fragen dazu auf sie zukommen, gerade wenn herstellerunabhängiger Rat gesucht oder gebraucht wird)
- der Verkauf von Smartcard-Lesern
- Anonymitäts- und Pseudonymitätsdienste, z.B. für die Nutzung des World Wide Web durch die UNI-Angehörigen
- das Signieren von Inhaltsbewertungen (*ratings*) für WWW-Seiten [MR96, DLLC97]
- Support für digitale Signaturen in PDF-Dokumenten [ENT99]
- Public-Key-authentisierter Ressourcen-Zugriff, z.B. auf den UNI-Compute-Server, wie dies bereits im UNICORE-Projekt [AS98] erprobt wird

Ferner wäre es mittelfristig denkbar, den Login auf den Rechnern des Campus-Netzes mit Nutzername und Paßwort auf eine Public-Key-Authentifizierung umzustellen oder zumindest eine kryptographische Absicherung einfacher Paßwort-Logins durch asymmetrische Verschlüsselungsverfahren [HK98] einzuführen.

Der zuletzt genannte Punkt könnte für die UNI noch aus einem anderen Grund sehr interessant sein: In Firmen wurden erhebliche Einsparmöglichkeiten durch Public-Key-Authentifizierung beim Login festgestellt [NC98], weil dadurch weniger Arbeitszeit bei der Login-Prozedur verlorengeht und weil der Helpdesk nicht so viele Anfragen wegen Login- oder Paßwort-Problemen bekommt.

Weiterhin wären auch Einsparungen für die Universität insgesamt denkbar, die möglich werden, wenn Arbeitsabläufe effizienter und ohne Medienwechsel gestaltet und dann Verschlüsselung und Signaturen nun vollständig elektronisch abgewickelt können.

Ein weiterer Punkt, der ebenfalls zu konkreten Einsparungen im Universitätshaushalt führen könnte, wäre die Möglichkeit, Universitätswahlen und -abstimmungen online durchzuführen und dadurch Versandkosten für die Wahlunterlagen und Wahlbriefe zu sparen.[7]

[7] Ein entsprechendes vom BMWi gefördertes Pilotprojekt wird an der Universität Osnabrück durchgeführt. Einen Teil des Projektes soll auch eine aufzubauende Zertifizierungsstelle darstellen [BMW99a].

6.3 Auslaufen des DFN-PCA-Projektes

Das jetzige DFN-PCA-Forschungsprojekt ist bis 31. Dezember 2000 befristet. Der Forschungsaspekt der Projektarbeit wird mit hoher Wahrscheinlichkeit auch über das Jahr 2000 hinaus fortbestehen bzw. fortgesetzt werden. Falls das Projekt jedoch nicht mehr als *Forschungs*projekt weitergeführt werden sollte, sondern ausschließlich Teil eines regulären Dienstangebots – z.B. der DFN-CERT GmbH – für die DFN-Mitgliedseinrichtungen würde, so würde sich auch die Frage stellen, ob dann nicht möglicherweise eine Lizensierung des in PGP verwendeten symmetrischen IDEA-Verfahrens erforderlich wird. (Zur Zeit sind keine Lizenzen erforderlich, wie Rechteinhaber ASCOM der DFN-PCA 1996 auf Anfrage bestätigt hat.[8]) Immerhin bliebe es ja auch nach Beendigungen des Forschungsprojektes DFN-PCA bei einer Nutzung von IDEA im Deutschen *Forschungs*netz – also auch weiterhin zu Forschungszwecken. ASCOM könnte das aber möglicherweise anders interpretieren und sich auf den Standpunkt stellen, ihre *"Special conditions applicable for European research projects"* gälten dann nicht mehr, so daß aus Sicht von ASCOM möglicherweise diese Passage ihrer Lizenzbedingungen[9] griffe:

> "Any use of the algorithm after termination of a project, including activities resulting from a project and for purposes not directly related to the project, strictly requires a site license, product license or end-user license."

Hier wird also Klärungsbedarf entstehen, inwieweit für den PGP-Einsatz in der DFN-PCA (oder deren Träger) und in den Sub-CAs zukünftig eventuell IDEA-Lizenzen erforderlich werden könnten, wenn das PCA-Projekt ausläuft. Für den einzelnen Anwender, der PGP privat nutzt, ändert sich mit dem Auslaufen des PCA-Projektes nichts, ebenso wenig wie für Projekte, in denen PGP verwendet wird.

[8] `http://www.pca.dfn.de/dfnpca/ascom.html`
[9] `http://www.ascom.ch/infosec/idea/types.html`

Teil II

Zertifizierung mit OpenSSL

Kapitel 7

OpenSSL-0.9.5

7.1 Einleitung

OpenSSL ist eine frei verfügbare Implementierung des SSL/TLS-Protokolls und bietet zusätzlich Funktionen zur Zertifikat-Verwaltung sowie verschiedene kryptographische Funktionen. Es basiert auf dem SSLeay-Paket, das von ERIC A. YOUNG und TIM HUDSON entwickelt wurde. OpenSSL als Nachfolger von SSLeay wird derzeit von einer unabhängigen Gruppe weiterentwickelt.

Das Paket umfaßt mehrere Applikationen, z.B. zur Erzeugung von Zertifikaten, von Zertifizierungsanträgen und zur Verschlüsselung. Diese einzelnen Applikationen sind zusammengefaßt in einem Kommandozeilen-Programm: `openssl`.

Der Schwerpunkt dieses Dokuments liegt auf dem praktischen Einsatz von OpenSSL zur Erzeugung, Verwaltung und Verwendung von Zertifikaten. Die Funktionen zur Zertifikatverwaltung in OpenSSL sind ursprünglich nicht für den Betrieb einer CA gedacht gewesen. Es sollten lediglich Zertifikate erzeugt werden können, um das (zertifikatbasierte) SSL/TLS-Protokoll sinnvoll einsetzen zu können. Die Entwicklung des OpenSSL-Teils, der für die Zertifikatverwaltung notwendig ist, ist inzwischen so weit vorangeschritten, daß der Betrieb kleiner bis mittlerer CAs gut möglich ist.

Die letzte OpenSSL-*Anwender*-Version (0.9.4) stammt vom 9.6.1999. Dieses Dokument bezieht sich auf die OpenSSL-*Entwickler*-Version 0.9.5-*dev*. Die in diesem Dokument gemachten Angaben sollten sich problemlos auf die nächste Anwender-Version beziehen lassen. Nach Herausgabe der neuen Anwender-Version wird dieses Dokument ggf. angepaßt werden. Die Entwickler-Versionen, „snapshot“ genannt, tragen die Bezeichnung `openssl-SNAP-2000`*mmdd*, wobei *mm* und *dd* für die Monats- bzw. Tages-Zahl stehen. Die `SNAP`-Bezeichnung wird in diesem Dokument immer dann verwendet, wenn es um konkrete Kommandos zu Übersetzung bzw. Installation geht. Andernfalls wird die Bezeichnung `openssl-0.9.5-dev` verwendet.

Die Entwickler-Version zeichnet sich gegenüber der letzten Anwender-Version u.a. durch eine deutlich verbesserte Dokumentation aus. Für viele Teile des Programms sind jetzt Manual-Seiten vorhanden.

7.2 Installation von OpenSSL-0.9.5-dev

Es gibt zwei Möglichkeiten, das Paket zu kompilieren. Die erste Möglichkeit faßt die einzelnen Applikationen zu einem monolithischen Programm zusammen, `openssl`. Auf die einzelnen Applikationen wird dann zugegriffen, indem `openssl` mit dem Applikationsnamen als erstem Parameter aufgerufen wird.

Die zweite Möglichkeit kompiliert die Applikationen als eigenständige Programme, es gibt kein monolithisches Programm.

Für beide Möglichkeiten ist die Konfiguration und teilweise auch die Übersetzung und Installation des Paketes gleich. Daher wird ggf. beides gemeinsam behandelt.

Alle Angaben zur Übersetzung und Konfiguration des Paketes beziehen sich auf das Betriebssystem SunOS 5.5.1 (Solaris 2.5.1), den C-Compiler gcc-2.7.2.1 und `openssl-SNAP-20000222.tar.gz`.

7.2.1 Konfiguration und Übersetzung

Zunächst wird das OpenSSL-Quell-Paket ausgepackt, z.B. in `/usr/src`:

```
gzip -dc openssl-SNAP-2000mmdd.tar.gz | tar -xvf -
```

Nach Wechsel in das Quell-Verzeichnis `openssl-SNAP-2000mmdd` wird das Paket konfiguriert. Zur Konfiguration können mehrere Optionen angegeben werden. Hier wird nur auf zwei Optionen eingegangen, die anderen werden in der Datei `openssl-SNAP-2000mmdd/INSTALL` beschrieben. Mit der Option `--prefix=`*xxx* wird ein Pfad angegeben, unterhalb dessen die OpenSSL Programm-Dateien installiert werden. Es werden dann bei einer automatischen Installation die Verzeichnisse *xxx*`/bin`, *xxx*`/lib` und *xxx*`/include/openssl` angelegt. Mit der Option `--openssldir=`*xxx* wird festgelegt, in welchem Verzeichnis die Dateien bzw. Verzeichnisse zur Zertifikatverwaltung und die Manual-Seiten abgelegt werden. Es werden dann die Verzeichnisse *xxx*`/certs`, *xxx*`/misc`, *xxx*`/private`, *xxx*`/man` und die Konfigurationsdatei *xxx*`/openssl.cnf` angelegt.

Konfiguriert wird das Paket durch folgenden Befehl:

```
sh ./config --prefix=/usr/local --openssldir=/usr/local/etc/ssl
```

Nun kann das Paket übersetzt werden. Durch die Übersetzung werden Bibliotheken erzeugt, die auch für die Erzeugung der Programmsammlung (siehe 7.2.2) benötigt werden. Daher sind die folgenden Befehl auch auszuführen, wenn nur die Programmsammlung erzeugt werden soll.

```
make
```

Mit folgendem Befehl werden umfangreiche Tests des übersetzten Pakets durchgeführt:

```
make test
```

Erfolgte der Testlauf im Sinne der Tests fehlerfrei, kommt abschließend eine Meldung ähnlich der folgenden:

```
OpenSSL 0.9.5-dev 09 Aug 1999
built on: Mon Feb 14 10:44:30 MET 2000
platform: solaris-sparcv8-gcc
options:  bn(64,32) md2(int) rc4(ptr,char) des(idx,cisc,16,long) idea(int) blowfish(ptr)
compiler: gcc -DTHREADS -D_REENTRANT -mv8 -O3 -fomit-frame-pointer -
Wall -DB_ENDIAN -DBN_DIV2W
`test' is up to date.
```

Soll das monolithische Programm eingesetzt werden, kann jetzt mit der Installation, die im übernächsten Abschnitt (7.2.3) beschrieben wird, fortgefahren werden.

Zur Erzeugung der Programmsammlung ist eine weiterer Übersetzungsschritt notwendig. Er wird im nächsten Abschnitt beschrieben.

7.2.2 Übersetzung als Programmsammlung

Nach Ausführung des `make`-Befehls im vorigen Abschnitt (7.2.1) liegt jetzt das komplette Paket übersetzt vor. Die Übersetzung diente aber nur dazu, die Bibliotheken `libssl.a` und `libcrypto.a` im Quellverzeichnis zu erzeugen. Sie sind notwendig für die weitere Übersetzung. Die einzelnen Applikationen werden am einfachsten mit folgendem Shell-Skript im Quell-Verzeichnis übersetzt. Da die einzelnen Applikationen z.T. unterschiedliche Programmteile und Bibliotheken benötigen, fällt die Übersetzung für jedes Programm etwas unterschiedlich aus. Die benötigten Bibliotheken werden statisch in die Programme gelinkt, so daß die Programme jeweils um die fünf Megabytes groß sind.

Vor der Ausführung des Skriptes muß ein Verzeichnis `out` angelegt werden. Im Skript wird angenommen, daß `out` und das OpenSSL-Quell-Verzeichnis im selben Verzeichnis liegen, z.B. `/usr/src`. Außerdem muß in der Datei `openssl-SNAP-2000mmdd/apps/app_rand.c` eine kleine Änderung vorgenommen werden:

Die folgende Zeile in `app_rand.c`

```
#include "apps.h"
```

ersetzen durch

```
#define NON_MAIN
#include "apps.h"
#undef NON_MAIN
```

Nach dem Wechsel in das Verzeichnis `/usr/src` und Anlegen des Verzeichnisses `/usr/src/out` kann das folgende Skript ausgeführt werden:

```
#!/bin/sh

FLAGS='-mv8 -O3 -fomit-frame-pointer -Icrypto -Iinclude -Wall'

cd openssl-SNAP-2000mmdd

for i in speed verify version ; do
      gcc ${FLAGS} -o ../out/$i apps/$i.c libcrypto.a
done

for i in asn1pars crl crl2p7 dgst dh dsa enc nseq pkcs7 pkcs8 rsa ; do
      gcc ${FLAGS} -o ../out/$i apps/$i.c apps/apps.c libcrypto.a
done

mv ../out/asn1pars ../out/asn1parse
mv ../out/crl2p7 ../out/crl2pkcs7

gcc ${FLAGS} -o ../out/errstr apps/errstr.c apps/apps.c \
    libssl.a libcrypto.a

for i in ciphers sess_id ; do
      gcc ${FLAGS} -o ../out/$i apps/$i.c apps/apps.c \
          libssl.a libcrypto.a -lsocket
done

for i in ca dhparam dsaparam gendh gendsa genrsa passwd pkcs12        \
         req smime x509 ; do
      gcc ${FLAGS} -o ../out/$i apps/$i.c apps/apps.c apps/app_rand.c \
          libssl.a libcrypto.a
done

gcc ${FLAGS} -o ../out/s_time apps/s_time.c apps/app_rand.c ap-
ps/s_cb.c \
    libssl.a libcrypto.a -lsocket -lnsl

for i in s_client s_server ; do
    gcc ${FLAGS} -o ../out/$i  apps/$i.c             \
         apps/app_rand.c apps/s_cb.c apps/s_socket.c \
         libssl.a libcrypto.a -lsocket -lnsl
done

cd ..
```

Für die `spkac`-Applikationen war die Übersetzung nicht erfolgreich.

Die Installation erfolgt durch Kopieren der Dateien. Dazu kann wie in der zweiten Hälfte des nächsten Abschnitts (7.2.3) beschrieben vorgegangen werden. Dabei muß allerdings der dritte Punkt ersetzt werden. Statt

```
cp apps/openssl tools/c_rehash $(SSLDIR)/bin/
```

muß folgendes Kommando gegeben werden:

```
cp ../out/* tools/c_rehash $(SSLDIR)/bin/
```

7.2.3 Installation

Nachdem das Paket übersetzt wurde, kann es jetzt installiert werden. Zunächst wird eine automatischen Installation beschrieben, und dann eine alternative Installation durch Kopieren. Die Programmsammlung kann nicht automatisch installiert werden. Sie muß durch Kopieren Installiert werden.

Die automatische Installation erfolgt durch das Kommando

```
make install
```

Achtung:
In `$(SSLDIR)` muß noch das Verzeichnis `newcerts` angelegt werden. Der `install`-Befehl erzeugt es nicht. Das Verzeichnis wird aber von der Default-Konfigurationsdatei `openssl.cnf` (s.u.) verlangt, um dort die erzeugten Zertifikate abzulegen.

Jetzt muß noch eine Datei angelegt werden, die die aktuelle Seriennummer des herauszugebenden Zertifikats in hexadezimaler Form enthält:

```
echo "01" > $(SSLETC)/serial
```

Dann muß noch eine Indexdatei für die erzeugten Zertifikate angelegt werden:

```
touch $(SSLETC)/index.txt
```

Die beiden Dateien `serial` und `index.txt` werden nach jeder erfolgreichen Zertifizierung eines Requests durch das Programm `ca` geändert, d.h. die Seriennummer in `serial` wird um den Wert eins erhöht, und in `index.txt` wird das herausgegebene Zertifikat registriert (siehe `index.txt (11.1)`).

Es ist wichtig, daß die OpenSSL-Konfigurationsdatei `$(SSL_DIR)/lib/openssl.cnf` vor dem Benutzen der OpenSSL-Applikationen durchgesehen und den Erfordernissen angepaßt wird. (Siehe Beispiel im Anhang `opsenssl.cnf (E)`.)

Wer etwas mehr Kontrolle über die Installation haben will, kann sich an folgendes Verfahren halten (Installation durch Kopieren).

Die ausführbaren Dateien, die Header-Dateien, die Bibliotheken und die Manual-Seiten können in eine vorhandene Verzeichnisstruktur kopiert werden. Das können die entsprechenden Verzeichnisse in `/usr/local` sein (`bin`, `include`, `lib` und `man`). Für die Installation der Konfigurationsdatei und der Verzeichnisse zur Zertifikatverwaltung muß zunächst ein Verzeichnis erzeugt werden, z.B. `/usr/local/etc/ssl`. Der Pfad dieses Verzeichnisses sollte günstigerweise mit dem vor der Kompilierung angegebenen (`--openssldir`) übereinstimmen. Andernfalls funktionieren einige Applikationen nicht ganz vollständig, da der Pfad in die Bibliothek `libcrypto.a` einkompiliert wird. In diesem Verzeichnis werden dann die Verzeichnisse `crl`, `certs`, `newcerts`, `misc` und `private` erzeugt. Anschließend werden die Dateien kopiert:

$(SSLDIR) = Installationspfad für Programmdateien
$(SSLETC) = Installationspfad für Dateien zur Zertifikatverwaltung

- `cp libcrypto.a libssl.a $(SSLDIR)/lib/`
- `chmod 644 $(SSLDIR)/lib/*`
- `cp apps/openssl tools/c_rehash $(SSLDIR)/bin/`
- `chmod 755 $(SSLDIR)/bin/*`
- `cp -r include/openssl $(SSLDIR)/include/`
- `rsaref.h` wird nicht benötigt und kann gelöscht werden:
  `rm $(SSLDIR)/include/openssl/rsaref.h`
- `chmod 644 $(SSLDIR)/include/*`
- `cp tools/c_i* tools/c_hash tools/c_name apps/CA.pl apps/CA.sh apps/der_chop $(SSLETC)/misc/`
- `chmod 755 $(SSLETC)/misc/*`
- `cp apps/openssl.cnf $(SSLETC)`
- `chmod 644 $(SSLETC)/openssl.cnf`

Die Installation der Manual-Seiten ist etwas aufwendiger, da sie im `pod`-Format vorliegen. Sie müssen durch das Perl-Skript `pod2man` in Manual-Seiten gewandelt werden. Das kann durch Aufruf des entsprechenden Abschnitts im Makefile geschehen:

```
make install_docs
```

Alternativ kann auch das folgende Shell-Skript aufgerufen werden, welches eine Adaption des `install_docs`-Abschnitts ist. Die einzelnen Manual-Sektionen werden durch das Skript in `/tmp` installiert und können dann von `root` in das gewünschte Verzeichnis kopiert werden.

```
#!/bin/sh

VERSION=0.9.5-dev
TOP=/usr/src/openssl-SNAP-2000mmdd

mkdir /tmp/man1 /tmp/man3 tmp/man5 /tmp/man7

for i in ${TOP}/doc/apps/*.pod; do
    cd `dirname $i`
    fn=`basename $i .pod`
    sec=`[ "$fn" = "config" ] && echo 5 || echo 1`
    perl ${TOP}/util/pod2man.pl --section=$sec --center=OpenSSL \
            --release=${VERSION} `basename $i` \
            >  /tmp/man$sec/`basename $i .pod`.$sec
done

for i in ${TOP}/doc/crypto/*.pod ${TOP}/doc/ssl/*.pod; do
    cd `dirname $i`
    fn=`basename $i .pod`
    sec=`[ "$fn" = "des_modes" ] && echo 7 || echo 3`
    perl ${TOP}/util/pod2man.pl --section=$sec --center=OpenSSL \
            --release=${VERSION} `basename $i` \
            >  /tmp/man$sec/`basename $i .pod`.$sec
done
```

Nun können die Seiten von `root` kopiert werden:

```
#!/bin/sh
for i in 1 3 5 7 ; do cp /tmp/man${i}/* /usr/local/man${i}/ ; end
```

Jetzt muß noch eine Datei angelegt werden, die die aktuelle Seriennummer des herauszugebenden Zertifikats in hexadezimaler Form enthält:

```
echo "01" > $(SSLETC)/serial
```

Dann muß noch eine Indexdatei für die erzeugten Zertifikate angelegt werden:

```
touch $(SSLETC)/index.txt
```

Kapitel 8

Unterstützte Zertifikaterweiterungen

Zertifikaterweiterungen (*Extensions*) sind von großer Bedeutung für den Umgang mit Zertifikaten. Die Extensions steuern den Verwendungszweck von Zertifikaten, sofern die Anwendungssoftware diese Extensions korrekt interpretiert. Leider ist es so, daß Anwendungssoftware und Zertifikat-Herausgeber die Verwendung und Bedeutung identischer Extensions teilweise recht unterschiedlich interpretieren. Genaueres zu dieser Problematik findet sich im lesenswerten *X.509 Style Guide*[1] von PETER GUTMANN.

Üblicherweise werden Extensions durch eine (P)CA beim Signieren eines Request in das dann erstellte Zertifikat gebracht. Die Bedeutung der „Certificate Standard Extensions" wird im RFC 2459 [FHPS99] beschrieben. Für den Betrieb einer CA ist es *unumgänglich*, sich mit diesen Extensions auseinanderzusetzen.

OpenSSL in der Version 0.9.5-dev unterstützt die folgenden *X.509v3-Extensions*:

- *Standard Certificate Extensions*
  - Basic Constraints (siehe Abschnitt 8.2)
  - Key Usage (siehe Abschnitt 8.3)
  - Extended Key Usage (siehe Abschnitt 8.4)
  - Subject Key Identifier (siehe Abschnitt 8.5)
  - Authority Key Identifier (siehe Abschnitt 8.6)
  - Subject Alternative Name (siehe Abschnitt 8.7)
  - Issuer Alternative Name (siehe Abschnitt 8.8)
  - Certificate Policies (siehe Abschnitt 8.9)
  - CRL Distribution Points (siehe Abschnitt 8.10)
- Netscape Certificate Extensions[2] (proprietär) (siehe Abschnitt 8.11)

[1] `http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt`
[2] `http://home.netscape.com/eng/security/certs.html`

Es ist auch möglich, eigene Erweiterungen zu registrieren (was Netscape mit den „Netscape Certificate Extensions“ gemacht hat). Solange diese Erweiterungen nicht als „Critical“ markiert sind, sollte jede Anwendung, die diese Erweiterungen nicht kennt, diese ignorieren (das Zertifikat also akzeptieren).

Extensions, die von OpenSSL (noch) nicht direkt unterstützt werden, können in ihrer hexadezimalen Kodierung angegeben werden. Dazu müssen der *object identifier* (OID) und die ASN.1-Syntax der Extension bekannt sein. Die ASN.1-Syntax wird dann in ihrer DER-Form (*distinguished encoding rules*, beschrieben in ITU-T-Empfehlung X.208) als hexadezimale Kodierung direkt angegeben. Auf diese Möglichkeit wird hier nicht weiter eingegangen. Genaueres dazu findet sich in der Datei `openssl.txt` im Verzeichnis `doc` der OpenSSL-Quellen.

8.1 Critical Bit

Das *Critical Bit* ist ein Flag, das für die meisten Extensions im Zertifikat gesetzt werden kann. Es wird beim Signieren durch eine CA zusammen mit der Extension gesetzt. Bei korrekter Implementierung einer Anwendung (z.B. eines Browsers) *muß diese eine als Critical markierte Extension interpretieren* können. Ist die Anwendung dazu nicht in der Lage (die Anwendung „kennt“ die Extension nicht), hat sie das Zertifikat, das diese Extension enthält, zurückzuweisen. Auch dann, wenn das Zertifikat technisch korrekt ist. Das Critical Bit soll also eine Möglichkeit bieten, eine bestimmte Verwendung eines Zertifikats zu erzwingen.

Da Zertifikate, die dieses Bit gesetzt haben, zurückgewiesen werden können, sollte der Einsatz des Critical Bit gut überlegt werden. Es gibt beispielsweise verschiedene Ansichten über die Verwendung der einzelnen Key Usage Attribute (s.u. (8.3)), so daß eine Critical-Markierung hier nicht ohne „Risiko“ ist.

8.2 Basic Constraints

Mittels Basic Constraints kann eine Anwendung erkennen, ob es sich bei einem Zertifikat um ein CA-Zertifikat handelt oder nicht. Diese Extension sollte in jedem CA-Zertifikat verwendet und als Critical markiert werden, auch wenn möglicherweise einige Anwendungen wegen der Critical-Markierung das Zertifikat zurückweisen.

Basic Constraints besteht aus einem Feld `cA`, welches ein BOOLEAN ist, sowie einem optionalen INTEGER-Feld, `pathLenConstraint`. Für CA-Zertifikate muß das `cA`-Feld auf TRUE gesetzt werden, für andere auf FALSE. Laut RFC 2459 sollte Basic Constraints in Nicht-CA-Zertifikaten nicht verwendet werden, also auch nicht, wenn die Extension FALSE markiert ist. `pathLenConstraint` ist nur sinnvoll in CA-Zertifikaten und gibt an, wieviele CA-Ebenen unterhalb des CA-Zertifikats maximal zulässig sind. Ein Wert 0 bedeutet hierbei, diese CA gibt nur Anwendungs- bzw. Benutzerzertifikate heraus. Basic Constraints sollte immer als Critical markiert werden.

Laut STEPHEN N. HENSON[3] ist die Basic Constraints Extension unbedingt erforderlich in CA-Zertifikaten, die S/MIME Benutzer zertifizieren. Andernfalls wird die S/MIME-Mail beim Empfänger aufgrund eines ungültigen CA-Zertifikats zurückgewiesen.

Sowohl der Netscape- als auch der Microsoft-Browser interpretieren die Extension. (HENSON hat allerdings die Beobachtung gemacht[4], daß „Microsoft Outlook Express 98“ grundsätzlich Schwierigkeiten mit Critical markierten Extension hat: *“Unfortunately Microsoft Outlook 98 chokes on critical extensions...”*.) Outlook Express 5 scheint diese Probleme nicht mehr zu haben.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
basicConstraints = [critical,] CA:<TRUE|FALSE>[, pathlen:n]
```

8.3 Key Usage

Key Usage steuert den Verwendungzweck des zu einem Zertifikat gehörendenen Schlüssels: z.B. darf ein Schlüssel nur zum Signieren von CRL’s, nur zur Daten-Verschlüsselung oder nur zum Unterschreiben verwendet werden. Laut Netscape Dokumentation vom 13.08.97[5] wird Key Usage vom Netscape Browser (NSC) zur Beschränkung der Verwendung von Zertifikaten ausgewertet. Allerdings nur, wenn Key Usage als **Critical** (s.u. (8.1)) markiert ist. Liegen andererseits mehrere Zertifikate vor (z.B. eines zum Signieren, eines zum Verschlüsseln), bestimmt Key Usage (Critical oder nicht), welches Zertifikat vom Browser verwendet wird.

Laut Microsoft-Dokumentation[6] wertet der MS Internet Explorer (MSIE) Key Usage aus, egal ob Critical oder nicht. Das deckt sich aber nicht ganz mit den gemachten Erfahrungen (siehe 12.4).

Üblicherweise wird Key Usage eingesetzt, um die Verwendung des Public Keys (und somit auch des Private Keys), an den diese Extension durch die Zertifizierung gebunden wird, zu beschränken. *Das funktioniert natürlich nur, wenn die Applikation, mit der das Zertifikat verwendet wird, diese Extension auch interpretiert*. Der Netscape Browser (Version 4.06) beispielsweise verweigert den Kontakt zu einem SSL-Server, wenn im Server-Zertifikat das Flag für Digital Signature gesetzt *und* dieses Critical markiert ist. Laut Netscape-Dokumentation sollte dieses Flag in einem SSL-*Client*-Zertifikat gesetzt sein. Für einen SSL-Server hätte dagegen Key Encipherment gesetzt sein müssen. Der Browser läßt daher keine SSL-Verbindung zu und bricht den Verbindungswunsch mit der Meldung “The certificate is not approved for the attempted operation” ab. Dasselbe Zertifikat wurde aber vom Microsoft-Browser (Ver. 4.01) problemlos akzeptiert. In der Microsoft-Dokumentation steht im Abschnitt zum Thema Key Usage: “The only Microsoft Application that currently enforces KeyUsage is Microsoft Outlook.”

---

[3] `http://www.drh-consultancy.demon.co.uk/caornot.html`

[4] `http://www.drh-consultancy.demon.co.uk/ca-fix.html`

[5] *Netscape Certificate Extensions – Communicator 4.0 Version*, `http://home.netscape.com/eng/security/comm4-cert-exts.html`

[6] *Structuring X.509 Certificates for Use with Microsoft Products*, `http://www.microsoft.com/security/ca/structuring.htm`

Die nachstehende Beschreibung erfolgt in Anlehnung an die oben erwähnte Netscape-Dokumentation und RFC 2459. Es stehen neun Werte für das Schlüsselwort `keyUsage` in der Konfigurationsdatei `openssl.cnf` zur Verfügung:

Bezeichnung	Wert für `keyUsage` in `openssl.cnf`	Verwendung des Public Keys
Decipher Only	`decipherOnly`	Ist *Key Agreement* gesetzt, darf der Public-Key innerhalb eines Schlüsselaustausches zur Entschlüsselung von Daten verwendet werden. Andernfalls undefiniert.
Encipher Only	`encipherOnly`	Ist *Key Agreement* gesetzt, darf der Public-Key innerhalb eines Schlüsselaustausches zur Verschlüsselung von Daten verwendet werden. Andernfalls undefiniert.
CRL Signing	`cRLSign`	Public Key kann verwendet werden, um CRLs zu verifizieren.
Key Cert Sign	`keyCertSign`	Public Key kann verwendet werden, um Zertifikate zu verifizieren.
Key Agreement	`keyAgreement`	Zur Verwendung beim Schlüsselaustausch.
Data Encipherment	`dataEncipherment`	Zur Verschlüsselung von „normalen“ Daten, also keinen Schlüsseln.
Key Encipherment	`keyEncipherment`	Public Key wird zum Schlüsselmanagement verwendet.
Non Repudiation	`nonRepudiation`	Key zur Prüfung von „bewußten“ Signaturen (außer CRLs und bei Zertifikaten).
Digital Signature	`digitalSignature`	Key zur Prüfung von „automatisierten“ Signaturen (außer bei CRLs und bei Zertifikaten).

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der der Konfigurationsdatei:

`keyUsage = [critical,]` *ein oder mehrere der obigen Bezeichner (mittlere Spalte)*

8.4 Extended Key Usage

Extended Key Usage kann ergänzend oder anstelle von Key Usage verwendet werden. Die Extension beschränkt analog Key Usage die Verwendung des zertifizierten Public-Keys. Beispielsweise könnte die Verwendung von CA-Zertifikaten, welche die entsprechende Basic Constraints und Key Usage Extension enthalten, feiner unterschieden werden. Extended Key Usage könnte dazu auf `serverAuth` gesetzt sein, so daß der CA-Schlüssel lediglich zum Überprüfen von SSL-Server- aber nicht von SSL-Client-Zertifikaten dient.

Auch Extended Key Usage kann `critical` gesetzt sein und es gilt das im Abschnitt Critical Bit (8.1) gesagte.

Jede Organisation kann eigene Werte für die Extension Extended Key Usage registrieren lassen. Unter anderem Netscape und Microsoft haben das getan.

OpenSSL kennt die in der Tabelle aufgeführten symbolischen Werte für das Schlüsselwort `extendedKeyUsage`. Es können aber auch andere Werte für die Extension durch Angabe der entsprechenden OID gesetzt werden.

Bezeichnung	Wert für `extKeyUsage` in `openssl.cnf`	Verwendung des Public Keys
RFC2459-Extensions		
TLS Web Server Authentication	`serverAuth`	Authentisierung von Web-Servern durch Web-Clients
TLS Web Client Authentication	`clientAuth`	Authentisierung von Web-Clients durch Web-Server
Code Signing	`codeSigning`	Key zur Signierung von Programm-Code
Email Protection	`emailProtection`	Key zur Verwendung mit S/MIME-Software
Time Stamping	`timeStamping`	Signierung von Objekt-Hashwerten und zugehörigen vertrauenswürdigen Zeitstempeln
Microsoft-Extensions		
Individual Code Signing	`msCodeInd`	
Commercial Code Signing	`msCodeCom`	
Trust List Signing	`msCTLSign`	
Server Gated Crypto	`msSGC`	Server-Zertifikat mit „Global Server ID“
Encrypted File System	`msEFS`	Verschlüsselung von symmetrischen Keys zur Dateissystem-Verschlüsselung
Netscape-Extensions		
Server Gated Crypto	`nsSGC`	Server-Zertifikat mit „Global Server ID“

In der Microsoft-Dokumentation (s.o.) steht, daß für den Einsatz von Zertifikaten im Zusammenhang mit Microsofts „Authenticode“ nur der Wert `codeSigning` für Extended Key Usage angegeben sein darf. Ebenfalls in dieser Dokumentation steht, daß die Gültigkeit von Extended Key Usage im Anwendungszertifikat nur gegeben ist, *wenn sämtliche Zertifikate der Zertifikatkette diese Extension enthalten*. Ist die Extension dagegen garnicht enthalten, sollen MS-Anwendungen das Zertifikat für jeden durch Key Usage beschriebenen Zweck als gültig interpretieren.

Auch Netscape scheint das Setzen der Extension in allen Zertifikaten der Kette zu unterstützen. Es scheint aber doch fraglich, wieviel Sinn es beispielsweise macht, in einen CA-Zertifikat Extended Key Usage auf `email` zu setzen, damit die Extension in einem Anwendungszertifikat gültig ist. Die Empfehlungen von Netscape für diese Extension können im *Installation and Deployment Guide*[7], Appendix B, nachgelesen werden.

Zu Netscapes bzw. Microsofts SGC (“Server Gated Cryptography”) ist anzumerken, daß die Verwendung dieser Werte nur im Zusammenhang mit speziellen CA-Zertifikaten sinnvoll ist. Diese CA-Zertifikate werden durch ein Flag *im Browser* als für SGC geeignet gekennzeichnet. Ein eigenes CA-Zertifikat kann nur nach Import in den Browser und anschließendem Patchen der Zertifikat-Datenbank (Netscape) mit diesem Flag ausgestattet werden. Genaueres steht in der Datei `README.GlobalID` des SSL-Apache-Pakets[8].

Eine Empfehlung für die Werte dieser Extension ist nur schwer zu geben, gerade wegen der MS-Forderung, daß die Extension in allen Zertifikaten der Kette enthalten sein muß. Am sinnvollsten scheint es zu sein, ganz auf diese Extension zu verzichten.

[7] `http://developer.netscape.com/docs/manuals/cms/41/dep_gide/ext.htm`

[8] `http://www.modssl.org/`

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

`keyUsage = [critical,]` *ein oder mehrere der obigen Bezeichner (mittlere Spalte)* `[, OID]`

8.5 Subject Key Identifier

Die Extension Subject Key Identifier enthält den Hash-Wert des Public Keys eines Zertifikates. Dadurch kann ein zu einem Public Key gehörendes Zertifikat effizient gesucht werden. So wird die Überprüfung von Zertifikatketten und bei mehreren Anwendungszertifikaten die Auswahl des richtigen Zertifikats unterstützt. Diese Extension sollte sowohl in CA-Zertifikaten als auch in Anwendungszertifikaten enthalten sein.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
subjectKeyIdentifier = hash
```

8.6 Authority Key Identifier

Die Extension Authority Key Identifier besteht aus drei Feldern: `keyIdentifier`, `authorityCertIssuer` und `authorityCertSerialNumber`. Laut RFC 2459 kann ein Schlüssel mittels dieser Extension auf zwei Arten identifiziert werden: entweder durch alleiniges Setzen des `keyIdentifier`-Feldes, oder durch Setzen der anderen beiden Felder. Diese Extension unterstützt die Überprüfung von Zertifikatketten.

Microsoft empfiehlt die zweite Variante, damit eine Zertifikatkette überprüft werden kann. RFC 2459 empfiehlt die erste Variante.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
authorityKeyIdentifier = <[keyid[:always]][, issuer[:always]]>
```

Ist `keyid` gesetzt, wird Subject Key Identifier (siehe oben (8.5)) des Herausgeber-Zertifikats kopiert. Kann der Wert dieser Extension nicht kopiert werden (weil Subject Key Identifier nicht gesetzt war) und ist `keyid` auf `always` gesetzt, wird mit einer Fehlermeldung abgebrochen. Ist `keyid` nicht `always` gesetzt, werden alternativ das Issuer-Feld und die Seriennummer kopiert. Ist `issuer` auf `always` gesetzt, werden immer das Issuer-Feld und die Seriennummer kopiert.

8.7 Subject Alternative Name

Die Extension Subject Alternative Name kann verwendet werden, um weitere Bezeichner für ein Subject in das Zertifikat zu bringen. Es sind E-Mail-Adressen, DNS-Namen, IP-Adressen, URIs und registrierte IDs (RIDs), auch mehrfach, möglich. Diese können auch beliebig kombiniert werden.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
subjectAltName=<[email:<copy|name@mail.de>] [,
URL:http://my.url.here/] [, RID:1.2.3.4][, IP:1.2.3.4]>
```

8.8 Issuer Alternative Name

Die Extension Issuer Alternative Name ist wie Subject Alternative Name (8.7) aufgebaut.

Hier wird allerdings nicht `email:copy` sondern `issuer:copy` unterstützt. Dabei werden die Angaben vom Subject Alternative Name des *Herausgeber*-Zertifikats kopiert.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
subjectAltName=<[issuer:copy][, URL:http://my.url.here/][,
RID:1.2.3.4][, IP:1.2.3.4]>
```

8.9 Certificate Policies

Die Extension Certificate Policies verweist auf die Policy, unter der ein Zertifikat herausgegeben wurde. Laut RFC 2459 sollte die Extension lediglich aus einer OID bestehen. Eine Anwendung, die Zertifikate prüft, sollte eine Liste mit den OIDs akzeptierter Policies enthalten und diese gegen die Policy-OID eines Zertifikats vergleichen. Dann wird das Zertifikat entsprechend akzeptiert oder zurückgewiesen.

Wenn die OID nicht ausreichend ist (weil z.B. die eingesetzten Applikation keine OID-Listen unterstützen), beschreibt RFC 2459 die Möglichkeit einen URI anzugeben, der auf die Policy der CA verweist (*Certification Practice Statement, CPS*). Für Anwendungszertifikate und CA-Zertifikate, die an andere Organisationen (also nicht die der Herausgeber-CA) herausgegeben werden, kann zusätzlich eine *User Notice* festgelegt werden.

Die User Notice besteht aus zwei optionalen Feldern: `noticeRef` und `explicitText`. Das erste Feld `noticeRef` besteht aus dem Organisations-Namen und einer Nummer. Der Nummer ist ein Text zugeordnet und soll wieder durch eine Anwendung mit einer entsprechen Liste interpretiert werden. Findet eine Anwendung diese Nummer in ihrer Liste, soll sie den zugehörigen Text anzeigen. Das zweite Feld enthält einen, maximal 200 Zeichen umfassenden, frei gestaltbaren Text und steht direkt im Zertifikat. Eine Anwendung soll diesen dann bei Verwendung des Zertifikats anzeigen.

Microsoft-Anwendungen scheinen mit der Extension Schwierigkeiten zu haben, wie PETER GUTMANN in seinem *X.509 Style Guide* beschreibt:

> "Although various MS programs give the impression of handling certificate policies, they only have a single hardcoded policy which is the Verisign CPS. To see an example of this, create a certificate with a policy of (for example) 'This policy isn't worth the paper it's not written on' and view the cert using Outlook Express. What's displayed will be the Verisign CPS."

Outlook Express 5 und der Internet Explorer 5 scheinen diese Probleme nicht mehr zu haben. Eine User Notice wird für Benutzerzertifikate korrekt angezeigt. Die URL für das CPS wird ebenfalls ausgewertet und die entsprechende Seite angezeigt.

(Einzelheiten siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
certificatePolicies=[ia5org,]OID[, OID, ...][,@polsect]
```

Laut OpenSSL-Dokumentation von STEPHEN N. HENSON (im Quell-Verzeichnis des OpenSSL-Pakets unter `doc/openssl.txt`) ist die `ia5org`-Option für die Verwendung mit dem Internet Explorer erforderlich, obwohl sie nicht RFC-konform ist.

Die Option `@polsect` verweist auf einen Abschnitt in der Konfigurationsdatei, der die Werte für das CPS und indirekt für das User Notice Feld enthält:

```
[ polsect ]
policyIdentifier=OID
CPS="http://www.ca.de/policy.html"
userNotice=@notice

[ notice ]
explicitText="Nur zur Verschluesselung von E-Mail (S/MIME)"
organisation="CA Org."
noticeNumbers=4, 2
```

8.10 CRL Distribution Points

Die Extension CRL Distribution Points legt fest, wo eine Widerrufliste (*CRL*) der Herausgeber-CA abgerufen werden kann. Grundsätzlich sind laut RFC 2459 alle Optionen, die bei Subject Alternative

Name (siehe 8.7) verwendet werden können, auch hier einsetzbar. OpenSSL unterstützt allerdings bisher nur die Verwendung eines URIs. Dieser muß aus der absoluten Adresse der CRL bestehen.

(Siehe RFC 2459 [FHPS99] oder die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
crlDistributionPoints=URI:http://www.ca.de/ca1.crl [,
URI:http://www.ca.de/ca2.crl ... ]
```

8.11 Netscape Certificate Extensions

Zu den *Netscape Certificate Extensions*[9] ist anzumerken, daß diese nicht Critical gesetzt werden sollten. Ein SSL-Server-Zertifikat, in dem beispielsweise das Netscape-SSL-Server-Bit (`nsCertType=critical,server`) Critical gesetzt ist, wird von einem Microsoft-Browser (der diese Extension nicht kennt) zurückgewiesen werden. Ein solches Server-Zertifikat würde also eine SSL-Verbindung von Microsoft-Clients zum Server ausschließen.

Folgende Netscape-Erweiterungen werden von OpenSSL-0.9.5-dev unterstützt (die Schlüsselwörter für die OpenSSL-Konfigurationsdatei stehen in Klammern):

- netscape-cert-type (`nsCertType`)
- netscape-base-url (`nsBaseUrl`)
- netscape-revocation-url (`nsRevocationUrl`)
- netscape-ca-revocation-url (`nsCaRevocationUrl`)
- netscape-cert-renewal-url (`nsRenewalUrl`)
- netscape-ca-policy-url (`nsCaPolicyUrl`)
- netscape-ssl-server-name (`nsSslServerName`)
- netscape-comment (`nsComment`)

Wichtig ist die Erweiterung Netscape Cert Type, um für die mit OpenSSL erzeugten Zertifikate den Verwendungszweck (zumindest für Netscape-Browser) zu beeinflussen. Der Internet-Explorer scheint die Netscape-Extensions zu ignorieren. Für die Erweiterung Netscape Cert Type stehen die folgenden acht Werte für das Schlüsselwort `nsCertType` in der Konfigurationsdatei zur Verfügung:

[9] `http://home.netscape.com/eng/security/certs.html`

Wert für `nsCertType` in `openssl.cnf`	Bedeutung
`objCA`	Ein CA-Zertifikat um Zertifikate zum Signieren von Objekten (Java-Applets etc.) herauszugeben.
`emailCA`	Ein CA-Zertifikat um E-Mail-Benutzer zu zertifizieren
`sslCA`	Ein CA-Zertifikat um Server oder Clients zu zertifizieren
`reserved`	Reserviert für zukünftige Nutzung
`objsign`	Zertifikat um Objekte (Java-Applets etc.) zu signieren
`email`	Ein E-Mail Benutzer-Zertifikat
`server`	Ein SSL-Server-Zertifikat
`client`	Ein SSL-Client-Zertifikat

Es sind auch Kombinationen der einzelnen Bezeichnungen möglich, so steht z.B. `nsCertType=objCA, emailCA, sslCA` für ein CA-Zertifikat, mit dem Zertifikate für Objekt-Signierung, S/MIME-Benutzer und SSL-Server/-Clients herausgegeben werden können. Die Bedeutung der anderen Attribute kann Anhang E entnommen werden.

Kapitel 9

Die OpenSSL-Konfigurationsdatei

Eine Beispiel-Konfigurationsdatei findet sich im Anhang `openssl.cnf (E)`.

Die Belegung von Aufruf-Optionen der OpenSSL-Applikationen wird weitestgehend durch die Konfigurationsdatei `$(SSLETC)/openssl.cnf` bestimmt. Sie ist ähnlich aufgebaut wie eine Windows-Ini-Datei. Einzelne Abschnitte sind durch Bezeichner der Form `[ uvwxyz ]` gekennzeichnet. Innerhalb dieser Abschnitte können Variablen vereinbart werden. Dabei ist es auch möglich, Umgebungsvariablen zu überschreiben, z.B. mit

```
ENV::PATH = /usr/local/bin:$PATH
```

Außerdem können Werte von gesetzten Umgebungsvariablen einzelnen Variablen der Konfigurationsdateien zugeordnet werden, z.B. mit

```
ca_default = $ENV::ENV_CA_DEFAULT
```

Enthält die Umgebungsvariable `ENV_CA_DEFAULT` den Wert `Server_CA`, wird bei der nächsten Zertifizierung durch die Applikation `ca` der Abschnitt `ca_default` gewählt. Analog könnte eine Umgebungsvariable `ENV_EXT`, die die Bezeichnung eines Abschnitts mit Zertifikat-Erweiterungen *Extensions* enthält, gesetzt werden:

```
x509_extensions = $ENV::ENV_EXT
```

Die Datei gliedert sich grob in zwei Abschnitte: `[ ca ]`, in dem Voreinstellungen für die Erzeugung eines Zertifikats vorgenommen werden, und entsprechend `[ req ]` für die Erzeugung eines „Zertifizierungswunsches“ (*Request*). Auf diese voreingestellten Abschnitte wird zugegriffen, wenn `openssl` mit dem Parameter `ca` bzw. `req` aufgerufen wird. (`openssl req` *Parameter...* **erzeugt** einen Request, `openssl ca` *Parameter...* **signiert** einen Request.)

Sowohl `ca` als auch `req` bieten die Möglichkeit, über die Option `-config` die zu verwendende Konfigurationsdatei explizit anzugeben. So könnte jeweils eine Konfigurationsdatei speziell für Netscape-Browser und eine andere für Microsoft-Browser gestaltet sein.

Es ist auch möglich, innerhalb einer Konfigurationsdatei mehrere CA-Abschnitte zu definieren, je nachdem, welche Art Request signiert werden soll. Dadurch kann innerhalb einer Konfigurationsdatei eine CA-Konfiguration zur Server-Zertifizierung (z.B. `[ Server_CA ]`), eine Konfiguration zur Client-Zertifizierung (z.B. `[ Client_CA ]`) und eine Konfiguration zur S/MIME-Zertifizierung[1] (z.B. `[ SMIME_CA ]`) verwaltet werden. Bei Aufruf von `openssl` mit dem Parameter `ca` kann dann mit der Option `-name Client_CA` z.B. auf eine Client-CA-Konfiguration zugegriffen werden. Alternativ kann die Auswahl des Konfigurations-Abschnitts durch die oben erwähnte Umgebungsvariable `ENV_CA_DEFAULT` erfolgen. (Statt der im letzten Absatz erwähnten zwei speziellen Konfigurationsdateien kann das gleiche auch über entsprechende Konfigurationsabschnitte innerhalb einer Datei erreicht werden.)

Sollen verschiedene Zertifikattypen durch eine CA herausgegeben werden, können unterschiedlich benannte Abschnitte, die die jeweiligen Extensions enthalten, in der Konfigurations-Datei angelegt werden. Der gewünschte Abschnitt wird dann über die Umgebungsvariable `ENV_EXT` ausgewählt, wenn in der Konfigurationsdatei `x509_extensions = $ENV::ENV_EXT` gesetzt ist. (Anm.: Das gleiche kann auch über die oben erwähnten speziellen Konfigurationsdateien erreicht werden.)

Innerhalb von CA-Abschnitten werden drei Schlüsselwörter erkannt, die auf weitere Abschnitte in einer Konfigurationsdatei verweisen. Die Schlüsselwörter lauten:

- `x509_extensions`
- `crl_extensions`
- `policy`

`x509_extensions` verweist auf einen Abschnitt, in dem Extensions für neue Zertifikate festgelegt sind. In diesem Abschnitt könnten die Extensions für ein Benutzer- oder ein CA-Zertifikat festgelegt werden.

`crl_extensions` verweist auf einen entsprechenden Abschnitt für Extensions, die in eine Zertifikat-Widerrufliste (*certificate revocation list, CRL*) gebracht werden sollen.

`policy` schließlich weist auf einen Abschnitt, in dem festgelegt wird, inwieweit der Name in einem zu signierenden Request mit dem des CA-Zertifikats übereinstimmen muß. Beispielsweise kann festgelegt werden, daß die Angaben zum Land übereinstimmen müssen.

Im `req`-Abschnitt werden vier Schlüsselwörter erkannt:

- `distinguished_name`
- `attributes`
- `x509_extensions`
- `req_extensions`

[1] `http://www.rsa.com/rsa/S-MIME/`

`distinguished_name` weist auf einen Abschnitt, in dem festgelegt wird, welche Namensfelder bei der Erzeugung des Requests abgefragt werden, sowie deren Default-Werte.

Über `attributes` werden optionale Felder festgelegt, die zusätzlich in den Request gebracht werden können. Diese dienen zum Widerruf eines Zertifikats bei Verlust des Private-Key. Genaueres steht im Abschnitt Erzeugen von Requests (10.2).

Auch im `req`-Abschnitt gibt es einen Bereich, auf den ein Schlüsselwort `x509_extensions` verweist. Im Unterschied zu `x509_extensions` im `ca`-Bereich, werden hier die Extensions festgelegt, die bei Erzeugung eines *selbstsignierten* Zertifikats (*Wurzel-* oder *Root-Zertifikat*) in ein solches Zertifikat gebracht werden.

Im vierten Bereich, auf den `req_extensions` weist, werden Extensions festgelegt, die in einen *Request* gebracht werden. Üblicherweise werden Extensions allerdings durch eine CA beim Signieren eines Requests in das Zertifikat gebracht. Es ist aber auch denkbar, daß der Zertifikatnehmer bei der Erzeugung eines Requests festlegt, welche Extensions sein Zertifikat enthalten soll. Eine CA könnte dann diese Extensions in das Zertifikat übernehmen. Es sollte unbedingt mit der CA abgesprochen werden, ob sie Requests mit Extensions signiert.

Ein ausführliches Beispiel einer Konfigurationsdatei mit mehreren Abschnitten findet sich im Anhang `openssl.cnf (E)`

Kapitel 10

Erzeugen von Requests und Zertifikaten

Die folgenden Abschnitte beschreiben die Erzeugung eines selbstsignierten Zertifikats, das sogenannte *Root-Zertifikat*, sowie Erzeugung und Signierung eines Requests. Dabei sollte beachtet werden, daß die Gültigkeitsdauer eines neuen Zertifikats vollständig innerhalb des Gültigkeitsbereiches des Zertifikats der signierenden CA liegt. MS-Produkte weisen sonst möglicherweise dieses neue Zertifikat als ungültig zurück. Weiter ist es empfehlenswert, nach Herausgabe eines CA-Zertifikats einen Tag zu warten, bevor der zugehörige Private-Key zur Zertifizierung eingesetzt wird.

Netscape- und Microsoft-Browser mit einer Versionsnummer kleiner als 4.0 *können nicht mit Schlüssellängen größer als 1024 Bit umgehen*. Das gilt sowohl für CA- als auch für Anwendungs-Zertifikate. Die Browser zeigen eine irreführende Meldung an, daß das Server-Zertifikat eine ungültige Signatur hat oder grundsätzlich fehlerhaft ist, obwohl das Zertifikat technisch korrekt ist. (Vgl. Anhang I.1)

Im Verzeichnis `$(SSLETC)/misc` gibt es ein Perl-Skript, `CA.pl`. Mit dem Skript können ebenfalls Zertifikate und Requests erzeugt werden. Das Skript soll den Einstieg in die Handhabung von Zertifikaten mit OpenSSL erleichtern, indem komplexe Kommandos und die Konfiguration gekapselt werden. Das Skript ist dokumentiert, muß aber möglicherweise noch angepaßt werden. Auf den Einsatz des Skriptes wird hier nicht weiter eingegangen.

10.1 Erzeugen eines Root-CA-Zertifikats

Die im obigen Abschnitt (8) aufgeführten Erweiterungen werden üblicherweise beim Zertifizieren eines Requests mit OpenSSL durch eine CA in das Zertifikat gebracht. In der Konfigurationsdatei (siehe `openssl.cnf (E)`) gibt es einen Abschnitt `[ v3_ca ]`, in dem festgelegt werden kann (und sollte!), welche Zertifikat-Erweiterungen bei Erzeugung eines (selbstsignierten) **Root-Zertifikats** gesetzt werden.

Im folgenden ein Beispiel zur Erzeugung eines Root-CA-Zertifikats:

1. Editieren von `openssl.cnf`, Setzen der Zertifikat-Erweiterungen im Abschnitt `[ v3_ca ]`. Um z.B. ein Root-CA-Zertifikat zu erzeugen, mit dem SSL-CA-, S/MIME-

CA- und Object-Signing-CA-Zertifikate herausgegeben werden können, sollte die Netscape Certificate Extension folgenderweise gesetzt sein:

```
nsCertType = sslCA, emailCA, objCA
```

2. Mögliche Werte für die anderen Extensions:

```
basicConstraints       = critical, CA:TRUE
keyUsage               = cRLSign, keyCertSign
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid, issuer:always
subjectAltName         = email:copy
issuerAltName          = issuer:copy
crlDistributionPoints  = URI:http://crlserver.domain.de/CA.crl
```

   Die unter 1. und 2. aufgeführten Einträge müssen in der Konfigurationsdatei in dem beim Schlüsselwort `x509_extensions` genannten Bereich gemacht werden, also z.B. im Bereich `[ v3_ca ]`. Das Schlüsselwort `x509_extensions` kann sowohl im `ca`-Abschnitt als auch im `req`-Abschnitt stehen. Für das selbstsignierte Zertifikat muß `x509_extensions` im Bereich `[ req ]` der Konfigurationsdatei auf `v3_ca` gesetzt werden.

3. Initialisieren des Zufallszahlen-Generator-Status. Dieser Status ist eine Datei, die durch jeden Zugriff verändert wird. Die Datei kann z.B. als `/usr/local/etc/ssl/private/.rand` engelegt werden. Es sollte die Umgebungsvariable `$RANDFILE` auf diese Datei gesetzt werden. Initialisiert wird der Status dann durch folgenden Befehl:

```
cp ~/.pgp/randseed.bin $RANDFILE
```

4. Erzeugen eines 2048-Bit-Schlüssels:

   Vor Aufruf dieses Befehls muß die Umgebungsvariable `$RANDFILE` gesetzt sein, sonst findet das folgende Kommando die Datei mit dem Zufallszahlen-Status nicht und ein einkompilierter Default wird wirksam.

```
openssl genrsa -des3 -out $(SSLETC)/private/CAkey.pem -rand
Zufallsdaten 2048
```

   Achtung:
   Ohne die Option `-des3` (oder `-des`, `-idea`) wird der Schlüssel unverschlüsselt abgespeichert!

5. Erzeugen des selbstsignierten Root-CA-Zertifikats:

```
openssl req -new -x509 -days 730 -key
$(SSLETC)/private/CAkey.pem -out $(SSLETC)/private/CAcert.pem
```

Achtung:
Über die Option `-days` wird die Gültigkeitsdauer des Root-Zertifikats in Tagen festgelegt. Beginn des Zeitraums ist der Zeitpunkt der Erzeugung des Zertifikats.

6. Anzeigen des erzeugten Zertifikats:

```
openssl x509 -in ./CAcert.pem -text | more
```

Das Zertifikat muß jetzt noch in das Verzeichnis `$(SSLETC)/certs` als `00.pem` kopiert werden. Der Name ergibt sich aus der hexadezimalen Seriennummer des Zertifikats (hier 00) und dem Suffix `.pem`. Anschließend sollte das Zertifikat noch über seinen Hash-Wert verlinkt werden (siehe Anmerkung unten).

- `cp $(SSLETC)/private/CAcert.pem $(SSLETC)/certs/00.pem`
- `cd $(SSLETC)/certs`
- ``ln -s 00.pem `openssl x509 -hash -noout -in 00.pem`.0``

Alternativ zu dem Link-Befehl kann auch das Shell-Skript `c_rehash` in `$(SSLETC)/bin` aufgerufen werden. Das Skript erzeugt für alle im Verzeichnis `certs` gefundenen Zertifikate die nötigen Links. Der Hash-Wert besteht aus 64 Bit und wird aus den Angaben des Subject-Feldes (also Distinguished Name und eventuell Email-Adresse des Herausgebers) des Zertifikats gebildet.

Anmerkung:
Alle neuen Zertifikate sollten über ihren Hash-Wert verlinkt werden. Das ist deshalb von Bedeutung, weil die Suche nach Zertifikaten und der damit verbundene Vergleich *ausschließlich* über die Hash-Werte der Zertifikate erfolgt.

10.2 Erzeugen eines Certificate Requests

Zur Erzeugung eines beliebigen Requests (CA, Server, Client) muß zunächst ein Schlüsselpaar generiert werden. Mit folgendem Befehl wird ein 1024 Bit Schlüssel erzeugt:

```
openssl genrsa -des3 -out MyKey.pem -rand file1:file2:... 1024
```

Vorher sollte allerdings der Zufallszahlen-Status initialisiert worden sein. Die nach der Option `-rand` angegebenen Dateien dienen als zusätzliche Zufallsdaten für die Schlüsselerzeugung. Abhängig vom später verwendeten Browser ist die Schlüssellänge zu beachten. Für den MS-Internet-Explorer in der *internationalen (Export-)Version* ist die Schlüssellänge für ein Benutzerzertifikat grundsätzlich auf 512 Bit begrenzt. Die `1024` im obigen Kommando muß dann durch `512` ersetzt werden. Für die Netscape Navigator/Communicator (NSC) Export-Version gilt im Prinzip die gleiche Beschränkung. Die Schlüssellänge kann aber bis auf 2048 Bit erhöht werden, indem das Zertifikat extern (z.B. mit `openssl`) erzeugt und anschließend als `.p12`-Datei (siehe HENSONS *PKCS#12-Seite*[1]) in den NSC importiert wird (siehe PFX (13.1) bzw. PKCS12 (12.2)). *Da die USA die Export-Beschränken für Kryptosoftware gelockert haben, sollten demnächst Browser-Versionen zur Verfügung stehen, die diese Beschränkung der Schlüssellänge nicht mehr haben.* (Siehe dazu Abschnitt 4.15.1.)

Die Schlüsselerzeugung hätte bei dem folgenden Request-Kommando gleichzeitig mit der Request-Erzeugung durch die Option `-newkey` veranlaßt werden können; diese Variante bietet aber keine Möglichkeit, zusätzliche Zufallsdaten anzugeben.

Die Erzeugung des Requests erfolgt durch folgenden Befehl:

```
openssl req -new -key MyKey.pem -out MyReq.pem
```

Nach Eingabe des Befehls müssen folgende Angaben gemacht werden (beispielhafte Eingaben sind in doppelten Anführungsstrichen):

```
Using configuration from /usr/local/etc/ssl/openssl.cnf
Enter PEM pass phrase:  "passphrase"
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: "DE"
State or Province Name (full name) [Some-State]: "Schleswig-Holstein"
Locality Name (eg, city) []: "Kiel"
Organization Name (eg, company) [Internet Widgits Pty Ltd]: "Universitaet Kiel"
Organizational Unit Name (eg, section) []: "Studis"
Common Name (eg, YOUR name) []: "Fred Neumann"
Email Address []: "neumann@inf.uni-kiel.de"

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: "passphrase vergessen"
An optional company name []: "Informatik Uni Kiel"
```

(Für den genauen Inhalt muß die zertifizierende CA bzw. deren Policy konsultiert werden.)

Die beiden letzten Angaben tauchen als Klartext im Attribut-Bereich des Requests auf. Soll später ein Zertifikat zurückgerufen werden, kann sich der Inhaber gegenüber der CA, auch bei Verlust des Private Keys, durch Angabe dieser Felder als Inhaber „ausweisen“. Die Verwaltung dieser Angaben kann von der CA durchgeführt werden.

Je nach Verwendungszweck des späteren Zertifikats muß bei den Angaben folgendes beachtet werden:

[1] `http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs-12.html`

- Verwendung als S/MIME-Zertifikat:

  Die E-Mail-Adresse muß vorhanden sein oder sie wird über die Extension Subject Alternative Name (8.7) gesetzt (Aufgabe der CA).

- Verwendung als SSL-Server-Zertifikat:

  Als CommonName muß der Server-Name angegeben werden, z.B. `www.site.com`. Ebenfalls sinnvoll ist es, den Server-Namen zusätzlich über die Extension `nsSslServerName` zu setzen (Aufgabe der CA).

- Verwendung als CA-Zertifikat:

  Keine Vorgaben. Sinnvoll wäre die Angabe der E-Mail-Adresse des CA-Administrators.

Es scheint Probleme mit dem Navigator und dem Internet-Explorer zu geben, wenn einige Zeichen wie '_' oder '`&`' im DN auftauchen. HENSON empfiehlt daher, nur Zeichen vom Typ *Printable String* (mit Ausnahme der E-Mail-Adresse) zu verwenden. Die so zulässige Menge an Zeichen umfaßt die folgenden:

```
A-Z a-z 0-9 ' ( ) + , - . / : = ? leerzeichen
```

Der erzeugte Request (also nur die Datei *MyReq.pem*, nicht die Datei mit dem Schlüsselpaar) kann dann auf Diskette kopiert und einer CA zum Signieren (also: Zertifizieren, siehe Abschnitt Signieren (10.3)) vorgelegt werden. Nach der Zertifizierung kann dann das Zertifikat zusammen mit dem privaten Schlüssel als `.p12`-Datei in den jeweiligen Browser importiert werden (siehe PFX (13.1) und PKCS12 (12.2)).

10.3 Signieren eines Certificate Requests

Das Signieren eines einzelnen Requests erfolgt durch folgenden Befehl:

```
openssl ca -name ca_name -keyfile CAkey.pem -in MyReq.pem -out
MyCert.pem -outdir $(SSLETC)/certs
```

Der Wert von *ca_name* steht hier für den gewünschten Abschnitt der Konfigurationsdatei, also z.B. `Client_CA`. Es ist auch möglich, Gültigkeitsbeginn und -ende für das Zertifikat festzulegen. Dazu muß das obige Kommando um die Optionen `-startdate` und `-enddate` erweitert werden. Als Parameter wird bei den Optionen ein UTC-Zeitstempel der Form `YYMMDDHHMMSSZ` angegeben. Wegen der UTC-Zeit muß bei der Angabe der Stunden während der Winterzeit eine Stunde, und während der Sommerzeit zwei Stunden zur lokalen Zeit (Deutschland), bzw. zu dem gewünschten Zeitpunkt, hinzugefügt werden.

Nach Eingabe des Befehls kommt eine Meldung folgender Art (Eingaben stehen in doppelten Anführungsstrichen):

```
Using configuration from /usr/local/etc/ssl/openssl.cnf
Enter PEM pass phrase: "passphrase"
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Schleswig-Holstein'
localityName          :PRINTABLE:'Kiel'
organizationName      :PRINTABLE:'Universitaet Kiel'
organizationalUnitName:PRINTABLE:'Studis'
commonName            :PRINTABLE:'Fred Neumann'
emailAddress          :IA5STRING:'neumann@inf.uni-kiel.de'
Certificate is to be certified until Feb 12 12:10:12 2001 GMT (365 days)
Sign the certificate? [y/n]: "y"

1 out of 1 certificate requests certified, commit? [y/n] "y"
Write out database with 1 new entries
Data Base Updated
```

Findet sich die Konfigurationsdatei (KD) *openssl.cnf* nicht in dem bei der Konfiguration des OpenSSL-Pakets angegebenen SSL-Verzeichnis, kann die Angabe im `ca`-Kommando mittels `-config` */pfad/Konfigname* erfolgen. Oder es wird die Umgebungsvariable `OPENSSL_CONF` auf */pfad/Konfigname* gesetzt. Wenn die Datei mit dem CA-Key in der KD angegeben ist, kann die Angabe bei `-keyfile` ebenfalls wegfallen. Mit der Option `-name` kann ein spezieller Eintrag in der KD, z.B. der Abschnitt `Server_CA`, abgearbeitet werden (siehe Anhang `openssl.cnf E`). Durch das Signieren werden möglicherweise einige Extensions in das Zertifikat gebracht. Vor dem Signieren eines Requests muß unbedingt sichergestellt sein, daß die richtigen Extensions gesetzt sind bzw. der richtige Abschnitt in der KD gewählt wird (siehe Abschnitt 10.3 und Anhang E).

Für ein S/MIME-Zertifikat könnte Key Usage (`keyUsage`) auf *Digital Signature* und *Data Encipherment* gesetzt werden, sowie Netscape Cert Type (`nsCertType`) auf *email*.

Sind in der KD keine anderen Angaben gemacht worden, befindet sich das neu erstellte Zertifikat in der bei `-out` angegebenen Datei. Außerdem wird im bei `-outdir` angegebenen Verzeichnis `$(SSLETC)/certs` eine Kopie abgelegt. Der Name der Kopie setzt sich aus der Seriennummer des Zertifikats und der Endung `.pem` zusammen, also z.B. `0a.pem`. Die Seriennummer des gerade herausgegebenen Zertifikats findet sich im Zertifikat selber (`openssl x509 -in myCert.pem -noout -serial`) oder in der Datei `$(SSLETC)/serial.old`. Das neue Zertifikat muß noch über den Hash-Wert verlinkt werden. Dazu wird in das Verzeichnis `$(SSLETC)/certs` gewechselt und das folgende Kommando gegeben:

```
ln -s 0a.pem `openssl x509 -in 0a.pem -hash -noout`.0
```

Kapitel 11

Certificate Revocation Lists mit OpenSSL

CRLs sind Listen, die Seriennummer und Zeitpunkt des Rückrufs eines Zertifikats enthalten, sowie den DN der CA. Sie werden von der CA signiert und in regelmäßigen Abständen veröffentlicht (siehe Abschnitt 2.5).

Auch CRLs können Extensions enthalten. Sie werden beim Signieren einer CRL durch eine CA in die CRL gebracht. OpenSSL unterstützt bisher die Extensions *CRL Authority Key Identifier (11.3)* und *CRL Issuer Alternative Name (11.4)*. Die Verwendung dieser Extensions zeichnet (u.a.) eine X.509*v2*-CRL aus. Werden diese Extensions nicht verwendet (durch Löschen oder Auskommentieren des Schlüsselworts `crl_extensions` in `openssl.cnf`), erzeugt obiges Kommando eine X.509*v1*-CRL. Das ist deshalb von Bedeutung, weil *Netscape Browser mit v2 CRLs (noch) nicht umgehen können*. Beim Laden einer CRL gibt der Browser die folgende Meldung aus: "*The certificate revocation list you are trying to load has an invalid format.*"

11.1 Aufbau der Indexdatei `index.txt`

In der Datei `index.txt` ist eine Art text-basierte Datenbank, in der die herausgegebenen Zertifikate registriert werden. Die Datei ist von großer Bedeutung für die Verwaltung der Zertifikate. Die `ca`-Applikation z.B. prüft über die letzte Spalte der Datei, ob ein DN schon vergeben wurde.

Beispieleintrag eines Zertifikats in `index.txt`:

```
V    981210145000Z    "leer"    01    unknown    /C=DE/O=Uni/OU=Inf/CN=TestCA/Email= \
                                                                       testca@uni.de
```

Die Indexdatei besteht aus sechs Spalten, jeweils getrennt durch einen Tabulator (nicht durch Leerzeichen). Die Einträge in den einzelnen Spalten haben die folgende Bedeutung:

1. Status des Zertifikats. Einer der Buchstaben R (revoked), E (expired) oder V (valid).

2. Ablaufzeitpunkt des Zertifikats. Format ist YYMMDDHHMMSSZ in UTC-Zeit
3. Ist in obiger Beispielzeile leer. Die Spalte kann aber ebenfalls einen Ablaufzeitpunkt YYMMDDHHMMSSZ in UTC-Zeit enthalten. Ist hier ein Zeitstempel eingetragen, entspricht er dem Widerrufszeitpunkt des Zertifikats. Dann muß in der ersten Spalte (statt V) ein R stehen. Für ein gültiges Zertifikat ist diese Spalte *leer*.
4. Hexadezimale Seriennummer des Zertifikats.
5. Wo das Zertifikat zu finden ist. Wird derzeit immer mit „unknown“ besetzt.
6. Der Name des Zertifikatinhabers. Üblicherweise der Distinguished Name und die E-Mail-Adresse.

Grundsätzlich ist es möglich, über Veränderung der Indexdatei eine irrtümliche Zertifizierung „rückgängig“ zu machen. Das wird erreicht, indem die letzte Zeile gelöscht, und der Zähler in der Datei `serial` um Eins erniedrigt wird. Das gilt allerdings nur für das *zuletzt* herausgegebene Zertifikat, und auch nur, wenn es noch nicht veröffentlicht wurde. Grundsätzlich ist für einen ernsthaften CA-Betrieb von solchen Veränderungen der Indexdatei abzuraten. Der bessere Weg wäre der Widerruf des betreffenden Zertifikats.

11.2 Widerruf eines Zertifikats

In Abhängigkeit vom Status der Zertifikate, die in der Indexdatei `$(SSLETC)/index.txt` durch Gültigkeit, Seriennummer und Distinguished Name (DN) repräsentiert werden, kann mit dem folgenden Kommando eine Certificate Revocation List erzeugt werden. Anschließend muß die CRL noch in das DER-Format gewandelt werden, da die viele Browser nur mit diesem Format umgehen können.

```
openssl ca -gencrl -out $(SSLETC)/crl/crl.pem

openssl crl -in $(SSLETC)/crl/crl.pem -outform der -out
$(SSLETC)/crl/crl.der
```

Vorher müssen die gewünschten Zertifikate zurückgerufen werden. Dazu wird folgendes Kommando verwendet:

```
openssl ca -revoke cert.pem
```

Die von dem Kommando durchgeführten Änderungen der Indexdatei sind die selben, die im folgenden Verfahren beschrieben werden.

Alternativ kann die Indexdatei mit einem Texteditor geändert werden. Soll ein Zertifikat widerrufen werden, kann mit einem Editor die Indexdatei (von Hand oder per Skript) geändert werden. In der ersten Spalte muß das `V` durch ein `R` ersetzt und in der dritten (bisher leeren) Spalte muß der

Widerrufszeitpunkt eingetragen werden. Wegen der UTC-Zeit muß bei der Angabe der Stunden während der Winterzeit eine Stunde, und während der Sommerzeit zwei Stunden, zur lokalen Zeit (Deutschland) bzw. zu dem gewünschten Zeitpunkt, hinzugefügt werden.

Dann kann mit dem obigen Kommando `ca -gencrl` eine CRL mit dem widerrufenen Zertifikat erzeugt werden.

11.3 CRL Authority Key Identifier

Diese Extension besteht aus drei Feldern: `keyIdentifier`, `authorityCertIssuer` und `authorityCertSerialNumber`. Laut RFC 2459 kann der Schlüssel mittels dieser Extension auf zwei Arten identifiziert werden: entweder durch alleiniges Setzen des keyIdentifier-Feldes oder durch Setzen der beiden anderen Felder. Die Extension dient dazu, Zertifikate der Herausgeber-CA zu identifizieren.

RFC 2459 empfiehlt die erste Variante.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
authorityKeyIdentifier = <[keyid[:always]][,issuer[:always]]>
```

11.4 CRL Issuer Alternative Name

Diese Extension ist wie Subject Alternative Name (8.7) aufgebaut und dient dazu, zusätzliche Informationen zum Herausgeber in die CRL zu bringen.

(Für genauere Angaben siehe RFC 2459 [FHPS99] sowie die Beispiel-Konfigurationsdatei im Anhang E.)

In der Konfigurationsdatei:

```
subjectAltName=<[issuer:copy] [, URL:http://my.url.here/] [,
RID:1.2.3.4] [, IP:1.2.3.4]>
```

Kapitel 12

Testbericht: Praxis-Erfahrungen

12.1 OpenSSL

Die folgenden Angaben beziehen sich auf eine Entwickler-Version von OpenSSL. Es ist daher möglich, daß die geschilderten Erfahrungen nicht mehr für die nächste Anwender-Version von OpenSSL gelten.

Die OpenSSL-Applikation `ca` *unterscheidet zwischen Groß-/Klein-Schreibung in Zertifikaten und Anforderungen!* Mit anderen Worten, es ist möglich, ein inhaltlich gleiches Zertifikat zweimal herauszugeben; z.B. einmal mit Locality Name `Hamburg` und das andere Mal mit `HAMBURG`. Auch ist es möglich, daß sich vielleicht nur ein zusätzliches Leerzeichen eingeschlichen hat. Die Zertifikate würden sich jedoch durch die Seriennummer und die unterschiedliche Signatur unterscheiden.

Die Erzeugung eines Root-CA-Zertifikat ist in OpenSSL besser gelöst als in SSLeay, da es jetzt nicht mehr notwendig ist, Extensions nachträglich in das Zertifikat zu patchen. Allerdings wird bei der Erzeugung eines Root-Zertifikats kein Eintrag in die Index-Datei (`index.txt`) vorgenommen. Ebenso ist die Seriennummer des Root-Zertifikats auf „00“ festgelegt. Beides ist nicht immer wünschenswert.

Zertifikate werden von OpenSSL über Hash-Werte, die über den Distinguished Name (und eventuell die E-Mail-Adresse) gebildet werden, identifiziert. Probleme können auftreten, wenn ein Root-Zertifikat erneuert werden muß und der Distinguished Name nicht geändert werden kann oder soll. Die Konsequenzen sind offensichtlich: *Verläßt sich eine Anwendung (womit hier nicht nur OpenSSL gemeint ist) zur Identifizierung des Root-Zertifikats nur auf den Hash-Wert des Issuer-DN (aus dem zu prüfenden Zertifikat) und nicht zusätzlich auf die Extension Authority Key Identifier, kann das Root-Zertifikat nicht mehr (eindeutig) identifiziert werden.* Die Überprüfung von Zertifikatketten würde dann vermutlich fehlschlagen.

Ein Verzeichnis `newcerts` wird bei Aufruf von `make install` nicht angelegt. Die Default-Konfigurationsdatei erwartet dieses Verzeichnis aber. Das Verzeichnis muß daher nachträglich angelegt werden.

Ungünstig ist, daß der Pfad des Zertifikatverzeichnisses (z.B. `/usr/local/etc/ssl`) in die Bibliothek `libcrypto.a` einkompiliert wird. Einige Applikationen, z.B. `pkcs12` zum Lesen von

Zertifikatketten, greifen auf diese Pfade zu. Das erschwert die Verwaltung von mehreren unabhängigen Zertifikatverzeichnissen oder die Änderung dieses Verzeichnisses.

Ansonsten ist die Verwaltung von Zertifikaten und der Betrieb einer CA mit OpenSSL gut möglich. An einigen Stellen ist allerdings Handarbeit notwendig, die aber auch teilweise als Cron-Job mit Hilfe von Skripten erledigt werden kann. Dazu zählen folgende Punkte:

- Neu herausgegebene Zertifikate müssen über ihren Hash-Wert „verlinkt“ werden.
- Es gibt keinen Automatismus, der die Indexdatei auf abgelaufene Zertifikate überprüft und Einträge entsprechend markiert.

12.2 Zertifikate und Browser

Durch die OpenSSL-Applikation `pkcs12` können Anwendungs- und CA-Zertifikate in die Browser importiert werden. Das Programm arbeitet nur mit dem Navigator ab Vers. 4.04 und dem Explorer ab Vers. 4.0 zusammen. Für ältere Browser-Versionen muß das PFX-Programm (siehe Kapitel 13.1) verwendet werden. Außerdem ist es möglich, Benutzerzertifikate mit Schlüssellängen bis zu 2048 Bit als `.p12`-Datei in eine Export-Version des Navigators 4.x zu laden. *Da die USA die Export-Beschränken für Kryptosoftware gelockert haben, sollten in Kürze Browser-Versionen zur Verfügung stehen, bei denen diese* workarounds *nicht mehr erforderlich sind, um mit starken Schlüssellänge arbeiten zu können.*

PKCS-12 ist ein Standard, der ein Format beschreibt, in dem Zertifikate und Schlüssel außerhalb von Anwendungen gespeichert werden können. Zertifikate und Schlüssel, die in diesem Format vorliegen, können dann von verschiedenen Anwendungen gelesen bzw. in diesem Format geschrieben (exportiert) werden.

Neben dem Import von Zertifikaten besteht auch die Möglichkeit, das aus den Browser im PKCS-12-Format exportierte Zertifikate für OpenSSL verfügbar zu machen.

12.2.1 Export aus dem Browser

Um beispielsweise ein mittels der Netscape Export-Funktion aus dem Browser exportiertes Zertifikat *cert*`.p12` für OpenSSL zugänglich zu machen, ist folgender Befehl notwendig:

```
pkcs12 -in cert.p12 -out cert.pem [-des3 | -des | -idea]
```

Es wird nach einem Import-Paßwort gefragt; es ist das Paßwort, das beim Zertifikat-Export mit Netscape angegeben werden mußte und mit dem *cert*`.p12` verschlüsselt wurde. Die Optionen `-des3`, `-des`, bzw. `-idea` geben an, ob und wie das Zertifikat und der Schlüssel, welche sich nach Ausführung des obigen Kommandos in *cert*`.p12` befinden, verschlüsselt werden sollen. Es wird dazu wieder nach einem Paßwort gefragt.

12.2.2 Import in den Browser

Um ein Zertifikat in ein vom Browser importierbares Format zu wandeln:

```
pkcs12 -export -name Listbox-Name -in cert.pem -inkey key.pem
-out file.p12
```

Listbox-Name ist der Bezeichner, unter dem das importierte Zertifikat später in einer der Security-Rubriken des Browsers geführt werden soll. Nach dem die Datei *file*`.p12` erzeugt wurde, kann sie über die Import-Funktion der Browser importiert werden.

Zusammen mit einem Benutzer-Zertifikat können auch CA-Zertifikate in einen Browser importiert werden. Dazu werden Benutzer-Zertifikat und die CA-Zertifikate durch das `pkcs12`-Kommando in eine Datei geschrieben. Das folgende Kommando funktioniert allerdings nur, wenn das Verzeichnis `ssl/certs` unterhalb des Pfades, der bei der Konfiguration des Paketes (siehe 7.2.3) angegeben wurde, vorliegt. Der Pfad ist in die Bibliothek `libcrypto.a` einkompiliert und kann nicht geändert werden. Es müssen alle CA-Zertifikate der Kette (auch das Root-Zertifikat) im Verzeichnis `ssl/certs` vorliegen und über ihre Hash-Werte (siehe oben (10.3)) verlinkt sein.

Um eine Zertifikat-Kette in ein vom Browser importierbares Format zu wandeln:

```
pkcs12 -chain -export -name Listbox-Name -in cert.pem -inkey
key.pem -out file.p12
```

Sollen die CAs der Zertifikatkette ebenfalls eine Bezeichnung bekommen, kann dazu die Option `-caname "`*CA Name*`"`, auch mehrfach, verwendet werden. Damit die Zuordnung der Bezeichner zu den CAs stimmt, müssen als erstes der Bezeichner für die Root-CA und die nächsten Bezeichner entsprechend der Kette aufgeführt werden. Dieser sogenannte „Friendly Name“ für CA-Zertifikate wird von Microsoft-Produkten unterstützt.

Statt mit der Option `-chain` kann ein CA-Zertifikat über die Option `-certfile` *cacerts.pem* angegeben werden. Sollen es mehrere sein, werden sie mit in die Datei geschrieben, z.B. durch folgendes Kommando:

```
cat ca1.pem ca2.pem ca3.pem > cacerts.pem
```

Nachdem die `.p12`-Datei erzeugt wurde, kann das Benutzer-Zertifikat in den Browser importiert werden (und damit zugleich auch die CA-Zertifikate). Das Benutzer-Zertifikat kommt in die Rubrik „Security/Yours“ und die CA-Zertifikate nach „Security/Signers“. Für die CA-Zertifikate muß dann mittels des „Edit“-Buttons im NSC noch mitgeteilt werden, wofür sie jeweils gültig sein sollen. Es gibt drei Alternativen zur Auswahl: Zertifizierung von SSL-Servern, von E-Mail-Benutzern oder von Software-Entwicklern. Diese Auswahl gilt auch für Wurzel-Zertifikate, die nur untergeordnete CAs signieren, wobei dann die Auswahl inhaltlich wenig Sinn macht. Wenn das Wurzel-Zertifikat für einen Verwendungszweck als gültig akzeptiert wurde, muß eine Auswahl für untergeordnete CAs nicht mehr getroffen werden, auch wenn es hier u.U. Sinn machen würde. Erst nach Auswahl des Verwendungszwecks verläuft eine Überprüfung des Benutzer-Zertifikats erfolgreich.

Auf diese Weise können aber nur CA-Zertifikate mit gesetzten „CA-Bit“ importiert werden, also solche, bei denen `nsCertType` auf `sslCA`, `emailCA`, `objCA` oder `basicConstraints` bzw. eine von deren Kombinationen im CA-Zertifikat gesetzt sind.

12.3 Zertifikate und CRLs mit dem Netscape Browser

Zertifikate und CRLs können von einem HTTP-Server als spezielle MIME-Types an einen Browser gesendet werden. Dazu müssen die Zertifikate und die CRLs im (binären) DER-Format vorliegen. Üblicherweise erzeugen die OpenSSL-Applikationen Dateien im PEM-Format (Base64). Es besteht aber die Möglichkeit, die Dateien nachträglich in das DER-Format umzuwandeln. Zertifikate werden durch folgenden Befehl in das DER-Format umgewandelt:

```
openssl x509 -in cert.pem -outform der -out cert.der
```

Der entsprechende Befehl, um eine CRL in das DER-Format zu wandeln:

```
openssl crl -in crl.pem -outform der -out crl.der
```

Achtung:
Netscape-Browser akzeptieren (bisher) keine X.509v2-CRLs. Genaueres steht im Abschnitt über CRLs (11).

12.3.1 Zertifikate

Mit OpenSSL erzeugte Zertifikate lassen sich in den Netscape Communicator/Navigator relativ einfach importieren. Das Zertifikat im (binären) DER-Format kann von einem HTTP-Server als einer der folgenden MIME-Types an den Browser geschickt werden:

- `application/x-x509-email-cert`
- `application/x-x509-user-cert`
- `application/x-x509-ca-cert`

Dazu werden die Zertifikate auf einem Server zum Herunterladen zur Verfügung gestellt. Die Datei `mime.types` des jeweiligen Web-Servers (z.B. Apache) muß um die obigen Typen ergänzt werden. Die Dateiendung, die den MIME-Types zugeordnet wird, muß mit der Dateiendung der Zertifikate übereinstimmen. Der Server kennzeichnet dann beim Senden die Datei mit dem entsprechenden MIME-Type. Der MIME-Type signalisiert dem Browser, daß es sich um ein Zertifikat handelt, und der Browser startet dann einen entsprechenden Interaktions-Modus. Der Browser bietet dem Benutzer in Abhängigkeit vom MIME-Type und der Erweiterung *Netscape-Cert-Type* eine Auswahl an, für welchen Zweck ein Zertifikat bestimmt ist. Im folgenden eine Aufstellung der Browser-Vorschläge[1] in Abhängigkeit von Cert- und MIME-Type.

<table>
<tr><th><code>nsCertType</code></th><th>Beschreibung</th><th>Rubrik</th><th>Accept this CA for Certifying ...</th></tr>
<tr><td><code>objCA</code></td><td>CA-Zert. für 0x10-Zert.</td><td>Signers</td><td>... Software-Developers</td></tr>
<tr><td><code>emailCA</code></td><td>S/MIME-CA</td><td>Signers</td><td>... e-mail users</td></tr>
<tr><td><code>sslCA</code></td><td>SSL CA</td><td>Signers</td><td>... network sites</td></tr>
<tr><td><code>reserved</code></td><td>Reserviert</td><td>Signers</td><td rowspan="5">Für <u>alle fünf</u> CertTypes
folgende Auswahl:
... Software-Developers
... network sites
... e-mail users</td></tr>
<tr><td><code>objsign</code></td><td>Zur Objekt-Signierung</td><td>Signers</td></tr>
<tr><td><code>email</code></td><td>S/MIME Benutzer-Zert.</td><td>Signers</td></tr>
<tr><td><code>server</code></td><td>SSL-Server</td><td>Signers</td></tr>
<tr><td><code>client</code></td><td>SSL-Client</td><td>Signers</td></tr>
</table>

Zertifikate als MIME-Type `x-x509-ca-cert` gesendet: siehe obige Tabelle

Zertifikate als MIME-Type `x-x509-user-cert` gesendet:

Alle Zertifikate werden *unabhängig vom Netscape-Cert-Type* als „eigene E-Mail Benutzer-Zertifikate“ erkannt.

Zertifikate als MIME-Type `x-x509-email-cert` gesendet:

Alle Zertifikate werden *unabhängig vom Netscape-Cert-Type* als „fremdes E-Mail Benutzer-Zertifikat“ erkannt. Nachdem die Zertifikate akzeptiert wurden, werden sie allerdings in unterschiedliche NSC-Zertifikat-Rubriken, „People“ und „Certificate/Signers“ einsortiert. Die drei Zertifikat-Typen (`nsCertType objCA, objsign, server`), die in der Aufstellung fehlen, sind nach Akzeptieren des Zertifikats in keiner Rubrik des Browsers mehr aufzufinden.

<table>
<tr><th><code>nsCertType</code></th><th>Beschreibung</th><th>Rubrik</th><th>Accept this CA for Certifying ...</th></tr>
<tr><td><code>email</code></td><td>S/MIME-CA</td><td>Signers</td><td>... e-mail users</td></tr>
<tr><td><code>sslCA</code>
<code>sslCA</code></td><td>SSL-CA
SSL-CA</td><td>Signers</td><td>... network sites
... e-mail users</td></tr>
<tr><td><code>reserved</code></td><td>Reserviert</td><td>People</td><td></td></tr>
<tr><td><code>email</code></td><td>S/MIME Benutzer-Zert.</td><td>People</td><td></td></tr>
<tr><td><code>client</code></td><td>SSL-Client</td><td>People</td><td></td></tr>
</table>

Der NSC akzeptiert Schlüssellängen von 2048 Bit für CA-Zertifikate. Für Benutzerzertifikate beträgt die max. Schlüssellänge 512 Bit bei der Export-Version des Browser. Im allgemeinen werden CA-Zertifikate von einem Server (die entsprechende Konfiguration des Servers vorausgesetzt) im binären DER-Format zur Verfügung gestellt. Der Benutzer kann dann das CA-Zertifikat als MIME-Type `x-x509-ca-cert` (in der Regel über eine HTML-Seite) in den Browser laden. Er wird dabei durch einen Interaktionmodus geführt, in dessen Verlauf er bestimmen kann, welches Vertrauen er dem Zertifikat entgegen bringt. *Laut* STEPHEN N. HENSON[2] *kann jedes Zertifikat, unabhängig von der Art der enthaltenen Extensions, nach Durchlaufen des Interaktions-Modus als CA für jeden Zweck akzeptiert werden.* Das kann dann zu Problemen führen, wenn die CA für einen Zweck akzeptiert wurde, den die Extensions im CA-Zertifikat nicht unterstützen. Ist in dem CA-Zertifikat z.B. kein Bit gesetzt, welches die CA als Herausgeber von S/MIME-Zertifikaten kennzeichnet, und wird ein von dieser CA herausgegebenes Zertifikat zum Signieren von E-Mail verwendet, wird die Mail beim Empfänger wegen des ungültigen CA-Zertifikats abgewiesen.

Die Möglichkeit, CA-Zertifikate über Benutzerzertifikate ohne Web-Server in den NSC zu importieren, wird im Abschnitt über PFX (13.1) bzw. PKCS12 (12.2) beschrieben.

[1] `http://home.netscape.com/eng/security/comm4-cert-download.html`
[2] `http://www.drh-consultancy.demon.co.uk/caornot.html`

Eine Schlüsselerzeugung mit anschließender Online-Zertifizierung wird unterstützt. Da der Schlüssel vom Browser erzeugt wird, ist hier bei der Export-Version die Schlüssellänge auf 512 Bit beschränkt. Bei Verwendung der oben genannten Programme ist es möglich, die max. Schlüssellänge für Benutzerzertifikate auf 2048 Bit zu erhöhen (getestet mit Version 4.0 und 4.05). SSL-Server-Zertifikate werden beim Anwählen eines SSL-Servers automatisch geladen. Liegt das CA-Zertifikat für einen solchen Server nicht vor, wird ebenfalls ein Interaktionsmodus gestartet, in dessen Verlauf der Benutzer selber entscheiden muß, welches Vertrauen er dem Server-Zertifikat entgegenbringt. Liegt dagegen das CA-Zertifikat vor, bekommt der Benutzer in Abhängigkeit von der vorgenommenen Einstellung am Browser keine oder nur eine kleine Meldung. (Es sei an dieser Stelle darauf hingewiesen, daß der Server beim Verbindungsaufbau die Möglichkeit hat, zusätzlich zu seinem eigenen Zertifikat auch die Zertifikate der übergeordneten CA(s) mitzuliefern.)

12.3.2 CRLs

Achtung:
Netscape-Browser akzeptieren (bisher) keine X.509v2 CRLs. Genaueres steht in Kapitel 11.

Leider scheint der Netscape-Browser die Extension CRL Distribution Points nicht zu unterstützen. Eine CRL muß daher explizit vom Anwender geladen werden. Eine CRL kann nur in den Browser geladen werden, wenn sie durch den Anwender von einem Server als spezieller MIME-Type heruntergeladen wird. Diese Mechanismus wird im folgenden beschrieben. Die aufgeführten MIME-Types müssen wieder, wie im vorigen Abschnitt (12.3.1) beschrieben, eigentragen sein. Ebenso müssen die Dateiendungen übereinstimmen.

Die im folgenden verwendeten CRLs waren vom Typ X.509v1 und lagen im DER-Format vor.

CRL als MIME-Type `application/x-pkcs7-crl` gesendet:

Der Browser (Vers. 4.03) akzeptiert die CRL, ohne eine Meldung auszugeben. Wird versucht, die CRL ein zweites Mal zu laden, wird die in Abb. 12.1 dargestellte Error-Box angezeigt.

Abb. 12.1: Netscape-Fehlermeldung bei wiederholtem Laden einer CRL

Nachdem in einem Browser Vers. 4.05 eine CRL geladen wurde, taucht im Security Fenster nach Anwahl der Rubrik „Signers“ ein neuer Button „Edit/View CRL's “ auf. Darüber lassen sich CRLs anzeigen, löschen und neu laden.

CRL als MIME-Type `application/x-x509-crl` gesendet:

Der Browser (Vers. 4.03) lädt das CRL-Binary nicht als CRL, sondern als Text mit entsprechend kryptischen Zeichen im Browser-Fenster.

Wurde eine CRL, die ein widerrufenes SSL-Server Zertifikat enthielt, in einen Browser der Vers. 4.05 gebracht, wird anschließend eine Verbindung zu dem entsprechenden SSL-Server verweigert. Allerdings mit einer wenig hilfreichen Meldung der Art „Es gibt Schwierigkeiten...“

Der Browser gibt folgende Meldung aus, wenn der Gültigkeitszeitraum einer schon geladenen CRL überschritten ist:

Abb. 12.2: Netscape-Fehlermeldung: überschrittene Gültigkeitsdauer einer CRL

Es muß dann zunächst die aktuelle CRL geladen werden, bevor erneut eine Verbindung aufgebaut werden kann.

12.4 Zertifikate und CRLs mit Microsoft Browser

Der Internet Explorer (MSIE) unterstützt u.a. die Standard X509v3-Attribute wie Basic Constraints und Key Usage (siehe jedoch Zertifizieren (10.3)). Um ein CA-Zertifikat für den MSIE als solches zu kennzeichnen, muß das „CA-Bit“ in Form von Basic Constraints gesetzt werden. Außerdem kann über Key Usage die Verwendung des Zertifikats eingeschränkt werden. Der Browser interpretiert laut MS-Dokumentation Key Usage immer, egal ob Critical oder nicht.

Die von HENSON geschilderten Probleme mit Critical Extensions scheinen zumindest für MS Outlook Express 5 nicht mehr zu gelten. Eine Zertifikat-Kette mit Critical gesetzten Basic Constraints ließ sich problemlos importieren.

Die folgende Angaben beziehen sich hauptsächlich auf den MSIE 4.0 und zum Teil auf Version 5.x. Es ist dabei zu berücksichtigen, daß sich das Verhalten der Browsers unter Windows NT in Abhängigkeit vom installierten Servicepack ändern kann. Gleiches gilt vermutlich auch für Windows 9x.

12.4.1 Zertifikate

Root-CA-Zertifikate können über einen Web-Server als MIME-Type `application/x-x509-ca-cert` gesendet werden (s.o. (12.3.1)). Sie werden als CA-Zertifikate für „Netzwerk-Client-“, „Netzwerk-Server-Authentifikation“, „Sichere-E-Mail“ und „Software-Herausgeber“ akzeptiert. Es besteht die Möglichkeit, über Kontrollkästchen die Beschränkungen des Zertifikats einzeln zu (de-) aktivieren. Nach dem Akzeptieren findet sich das Zertifikat in der Rubrik „Ansicht / Internetoptionen / Inhalt / Agenturen".

Erhebliche Probleme gibt es allerdings mit CA-Zertifikaten, die nicht selbstsigniert, also keine Root-Zertifikate sind. Bis Version 4.x läßt sich ein solches Zertifikat nur durch eine der folgenden Improvisationen installieren:

1. Das Zertifikat wird als MIME-Type `application/pkcs7-mime` auf einem Server zur Verfügung gestellt. Zur Installation des MIME-Types siehe obigen Abschnitt (12.3.1). Beim Laden des Zertifikats wird das MS Address Book `wab.exe` aufgerufen, wenn es installiert ist. Danach besteht die Möglichkeit, das Zertifikat über eine Dialogbox als gültig zu akzeptieren. Das Zertifikat erscheint dann zwar nicht in der Liste der vertrauenswürdigen CAs, aber von dieser CA herausgegebene Server-Zertifikate werden als gültig anerkannt.

2. Soll das CA-Zertifikat in der Liste der vertrauenswürdigen CAs erscheinen, wird das MS-Programm `certmgr.exe` benötigt. Das CA-Zertifikat muß von einem Server als MIME-Type `application/x-x509-ca-cert` heruntergeladen und lokal abgespeichert werden, z.B. unter dem Namen `cacert.crt`. Jetzt muß eine DOS-Shell gestartet werden. In der DOS-Shell wird in das Verzeichnis gewechselt, in dem `certmgr.exe` und `cacert.crt` vorliegen. Jetzt wird das Zertifikat durch folgendes Kommando geladen (*Reihenfolge der Parameter unbedingt beachten!*):

   ```
   certmgr -c cacert.crt -s Root -add
   ```

   Hat das Importieren des Zertifikats fehlerfrei funktioniert, wird die Meldung „CertMgr Succeeded" ausgegeben, und es erscheint Dialogbox, die durch das Klicken auf „Yes" zu beenden ist (Abb. 12.3).

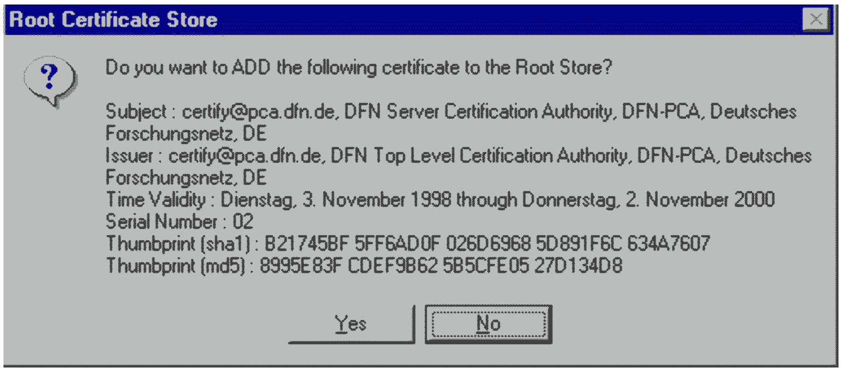


Abb. 12.3: Internet Explorer: Import eines Zertifikates

Mit dem IE Version 5.x ist dieses Verfahren nicht mehr notwendig. Es gibt jetzt eine Rubrik für nicht selbstsignierte CAs. Die Version 5.x zeichnet sich gegenüber der 4er-Version durch eine grundsätzlich verbesserte Verwaltung von Zertifikaten aus. Allerdings werden vom Benutzer akzeptierte *Server*-Zertifikate in die Rubrik „Eigene Zertifikate" einsortiert, es gibt keine Rubrik für Server-Zertifikate.

Benutzer-Zertifikate können auf zwei Arten in den MSIE gebracht werden. Die eine Art umfaßt die Schlüssel- und Request-Erzeugung mit dem Browser durch Ausführen eines Java- oder VB-Skript. Anschließend wird der Request online zertifiziert. Auf diese Methode wird hier nicht weiter eingegangen.

Die zweite Möglichkeit besteht darin, wie im Abschnitt Erzeugen von Requests (10.2) beschrieben einen mit OpenSSL erzeugten Request zu signieren und diesen anschließend als `.p12`-Datei in den Browser zu importieren. Dazu wird in der Rubrik „Ansicht / Internetoptionen / Inhalt / Eigene" eine Dialogbox geöffnet, wo über den Button „Importieren" ein Zertifikat importiert werden kann. Die in den Browser gebrachten Zertifikate stehen im MS-Mail Programm Outlook-Express (OE) zur Verfügung, wenn die im Zertifikat angegebene E-Mail-Adresse mit der im OE übereinstimmt.

Beim Versuch eine `.p12`-Datei in die 5er-Version des Browsers bzw. von Outlook Express zu laden, fällt auf, daß diese die Endung `.p12` nicht kennen. Erkannt wird die Endung `.pfx`. Die `.p12`-Datei läßt sich aber trotzdem laden, wenn die Ansicht auf „alle Dateien" erweitert und die Datei dann ausgewählt wird. Eine `.p12`-Datei läßt sich, zumindest unter Windows 98, auch durch einen „Doppelklick" auf die Datei laden. Es wird dann der „Internet-Assistent für die Zertifikatverwaltung" gestartet.

Unglücklicherweise werden persönliche Zertifikate und Private-Keys durch den 4er IE *nicht geschützt*. Der einzige Schutz besteht in der Zugangsbeschränkung durch das Betriebssystem. STEPHEN HENSON beschreibt in einem Dokument[3], wie der Schutz des Private-Keys durch den MS `certmgr` nach dem Installieren eines Zertifikats erhöht werden kann.

12.4.2 CRLs

CRLs bezieht der MSIE über eine im Zertifikat angegebene URI. Die URI wird über die Extension CRL Distribution Points in ein Zertifikat gebracht. Eine Zertifikatprüfung erfolgt erst, wenn im 4er-Browser in der Rubrik „Ansicht / Internetoptionen / Erweitert" im Abschnitt „Sicherheit" das Kontrollkästchen „Auf zurückgezogene Zertifikate überprüfen" angewählt ist.

In der 5er-Version heißt die Rubrik „Extras / Internetoptionen / Erweitert". Dort sind die Kontrollkästchen „Auf zurückgerufenen Serverzertifikate" bzw. „Zertifikate von Herausgebern prüfen" anzuwählen. Damit OE 5 auf zurückgerufene Benutzer-Zertifikate prüft, muß in der Rubrik „Extras / Optionen / Sicherheit / Weitere Einstellungen" das Kästchen „Auf zurückgezogene Digitale ID's prüfen" angewählt werden. Die CRLs werden dann automatisch über die entsprechende URL bezogen.

Es ist auch möglich, CRLs über die Import-Funktion des Zertifikat-Assistenten (OE 5) zu Laden. Der Assistent erkennt am Datei-Inhalt, daß es sich um eine CRL handelt. Leider scheint es keine Möglichkeit zu geben, die geladenen CRLs anzeigen zu lassen. Auch kann der Inhalt der CRLs nicht angezeigt werden.

Die Handhabung von CRLs scheint auch noch nicht ganz ausgereift zu sein. Ein Benutzer-Zertifikat wurde vom Zertifikat-Assistenten als nicht überprüfbar markiert, da keine CRL der CA vorlag. Die CA war eine Zwischen-CA, *für die aber eine CRL geladen war*. Für die Root-CA, die die Zwischen-CA zertifiziert hat, lag dagegen keine CRL vor ...

[3] `http://www.drh-consultancy.demon.co.uk/cexport.html`

Kapitel 13

Ergänzende Programme

13.1 `pfx-0.1.2`

Das Programm **`pfx-0.1.2`** von STEPHEN N. HENSON ermöglicht es, vom Netscape Communicator/Navigator und MS-Internet-Explorer (jeweils ab Vers. 4.0x) exportierte Zertifikate für OpenSSL lesbar zu machen, bzw. mit OpenSSL erzeugte Zertifikate in den NSC und den MSIE zu importieren. Außerdem ist es möglich, Benutzerzertifikate mit Schlüssellängen bis zu 2048 Bit als `.p12`-Datei in den NSC zu laden. Für NSC ab Vers. 4.04 und MSIE ab Vers. 4.0x sollte statt `pfx` das `pkcs12`-Programm (siehe PKCS12 (12.2)) verwendet werden.

13.1.1 Übersetzung und Installation

Folgende Änderungen sind im `Makefile` vorzunehmen:

> `($(SSLDIR)=`*`Installationsverzeichnis`*`, $(SSLSRC)=`*`Quellverzeichnis`*`)`

- Zeile 1, `SSLINC=/usr/local/ssl/include`:
  Wenn die Include-Dateien von OpenSSL mit installiert wurden, Pfad auf `$(SSLDIR)/include`, sonst Pfad auf `$(SSLSRC)/include` setzen.
- Zeile 2, `SSLLIB=/usr/local/ssl/lib`:
  Pfad auf `SSLLIB=$(SSLDIR)/lib` setzen.
- Zeile 3, `CFLAGS=-Wall -O2 -I$(SSLINC) -g`:
  Flags ändern in

  ```
  CFLAGS=-fomit-frame-pointer -mv8 -Wall -O2 -I$(SSLINC)
  ```
- In Zeile 11, `cc -g -L$(SSLLIB) -o pfx $(OBJS) -lcrypto`:
  `cc` durch `gcc` ersetzen und die Option `-g` entfernen.

Schließlich ist das Programm mit

```
make 'CC=gcc'
```

zu übersetzen.

Als Installationsverzeichnis bietet sich `$(SSLDIR)/bin` an:

- `cp $(PFX_SRC)/pfx $(SSLDIR)/bin`
- `chmod 755 $(SSLDIR)/bin/pfx`

13.1.2 Anwendung von `pfx`

Export aus einen Browser

Um ein mittels der Netscape Export-Funktion aus dem Browser exportiertes Benutzer-Zertifikat *cert*`.p12` für OpenSSL zugänglich zu machen, ist folgender Befehl notwendig:

```
pfx -in cert.p12 -out cert.pem [-des3 | -des | -idea]
```

Es wird nach einem Import-Paßwort gefragt; es ist das Paßwort, das beim Zertifikat-Export mit Netscape angegeben werden mußte und mit dem *cert*`.p12` verschlüsselt wurde. Die Optionen `-des3`, `-des`, bzw. `-idea` geben an, ob und wie das exportierte Zertifikat verschlüsselt werden soll.

Import in einen Browser

Um ein mit OpenSSL erstelltes Zertifikat in den NSC zu importieren:

```
pfx -export -name Listbox-Name -out cert.p12 -in cert.pem
```

Listbox-Name ist die Bezeichnung, unter der das Zertifikat nach dem Import im Browser in einer Security-Rubrik geführt wird.

Nach dem Schlüssel-Paßwort wird ein weiteres Paßwort erfragt; es ist das Paßwort um die *cert*`.p12`-Datei zu verschlüsseln. Das so vorbereitete Zertifikat kann über die NSC-Import-Funktion importiert werden. Der Browser erwartet als zweites Paßwort (das erste diente zum Freigeben der NSC Zertifikat-Datenbank) das Paßwort, mit dem die `.p12`-Datei verschlüsselt wurde. Die Zertifikate werden nur in den NSC-Bereich „Security/Yours", also als „eigene E-Mail-Zertifikate", gebracht.

Mit folgendem Befehl kann indirekt auch ein CA-Zertifikat in den Netscape-Browser geladen werden:

```
pfx -export -name Listbox-Name -in usercert.pem -inkey user-
key.pem -out mycert.p12 -certfile CAcert.pem
```

Das CA-Zertifikat wird an ein vorher erzeugtes Benutzer-Zertifikat *usercert*`.pem` gebunden. Anschließend kann das Benutzer-Zertifikat importiert werden und damit gleichzeitig das „angehängte“ CA-Zertifikat. Das Benutzer-Zertifikat kommt in die Rubrik „Security/Yours“ und das CA-Zertifikat nach „Security/Signers“. Für das CA-Zertifikat muß dann mittels des „Edit“-Buttons im NSC noch mitgeteilt werden, wofür das CA-Zertifikat gültig sein soll. Erst dann verläuft eine Überprüfung des Benutzer-Zertifikats erfolgreich. Auf diese Weise können aber nur Zertifikate mit gesetztem „CA-Bit“ importiert werden, also wenn `nsCertType` auf `sslCA`, `emailCA`, `objCA` bzw. deren Kombinationen gesetzt ist.

Weiter ist es möglich, Zertifikat-Ketten in den Browser zu importieren. Voraussetzung ist, daß alle $n-1$ Zertifikate das „CA-Bit“ gesetzt haben. Mit folgendem Befehl werden an ein Zertifikat alle in der Zertifikat-Hierarchie darüber liegenden Zertifikate einschließlich des Root-CA-Zertifikats an das Zertifikat gebunden:

```
pfx -export -name Listbox-Name -in usercert.pem -inkey user-
key.pem -out usercert.p12 -chain
```

Das Kommando funktioniert nur erfolgreich, wenn die CA-Zertifikate nach `$(SSLDIR)/certs` kopiert und deren Hash-Werte gebildet worden sind (s. die *Anmerkung* am Ende des Abschnitts über die Hash-Werte (10.1)). Die mit einem Zertifikat verbundenen CA-Zertifikate werden alle in den Browser gebracht und sind über die Browser-Rubrik „Signers“ zugänglich. Eine erfolgreiche Überprüfung der Zertifikate setzt voraus, daß zumindest für das Root-CA-Zertifikat der „CA-Typ“ (Server-CA, Client-CA usw.) mit Hilfe des Browsers eingestellt wird (s.o., Edit-Button (13.1.2)).

13.2 pkcs12-054

Das Programm `pkcs12-054` bzw. dessen bereitgestellte Funktionalität ist in OpenSSL-0.9.5-dev vollständig integriert (siehe Abschnitt 12.2). Somit wird `pkcs12-054` nicht mehr benötigt.

13.3 ca-fix-0.3

Die von dem Programm `CA-Fix` bereitgestellte Funktionalität ist in OpenSSL-0.9.5-dev vollständig integriert. Somit wird `CA-Fix` nicht mehr benötigt.

Teil III

Integration von SSL in den Apache-Webserver

Kapitel 14

Installation der Software

14.1 Einleitung

Dieser Teil des Handbuches richtet sich an alle, die sich für einen SSL-fähigen Apache-Web-Server interessieren und genaueres über die Konfiguration des SSL-Teils eines solchen Servers wissen wollen. Diese Dokumentation kann und soll nicht die grundsätzliche Konfiguration eines „normalen“ Apache beschreiben. Daher werden im folgenden Kenntnisse über Konfiguration und Betrieb eines solchen Servers vorausgesetzt.

Durch ein SSL-Paket (*Mod-SSL*) werden neue SSL-bezogene Server-Direktiven in den Apache gebracht. Diese SSL-Direktiven werden ausführlich in der Dokumentation des SSL-Pakets erläutert. Einige dieser Direktiven werden zusätzlich in den einzelnen Abschnitten von Kapitel 15 dargestellt.

Zunächst einige Anmerkungen über den Zusammenhang zwischen *Apache-SSL* und *SSL-Apache*. Der Apache enthält keinerlei Kryptoroutinen. Wie im nächsten Abschnitt erläutert wird, ist es deshalb notwendig, Schnittstellen zu einer Krypto-Software (*OpenSSL*) in die Apache-Quellen zu „patchen“. Dazu stehen zwei Software-Pakete zur Verfügung: ein Paket, das von BEN LAURIE entwickelt wird, und ein anderes, das unter Koordination von RALF S. ENGELSCHALL entwickelt wird. Der SSL-Server, der bei der Verwendung von LAURIES Paket entsteht, wird üblicherweise **Apache-SSL** genannt. Der unter Verwendung von ENGELSCHALLS Paket erstellte Server heißt dagegen **Mod-SSL-Apache**, im Folgenden kurz *SSL-Apache*.

Beide bieten eine recht ähnliche Funktionalität. Das Vergleichen der Pakete in den entsprechenden Mailinglisten hat in der Vergangenheit zu recht „engagierten“ Auseinandersetzungen geführt. Daher soll hier nur kurz erläutert werden, warum für diese Dokumentation die Wahl auf das Paket von ENGELSCHALL fiel. Zum Zeitpunkt der Entscheidung unterstützte das Paket die Verwendung von Widerruflisten und bestach durch seine umfangreiche Dokumentation. Dies kann inzwischen durchaus auch der Fall für das Paket von LAURIE sein. Die Empfehlung lautet daher, sich beide Pakete anzuschauen und dasjenige zu wählen, welches den eigenen Anforderungen am ehesten Genüge tut.

14.2 Benötigte Software-Pakete

Bevor der SSL-Apache-Server in Betrieb genommen werden kann, müssen zunächst einige Software-Pakete übersetzt und installiert werden. Der SSL-Server liegt nicht als ein komplettes Quell-Paket vor. Das liegt daran, daß der Apache-HTTP-Server (ohne SSL-Funktionalität) von einer internationalen Gruppe entwickelt wird, darunter auch Mitglieder aus den USA. In den USA bestanden und bestehen z.T. immer noch gewisse Export-Beschränkungen hinsichtlich Krypto-Software und -Schnittstellen. Daher erfolgt die Entwicklung des Servers ohne SSL-Funktionalität bzw. Schnittstellen. Die SSL-Funktionalität wird durch ein außerhalb der USA entwickeltes Software-Paket, *OpenSSL*, bereitgestellt. Die benötigten Schnittstellen zwischen Apache und dem OpenSSL-Paket werden schließlich durch das (ebenfalls außerhalb der USA entwickelte) *Mod-SSL*-Paket in die Apache-Server-Quellen „gepatched".

Die Übersetzung und Installation der einzelnen Pakete wird in den folgenden Abschnitten beschrieben.

Es sind im Prinzip vier Software-Pakete notwendig:

1. Die Sourcen des HTTP-Servers Apache, `apache_1.3.9.tar.gz`
2. Die Sourcen des TLS-/SSL-Pakets OpenSSL, `openssl-0.9.4.tar.gz`
3. Die Sourcen, die den Apache um die Schnittstellen zum TLS-/SSL-Paket erweitern, `mod_ssl-2.4.10-1.3.9.tar.gz`
4. die Quellen zur Verwaltung von Shared-Memory, `mm-1.0.12.tar.gz` (optional, weil „lediglich" die Performanz des SSL-Apache-Servers verbessert wird)

Informationen zu Installation und Betrieb des OpenSSL-Pakets sind wegen ihres Umfangs in einem separaten Teil dieses Handbuches (Teil II) beschrieben. Eine Kurzfassung zur Übersetzung und Installation der aktuellsten verfügbaren Anwenderversion von OpenSSL, OpenSSL-0.9.4, findet sich in Abschnitt 14.4.

Alle in diesem Dokument gemachten Angaben zur Übersetzung und Konfiguration der einzelnen Software-Pakete beziehen sich auf das Betriebssystem SunOS 5.5.1 (Solaris 2.5.1) und den C-Compiler gcc-2.7.2.1.

14.3 Das Paket `mm-1.0.12.tar.gz`

Wie im vorigen Abschnitt angedeutet, ist dieses Paket für den Betrieb des Servers nicht unbedingt erforderlich, aber wegen der verbesserten Performanz sinnvoll.

Das MM-Paket (MM steht für „memory mapped") wurde von RALF S. ENGELSCHALL entwickelt. Es stellt eine einheitliche Schnittstelle zur Verwaltung von Shared-Memory auf verschiedenen Plattformen zur Verfügung. Im Zusammenhang mit SSL-Apache kann bei Verwendung der MM-Bibliothek ein RAM-basierter SSL-Session-Cache anstelle eines Festplatten-Cache eingesetzt werden.

Auspacken der Quellen in einem Verzeichnis (z.B. `/usr/src`):

```
gzip -dc mm-1.0.12.tar.gz | tar -xvf -
```

Mit dem folgenden Befehl (nach dem Wechsel in das MM-Verzeichnis) wird das Paket konfiguriert:

- `cd mm-1.0.12`
- `sh ./configure --prefix=/usr/local`

Die Option `--prefix=`*xxx* ermöglicht die Angabe eines Pfades, unter dem nach der Übersetzung ein Konfigurationsskript (*xxx*`/bin`), die Bibliothek (*xxx*`/lib`), Header-Dateien (*xxx*`/include`) und Manual-Seiten (*xxx*`/man/man1` und *xxx*`/man/man3`) installiert werden. Soll später nicht das komplette Paket installiert werden sondern nur die Bibliothek, oder wenn die Installation durch Kopieren erfolgen soll, empfiehlt es sich, `--prefix=/usr/local` zu ersetzen durch `--prefix=‘pwd‘`. Die spätere, in jedem Fall notwendige Installation durch `make install` erfolgt dann im aktuellen Verzeichnis, also dem Quell-Verzeichnis.

Durch den Aufruf von `make install` werden nicht nur Dateien installiert, sondern es wird auch die benötigte Bibliothek erzeugt. Daher ist der Aufruf nicht so ohne weiteres zu ersetzen. Die Angabe von `--prefix=‘pwd‘` bietet die Möglichkeit einer lokalen Installation. Die benötigte Bibliothek kann dann in das gewünschte Verzeichnis kopiert werden. Entsprechendes gilt für die anderen Dateien.

Der Aufruf des `configure`-Befehls konfiguriert das Paket zur Erzeugung einer statischen und einer dynamischen Bibliothek. Soll die MM-Bibliothek statisch in den Apache gelinkt werden, muß die ausschließliche Erzeugung statischer Bibliotheken durch Konfiguration mit folgendem Kommando (anstelle des obenstehenden) veranlaßt werden:

```
sh ./configure --disable-shared --prefix=/usr/local
```

Die Übersetzung des Pakets erfolgt durch Aufruf von

```
make
```

Testen des Pakets durch Aufruf von

```
make test
```

Bei erfolgreichem Test wird nach einer Reihe von Test-Daten die folgende Meldung ausgegeben:

```
OK - ALL TESTS SUCCESSFULLY PASSED.
```

Abschließend kann das Paket mit folgendem Befehl installiert werden:

```
make install
```

Wurde bei obigem `Configure`-Kommando die Option `--prefix` nicht auf `'pwd'` gesetzt, ist die Installation abgeschlossen. Der restliche Teil dieses Abschnitts bezieht sich auf die Installation durch Kopieren der einzelnen Dateien.

Soll die statische MM-Bibliothek verwendet werden, ist eine Installation der Bibliothek im Prinzip nicht notwendig. Es reicht, bei der Konfiguration des Apache (s. 14.6) die MM-Konfigurations-Variable `EAPI_MM` auf den Pfad zum `lib`-Verzeichnis mit der statischen Bibliothek zu setzen, z.B. `EAPI_MM=/usr/src/mm-1.0.12`. Unterhalb dieses Verzeichnisses findet sich auch das `include`-Verzeichnis mit der MM-Header-Datei. Soll die statische Bibliothek trotzdem installiert werden, sind die folgenden Kommandos notwendig:

- `cp lib/libmm.a lib/libmm.la /usr/local/lib`
- `chmod 644 /usr/local/lib/libmm.a /usr/local/lib/libmm.la`
- `cp include/mm.h /usr/local/include`
- `chmod 644 /usr/local/include/mm.h`

Soll der SSL-Apache die dynamische Bibliothek verwenden, muß die Bibliothek entweder in einem der üblichen Bibliotheks-Verzeichnisse installiert werden oder es muß vor dem Start des SSL-Apache die Umgebungsvariable `LD_LIBRARY_PATH` um das Verzeichnis, welches die MM-Bibliothek enthält, erweitert werden. Soll die Installation der dynamischen Bibliothek z.B. in `/usr/local/lib` erfolgen, sind nachstehende Kommandos notwendig:

- `cp lib/libmm.so.10.0.12 /usr/local/lib`
- `cd /usr/local/lib && ln -s libmm.so.10.0.12 libmm.so`
- `ln -s libmm.so.10.0.12 libmm.so.10`
- `chmod 755 libmm.so.10.0.12`
- `cp include/mm.h /usr/local/include`
- `chmod 644 /usr/local/include/mm.h`

Damit ist die Installation des MM-Pakets abgeschlossen.

14.4 Das Paket `openssl-0.9.4.tar.gz`

Wie schon erwähnt wurde, fällt dieser Abschnitt knapp aus, da OpenSSL ein ganzer separater Teil dieses Handbuches gewidmet ist (Teil II).

Auspacken der Quellen in einem Verzeichnis (z.B. `/usr/src`):

```
gzip -dc openssl-0.9.4.tar.gz | tar -xvf -
```

Mit dem folgenden Befehl (nach dem Wechsel in das OpenSSL-Quell-Verzeichnis) wird das Paket konfiguriert:

```
sh config --prefix=/usr/local
```

Die Option `--prefix=`*xxx* ermöglicht die Angabe eines Pfades, unter dem nach der Übersetzung Bibliotheken (*xxx*`/lib`), Header-Dateien (*xxx*`/include/openssl`), Binaries (*xxx*`/bin`), sowie verschiedene Dateien zur Verwaltung und Herausgabe von Zertifikaten (*xxx*`/ssl`) installiert werden. Genaueres dazu steht in der Dokumentation des Pakets.

Übersetzen des Pakets durch Aufruf von

```
make
```

Ein Test des übersetzten Pakets wird durch folgendes Kommando durchgeführt:

```
make test
```

Bei erfolgreichem Test wird am Ende des Testlaufs eine Meldung ähnlich der folgenden ausgegeben:

```
Signed certificate is in newcert.pem
newcert.pem: OK
OpenSSL 0.9.4 09 Aug 1999
built on: Thu Aug 12 12:14:31 MET DST 1999
platform: solaris-sparcv8-gcc
options:  bn(64,32) md2(int) rc4(ptr,char) des(idx,cisc,16,long) idea(int) blowfish(ptr)
compiler: gcc -DTHREADS -D_REENTRANT -mv8 -O3 -fomit-frame-pointer -Wall -
DB_ENDIAN -DBN_DIV2W
`test' is up to date.
```

Abschließend kann das Paket mit dem folgenden Befehl installiert werden:

```
make install
```

Sollen die Dateien statt mit dem `install`-Befehl durch Kopieren installiert werden, sind folgende Kommandos notwendig:

(*$(SSLDIR)=Installationspfad von OpenSSL, z.B.* `/usr/local`):

- `mkdir $(SSLDIR)/include $(SSLDIR)/include/openssl $(SSL-DIR)/bin $(SSLDIR)/lib`
- `cd /usr/src/openssl-0.9.4`

- `cp lib* $(SSLDIR)/lib/`
- `cp include/openssl/* $(SSLDIR)/include/openssl/`
- `cp apps/openssl tools/c_rehash $(SSLDIR)/bin/`

Optional (weil nicht zur Übersetzung des SSL-Apache erforderlich) kann jetzt noch eine Verzeichnis-Struktur zur Herausgabe von Zertifikaten mittels des OpenSSL-Pakets etabliert werden:

- `mkdir $(SSLDIR)/ssl $(SSLDIR)/ssl/private $(SSLDIR)/ssl/misc $(SSLDIR)/ssl/lib $(SSLDIR)/ssl/certs`
- `cd /usr/src/openssl-0.9.4/tools`
- `cp c_hash c_issuer c_info c_name .$(SSLDIR)/ssl/misc`
- `cd /usr/src/openssl-0.9.4/apps`
- `cp CA.pl CA.sh der_chop $(SSLDIR)/openssl/misc`
- `cp apps/openssl.cnf $(SSLDIR)/ssl`

Die vernünftige Nutzung dieser Struktur, also letztlich der Betrieb einer CA, setzt Wissen über die Handhabung von Zertifikaten voraus. Das gilt besonders, wenn Zertifikate für den SSL-Apache und für Anwendungen selbst herausgegeben werden sollen. Aber auch, wenn „lediglich“ ein Request für eine Zertifizierung durch eine externe CA erzeugt werden soll, ist es unumgänglich, sich etwas eingehender mit dem Paket zu beschäftigen.

Wie eine Serverzertifikat-Anforderung (*Request*) erzeugt werden kann, steht in Kapitel 15.2.

Damit ist die Installation des OpenSSL-Pakets abgeschlossen.

14.5 Das Paket `mod_ssl-2.4.10-1.3.9`

Das Mod-SSL-Paket enthält eine umfangreiche Dokumentation zur Installation und zum Betrieb des SSL-Apache. Sie ist im Quell-Verzeichnis des Pakets unter `pkg.ssldoc` zu finden.

In diesem Abschnitt wird die Konfiguration des Mod-SSL-Pakets beschrieben. Dieses Paket wird nicht separat übersetzt, da es dazu dient, die Apache-Quellen zu verändern bzw. zu erweitern. Es werden die notwendigen Schnittstellen zum OpenSSL-Paket in die Apache-Quellen „gepatched“. Die so veränderten Apache-Quellen müssen dann später übersetzt werden.

Auspacken der Quellen in einem Verzeichnis (z.B. in `/usr/src`):

```
gzip -dc mod_ssl-2.4.10-1.3.9.tar.gz | tar -xvf -
```

Bevor das Mod-SSL-Paket konfiguriert werden kann, müssen zunächst die Apache-Quellen entpackt werden (z.B. in `/usr/src`):

```
gzip -dc apache_1.3.9.tar.gz | tar -xvf -
```

Mit den folgenden Befehlen wird das Mod-SSL-Paket konfiguriert und gleichzeitig die notwendigen Veränderungen am Apache-Quellcode vorgenommen:

- `cd mod_ssl-2.4.10-1.3.9`
- `sh ./configure --with-apache=../apache_1.3.9`

Bei der Option `--with-apache=` muß der Pfad angegeben werden, der auf die zuvor entpackten Apache-Quellen weist.

Verlief die Ausführung des vorstehenden Kommandos erfolgreich, wird eine abschließende Meldung ausgegeben:

```
Done: source extension and patches successfully applied.
```

14.6 Das Paket apache_1.3.9

Das Apache-Paket enthält eine umfangreiche Dokumentation des Servers im HTML-Format. Die Dokumentation befindet sich im Quell-Verzeichnis unterhalb des Verzeichnisses `htdocs`. Ebenfalls in diesem Verzeichnis befindet sich nach dem „Patchen" der Apache-Quellen die Mod-SSL-Dokumentation.

Der Apache-Web-Server besteht aus verschiedenen Modulen, die die Funktionalität des Servers bestimmen. Die Auswahl dieser Module ist abhängig vom vorgesehenen Einsatz des Servers: z.B. davon, ob CGI-Skripte oder Benutzerverzeichnisse zugelassen werden sollen. Zur Verwendung von CGI-Skripten ist dann das Modul `mod_cgi` erforderlich, und zur Unterstützung von User-Verzeichnissen das Modul `mod_userdir`. Daher ist es unumgänglich, sich mit den Modulen und deren Funktionalität auseinanderzusetzen. Je nach gewünschtem Einsatz des Servers werden die Module bei dem untenstehenden Konfigurationsbefehl mit angegeben.

Die im Folgenden vorgestellte Konfiguration erlaubt u.a. den Betrieb eines SSL-fähigen HTTP-Servers und die Verwendung von CGI-Skripten. Die vorgestellte Konfiguration unterstützt aber beispielsweise nicht die automatische Generierung von Index-Dateien in FTP-Verzeichnissen. Benutzerverzeichnisse werden ebenfalls nicht unterstützt.

Zur Übersetzung kann der Apache auf zwei verschiedene Arten konfiguriert werden: mit Unterstützung für zur Laufzeit ladbare Module, den sogenannten „DSO-Support" (DSO, *Dynamic Shared Object*), und ohne diesen DSO-Support.

In Hinsicht auf die SSL-Fähigkeit des Servers unterscheiden sich die beiden Konfigurationsvarianten in folgender Weise:

- *Werden DSO-Module vom Server nicht unterstützt*, müssen die Original-Server-Quellen bei Herausgabe einer neuen Mod-SSL-Version erneut „gepatched" und übersetzt werden.

- *Unterstützt der Server DSO-Module*, kann die Laufzeitkonfiguration des Servers (festgelegt in der Datei `httpd.conf`) einfach geändert werden. Erfordert eine neue Konfigurationsdirektive die Funktionalität eines nicht geladenen Moduls, kann dieses nachgeladen werden. Nicht benötigte Module werden nicht geladen. Voraussetzung ist, daß sämtliche Module übersetzt vorliegen.

  Neue Mod-SSL-Version können übersetzt und anstelle des alten SSL-Moduls geladen werden. So muß nur das Modul und nicht der ganze Server übersetzt werden. Genaueres steht im Abschnitt 14.7.

  Nachteilig an der DSO-Variante sind leichte Performanz-Einbußen zur Laufzeit und eine etwas längere Startdauer des Servers.

Denkbar ist auch, zunächst einen DSO-Server mit sämtlichen Modulen zu übersetzen. Mit diesen können dann verschiedene Modul-Konfigurationen getestet werden. Der letztlich eingesetzte Server könnte dann nur mit den benötigten Modulen in der Nicht-DSO-Variante übersetzt und anschließend eingesetzt werden.

Es sei hier angemerkt, daß unnötige Module Speicherplatz kosten (in der DSO-Variante nur, wenn sie geladen werden) und das Risiko erhöhen, daß der Server durch falsche Konfiguration kompromittierbar wird.

14.6.1 Apache ohne DSO-Support

Die gewünschten Module werden beim Aufruf des Konfigurations-Befehls `configure` angegeben. Wegen der verschiedenen Möglichkeiten, diese Module zusammenzustellen, wird im Folgenden eine Beispielkonfiguration aufgeführt, die mit der Server-Konfigurationsdatei im Anhang G funktioniert. Die unten angegebenen Module ermöglichen den Betrieb eines SSL-Servers und erlauben die Verwendung von CGI-Skripten.

Die Quellen des Apache liegen, wie im Abschnitt 14.5 beschrieben, „gepatched“ vor, z.B. unter `/usr/src/apache_1.3.9`. Nach Wechsel in das Apache-Quell-Verzeichnis erfolgt die Konfiguration durch das folgende, etwas längere Kommando:

```
env SSL_BASE=/usr/local \
EAPI_MM=/usr/local \
CC="gcc" OPTIM="-O3 -mv8 -fomit-frame-pointer" \
sh ./configure --prefix=/usr/local/apache \
             --disable-module=all        --disable-rule=SSL_COMPAT \
             --enable-module=env \
             --enable-module=log_config  --enable-module=mime \
             --enable-module=negotiation --enable-module=dir \
             --enable-module=cgi         --enable-module=actions \
             --enable-module=access      --enable-module=setenvif \
             --enable-module=alias       --enable-module=ssl
```

Die Variable `SSL_BASE` weist auf den Zweig des Dateisystems, in dem das OpenSSL-Paket installiert bzw. übersetzt wurde. Das gleiche gilt für `EAPI_MM` in Bezug auf die MM-Bibliothek. Der bei der Option `--prefix=` angegebene Pfad veranlaßt, daß bei einem späteren Aufruf von `make install` sämtliche Apache-Dateien unter diesem Verzeichnis in verschiedene Unterverzeichnisse installiert werden.

Die einfachste Möglichkeit, ein feiner angepaßtes Installations-Layout zu bekommen, besteht in der Anpassung der Datei `config.layout` im Apache-Quellverzeichnis (`apache_1.3.9`). Zu dieser Layout-Datei kann ein eigenes benanntes Layout hinzugefügt werden, welches statt mit der Option `--prefix=`*xxx* mit der Option `--with-layout=`*layoutname* angegeben wird. Es ist auch möglich, die einzelnen Dateien nach der Übersetzung durch Kopieren an die gewünschte Stelle zu kopieren. Diese Variante wird im Abschnitt 14.6.3 vorgestellt.

Die Übersetzung wird durch folgendes Kommando gestartet:

```
make
```

Die Übersetzung sollte ohne Fehler oder Warnungen erfolgen.

14.6.2 Apache mit DSO-Support

Die in diesem Abschnitt vorgestellte Konfiguration konfiguriert die Server-Quellen so, daß sämtliche möglichen Module des Servers übersetzt und installiert werden. Welche Module dann tatsächlich für den Betrieb des Servers geladen werden, wird über die Konfigurationsdatei (siehe Anhang G) festgelegt. (Alternativ können auch nur die tatsächlich benötigten Module durch mehrfache Verwendung der Option `--enable-shared=`*Modulname* bei untenstehendem Konfigurations-Kommando angegeben werden.)

Damit der Apache die Bibliotheken des MM-Pakets findet, muß die Umgebungsvariable `LD_LIBRARY_PATH` den Pfad `/usr/local/lib` enthalten.

Der folgende Befehl konfiguriert die Apache-Quellen so, daß sämtliche Apache-Module und das SSL-Modul kompiliert werden:

```
env SSL_BASE=/usr/local \
    EAPI_MM=/usr/local \
    CC='gcc' OPTIM='-O3 -mv8 -fomit-frame-pointer' \
sh ./configure \
   --prefix=/usr/local/apache \
   --enable-shared=ssl \
   --disable-rule=SSL_COMPAT \
   --enable-shared=max \
   --enable-shared=remain
```

Sollen einzelne Module von der Konfiguration ausgenommen werden, kann die Konfigurations-Option `--disable=`*Modulname* eingesetzt werden:

```
env SSL_BASE=/usr/local \
    EAPI_MM=/usr/local \
    CC='gcc' \
    OPTIM='-O3 -mv8 -fomit-frame-pointer' \
sh ./configure \
   --prefix=/usr/local/apache \
   --enable-shared=ssl \
   --enable-shared=max \
   --enable-shared=remain \
   --disable-shared=auth_db \
   --disable-module=auth_db \
   --disable-shared=auth_digest \
   --disable-module=auth_digest
```

Die Variable `SSL_BASE` weist auf den Zweig des Dateisystems, in dem das OpenSSL-Paket installiert bzw. übersetzt wurde. Das gleiche gilt für `EAPI_MM` in Bezug auf die MM-Bibliothek. Der bei der Option `--prefix=` angegebene Pfad veranlaßt, daß bei einem späteren Aufruf von `make install` sämtliche Apache-Dateien unter diesem Verzeichnis in verschiedene Unterverzeichnisse installiert werden.

Die einfachste Möglichkeit, ein feiner angepaßtes Installations-Layout zu bekommen, besteht in der Anpassung der Datei `config.layout` im Apache-Quellverzeichnis (`apache_1.3.9`). Zu dieser Layout-Datei kann ein eigenes benanntes Layout hinzugefügt werden, welches statt der Option `--prefix=`*xxx* mit der Option `--with-layout=`*layoutname* angegeben wird. Es ist auch möglich, die einzelnen Dateien nach der Übersetzung an die gewünschte Stelle zu kopieren. Diese Variante wird im nachfolgenden Abschnitt 14.6.3 vorgestellt.

Die Übersetzung erfolgt durch Aufruf von

```
make
```

Es ist möglich, die gewünschten Module alle explizit aufzuführen. Die im vorigen Abschnitt 14.6.1 vorgestellte Nicht-DSO-Konfiguration wird zur DSO-Konfiguration, wenn in der Konfiguration alle `--enable-module` Optionen durch `--enable-shared` ersetzt werden.

14.6.3 Installation des SSL-Apache durch Kopieren

Nachfolgend wird die Installation eines „normalen“ HTTP-Servers auf Port 80 und eines SSL-Servers auf Port 443 beschrieben.

Die vorgestellte Installation ist beispielhaft zu verstehen und kann von Fall zu Fall sehr unterschiedlich aussehen. Die Installation kann durch `make install` automatisch erfolgen. Dabei werden aber auch Dateien installiert, die eher für einen Test-Betrieb gedacht sind. Daher wird auf diese Option hier nicht weiter eingegangen und im folgenden gezeigt, wie die notwendigen Dateien durch Kopieren installiert werden.

Die Installation der Dateien und die Betriebs-Konfiguration des Servers sind voneinander abhängig. Die Betriebs-Konfiguration des Servers erfolgt über eine Konfigurationsdatei. Die Beispiel-Konfigurationsdatei im Anhang G beschreibt den Betrieb eines Nicht-SSL-Servers (üblicherweise auf Port 80) und eines SSL-Servers auf Port 443. Dafür werden zwei Hauptverzeichnisse angelegt. Das erste nimmt innerhalb von fünf oder sechs Unterverzeichnissen die Konfigurationsdateien, Log-Dateien, CGI-Skripte, Icons, SSL-Dateien (Zertifikate u.a.) und eventuell die Apache-Module (eines DSO-Servers) auf. Das andere Hauptverzeichnis enthält zwei Unterverzeichnisse, die für je einen Server das sogenannte „Document-Root-Verzeichnis" bilden. Diese Unterverzeichnisse nehmen die HTML-Seiten auf, die von dem jeweiligen Server gesendet werden sollen.

Zum Aufbau der Verzeichnisstruktur sind folgende Befehle notwendig:

(`$(APADIR)`=*Apache-Installationspfad*, z.B. `$(APADIR)=/usr/local/apache`),
(`$(WWW)`=*HTML-Root-Verzeichnis*, z.B. `$(WWW)=/var/www`)

- `mkdir $(APADIR) $(APADIR)/conf $(APADIR)/logs $(APADIR)/icons $(APADIR)/cgi-bin $(APADIR)/ssl $(APADIR)/ssl/cacerts $(APADIR)/ssl/crls`
- `mkdir $(WWW) $(WWW)/www80 $(WWW)/www443`

Wurde der Server mit DSO-Unterstützung übersetzt, muß zusätzlich noch ein Verzeichnis `libexec` angelegt werden, welches die übersetzten Apache-Module aufnimmt.

- `mkdir $(APADIR)/libexec`

Die `man`-Seiten, die ausführbaren Dateien bzw. Skripte sowie die HTML-Dokumentation des Servers werden im entsprechenden Zweig des Dateibaums installiert, also z.B. unterhalb `/usr/local/man` bzw. in `/usr/local/bin` und `/usr/local/doc`. Die Dokumentation bzw. die Manual-Seiten sind zur Laufzeit nicht erforderlich und können bei Bedarf weggelassen werden.

Jetzt können die Dateien kopiert werden.

Für einen ersten Test des Servers kann die mit dem Apache und die mit dem Mod-SSL-Paket gelieferte Hauptseite kopiert werden.

1. `cd /usr/src/apache_1.3.9`
2. `cp src/httpd /usr/local/bin`
3. `cd src/support`
4. `cp ap apachectl apxs dbmmanage htdigest htpasswd log_server_status logresolve split_logfile suexec /usr/local/bin`
   Diese Dateien müssen nicht alle vorhanden sein, da die Übersetzung in Abhängigkeit von den gewählten Modulen erfolgt.

5. `cd /usr/src/apache_1.3.9`
6. `cp conf/mime.types $(APADIR)/conf`
7. `cp conf/httpd.conf-dist $(APADIR)/conf/httpd.conf`
8. `cp support/*.1 /usr/local/man/man1`
9. `cp support/*.8 /usr/local/man/man8`
10. `cp htdocs/index.html $(WWW)/www80`
11. `cp htdocs/manual/mod/mod_ssl/index.html $(WWW)/www443`
12. `cp htdocs/apache_pb.gif $(APADIR)/icons`

Wurde der Server mit DSO-Unterstützung übersetzt, müssen noch die Module kopiert werden. Da diese zum Teil auf diverse Unterverzeichnisse verteilt sind, wird das `find`-Kommando eingesetzt:

- `cp ‘find . -name "*.soprint‘ $(APADIR)/libexec/`

Wichtig:
In jedem Fall ist vor Aufnahme des regulären Betriebs des Servers die Konfigurationsdatei `httpd.conf` im Verzeichnis `$(APADIR)/conf` anzupassen. Eine Beispieldatei findet sich im Anhang G.

14.6.4 Gruppen- und Benutzerrechte

Aus Sicherheitsgründen sollte für den Server unbedingt ein eigener Benutzer (hier `httpsd`) und eine eigene Gruppe (hier `www`) eingerichtet werden. Der Server muß vom Benutzer `root` gestartet werden, damit der Server sich an die privilegierten Ports (80 und 443) binden kann. Anschliessend werden Kind-Prozesse gestartet, die die Anfragen bedienen. Der Eltern-Prozeß läuft unter der User-ID `root`, die Kind-Prozesse unter der in der Konfigurationsdatei angegebenen (weniger privilegierten) User- bzw. Group-ID, also z.B. `httpsd` bzw. `www`. Weiterhin ist es sinnvoll, einen eigenen User (hier `wwwadm`) zur Pflege der Web-Seiten einzurichten. Der User `wwwadm` gehört dann derselben Gruppe wie der Server an (hier `www`).

Im folgenden ein Vorschlag zum Setzen von Benutzern, Gruppen und den Rechten.

Es soll hier nochmal betont werden, daß große Aufmerksamkeit bei Vergabe der Rechte und Festlegung der Gruppen walten sollte. Ein falsch konfigurierter Server eröffnet möglicherweise einen nicht autorisierten Zugang zum Rechner! Insbesondere sollten Dateien, die von Prozessen mit der UID root *gelesen werden, nicht von anderen geschrieben werden können.*

Setzen von Benutzer und Gruppe:

- `chown -R root:root $(APADIR)`

- `chown -R wwwadm:www $(APADIR)/icons $(APADIR)/cgi $(APADIR)/ssl $(www)/www80 $(WWW)/www443`

Setzen der Rechte:

- `chmod -R 755 $(APADIR)`
- `chmod -R 700 $(APADIR)/logs $(APADIR)/ssl`
- `chmod 750 $(APADIR)/conf $(APADIR)/icons $(APADIR)/cgi $(APADIR)/crls \`

  `$(APADIR)/cacerts $(WWW)/www80 $(WWW)/www443`
- `chmod 600 $(APADIR)/conf/*`
- `chmod 640 $(APADIR)/icons/* $(WWW)/www80/* $(WWW)/www443/*`

Für das DSO-Module-Verzeichnis:

- `chown root:root $(APADIR)/libexec/*`
- `chmod 755 $(APADIR)/libexec/*`

Zum Starten bzw. Stoppen des Servers siehe Abschnitt 15.5.

14.7 Übersetzung von neuen Mod-SSL-Versionen

Bevor ein neues Mod-SSL-Paket mit dem Apache in Betrieb genommen wird, sollte auf jeden Fall die Datei `CHANGES` *des Mod-SSL-Pakets gelesen werden.* Möglicherweise wurden nicht nur Fehler beseitigt, sondern auch neue Konfigurationsdirektiven eingeführt oder alte Direktiven geändert. Entsprechend müßte dann vor dem Start des Servers eine Anpassung der Konfigurationsdatei vorgenommen werden.

Wurde der Apache *ohne DSO-Support* installiert, also die Module statisch in den Apache eingebunden, muß der Apache zusammen mit dem neuen Mod-SSL-Paket komplett übersetzt und installiert werden (wie in 14.6 beschrieben). Dazu müssen die Apache-Quellen erneut ausgepackt werden, und nicht etwa die „alten“, schon „gepatchten“ Quellen wiederverwendet werden.

Wurde der Apache *mit DSO-Support und SSL-Modul* installiert, vereinfacht sich die Installation der neuen Mod-SSL-Version. Das Vorgehen wird im folgenden beschrieben.

Es wird angenommen, daß die Installation des OpenSSL-Pakets und des Apache unter `/usr/local` erfolgte. Insbesondere das mitgelieferte Perl-Skript mit Namen `apxs` muß unter dem angegebenen Pfad zur Verfügung stehen. In diesem Skript sind u.a. die Pfade eingetragen, in denen der Apache bzw. seine Komponenten installiert wurden.

- `cd /usr/src`
- gzip -dc mod_ssl-2.4.*x*-1.3.9.tar.gz
- cd mod_ssl-2.4.*x*-1.3.9
- ```
  env CC=gcc sh ./configure \
                --with-apxs=/usr/local/bin/apxs \
                --with-ssl=/usr/local
  ```
- `make`

Vor Ausführung des nächsten Befehls sollte das alte Modul umbenannt werden, um im Zweifelsfall die letzte (lauffähige) SSL-Apache-Version wiederherstellen zu können. Anschließend wird dann das neue Modul kopiert:

- `cp /usr/local/apache/libexec/libssl.so /usr/local/apache/libexec/libssl.so.old`
- cp mod_ssl-2.4.*n*-1.3.9/pkg.sslmod/libssl.so /usr/local/apache/libexec/
- `chown root:root $(APADIR)/libexec/libssl.so`
- `chmod 755 /usr/local/apache/libexec/libssl.so`

Dann muß der Server gestoppt und gestartet werden:

1. ``kill -TERM `cat /usr/local/apache/logs/httpsd.pid` ``
2. `/usr/local/bin/httpd -DSSL`

Genaueres zum Starten bzw. Stoppen steht in Kapitel 15.5.
  ```

Kapitel 15

Betrieb des SSL-Apache

Bevor der SSL-Apache in Betrieb genommen werden kann, muß normalerweise ein Schlüsselpaar nebst Zertifikat für den Server erzeugt werden. Ohne dieses Server-Zertifikat kommt keine SSL- und somit keine verschlüsselte Verbindung zustande. Das Zertifikat enthält den Public-Key sowie u.a. den Namen des Servers. Bei einem SSL-Verbindungswunsch des Client wird das Server-Zertifikat vom Server an den Client geschickt. Wurde das Zertifikat durch eine dem Client bekannte CA herausgegeben (d.h. das CA-Zertifikat liegt dem Client vor), kann der Client die Gültigkeit der Signatur des Server-Zertifikats mit Hilfe des Public-Keys im CA-Zertifikat prüfen. Ist die Signatur in Ordnung und das Zertifikat gültig, kann es akzeptiert werden.

Abhängig vom Browser und seiner Konfiguration wird der Benutzer von der Zertifikat-Prüfung u.U. nichts merken, wenn das CA-Zertifikat zuvor in den Browser importiert und als gültig anerkannt wurde. Liegt dem Client jedoch kein CA-Zertifikat vor, entscheidet der Benutzer, ob das Zertifikat vertrauenswürdig ist oder nicht. Der Benutzer muß dazu üblicherweise eine vom Browser initiierte Interaktion „durchklicken“, an deren Ende der Benutzer die Gültigkeit des Server-Zertifikats explizit anerkennt. Damit ist der Server authentifiziert. Anschließend werden dann Session-Keys vereinbart, so daß die Nutz-Informationen verschlüsselt übertragen werden können.

15.1 Virtuelle Server

Es ist möglich, mehrere *virtuelle Server* zu betreiben. Diese Server werden vom selben Eltern-Prozeß gestartet und können über verschiedene Namen angesprochen werden. Jeder Server kann sein eigenes Document-, CGI-Verzeichnis usw. haben. Auf diese Weise können mehrere WWW-Server auf demselben Rechner verwaltet werden.

Die Namen dieser (virtuellen) Server werden alle derselben IP-Nummer zugeordnet und müssen in einem Nameserver eingetragen sein. Der Apache wählt dann den gewünschten Server anhand des von den (meisten) Browsern mitgesendeten Servernamen aus. Mehrere SSL-Server können auf diese Weise *nicht* betrieben werden, denn bevor der Browser den Namen des gewünschten Servers sendet, erfolgt das SSL-Handshake. Zum Zeitpunkt des Handshakes sind jedoch lediglich IP- und Port-Nummern bekannt. Daher kann für **jede IP- und Port-Nummer nur ein SSL-Server aufgesetzt werden**. Daraus ergeben sich zwei Möglichkeiten, mehrere virtuelle SSL-Server zu betreiben: durch

Zuordnung von mehreren IP-Nummern zum Netzwerk-Interface bzw. durch die Verwendung von mehreren Ports.

Erfolgt der Einsatz der virtuellen SSL-Server mittels unterschiedlicher IP-Nummern, werden diese IP-basiert genannt. Zu der Verwendung von anderen Port-Nummern als 443 für einen SSL-Server ist anzumerken, daß die meisten Ports unterhalb von 1024 für andere Dienste vergeben sind. Zudem muß die Port-Nummer dann in der gewünschten URL mit angegeben werden, was aus Anwendersicht nicht sehr komfortabel ist. Virtuelle SSL-Server sind also möglich, sollten aber IP-basiert sein.

15.2 Erzeugen eines Server-Zertifikats

In diesem Abschnitt werden zwei Möglichkeiten vorgestellt, ein Server-Zertifikat zu erzeugen. Die erste Möglichkeit verwendet die vom Mod-SSL-Paket zur Verfügung gestellten Hilfsmittel. Es wird ein Server-Test-Zertifikat erzeugt, welches durch eine Test-CA signiert wurde. Die zweite Variante beschreibt die Erzeugung eines Requests mit anschließender Zertifizierung durch eine reale CA. Beide Varianten verwenden das `openssl`-Binary des SSL-Pakets *OpenSSL*. Da der SSL-Apache ohne kompiliertes OpenSSL-Paket nicht zu übersetzen wäre, sollte dieses Binary vorhanden sein.

15.2.1 Erzeugen eines Zertifikats durch `make certificate`

In diesem Abschnitt beschriebene Mod-SSL-Direktiven (siehe Dokumentation des Mod-SSL-Pakets):

- `SSLCertificateFile`
- `SSLCertificateKeyFile`
- `SSLCACertificateFile`
- `SSLCACertificatePath`

Nachdem der Apache übersetzt wurde, kann durch Aufruf von `make certificate` im Apache-Quell-Verzeichnis ein Server-Zertifikat mit zugehörigem 1024-Bit-Schlüssel erstellt werden. In einzelnen Schritten („`STEP 1-4`“) werden die Angaben zur Erzeugung einer Server-Zertifikatanforderung (Serverrequest) abgefragt. Die Fragen sind sehr ausführlich und sollten keine Probleme bereiten. U.a. werden Angaben zu Land, Bundesstaat usw. abgefragt. Wichtig ist die Angabe `Common Name`: Hier ist unbedingt der Server-Name anzugeben, also z.B. `www.name.de`. Andernfalls werden Browser eine Warnung ausgeben, daß der Server möglicherweise nicht derjenige ist, als der er sich ausgibt.

Im letzten Schritt („`STEP 4`“) wird nach einer Passphrase gefragt, mittels der der Private-Key des Servers geschützt werden soll. Diese Passphrase wird bei jedem Starten des Servers abgefragt, um den Private-Key entsperren zu können. (Diskussionen über Vor- und Nachteile eines verschlüsselten Server-Keys tauchen immer mal wieder auf, und auch hier wird keine „letzte“ Antwort gegeben.

Klar ist, daß die Angabe einer Passphrase beim Start des Servers ein unbeaufsichtigtes Starten erschwert.)

Abschließend wird der Request zertifiziert und liegt dann im Verzeichnis `conf/ssl.crt/` unter dem Namen `server.crt` vor. Das zugehörige (von `Mod-SSL` mitgelieferte) CA-Zertifikat findet sich im Apache-Quell-Baum unter `conf/ssl.crt/snakeoil-ca-rsa.crt`. Für einen Test können die beiden Zertifikate und der Server-Key `conf/ssl.key/server.key` in das Apache-Verzeichnis kopiert werden:

- `cp conf/ssl.crt/server.crt $(APADIR)/ssl/serverCert.pem`
- `cp conf/ssl.key/server.key $(APADIR)/ssl/serverKey.pem`
- `cp conf/ssl.crt/snakeoil-ca-rsa.crt $(APA-DIR)/ssl/cacerts/snakeCA.pem`

Rechte und Gruppe anpassen:

- `chown root:other $(APADIR)/ssl/*.pem`
- `chmod 400 $(APADIR)/ssl/*.pem`
- `chown -R www:wwwadm $(APADIR)/cacerts`
- `chmod 440 $(APADIR)/cacerts/*`

Die entsprechenden Konfigurations-Direktiven müssen in der Konfigurationsdatei `httpd.conf` (siehe Anhang G) gesetzt sein, z.B.:

`SSLCertificateFile /usr/local/apache/ssl/serverCert.pem`

Die Direktive weist auf die Datei mit dem Serverzertifikat.

`SSLCertificateKeyFile /usr/local/apache/ssl/serverKey.pem`

Die Direktive weist auf die Datei mit dem Private-Key des Servers. Sollte sich der Server-Key zusammen mit dem Server-Zertifikat in einer Datei befinden, muß hier ebenfalls die Zertifikatdatei angegeben werden.

`SSLCACertificateFile /usr/local/apache/ssl/cacerts/snakeCA.pem`

Die Direktive weist auf die Datei mit dem CA-Zertifikat. CA-Zertifikate werden benötigt, um eine Client-Zertifikatprüfung durchzuführen. Es können mehrere CA-Zertifikate in *eine Datei* geschrieben werden: z.B. `cat ca1.pem ca2.pem ca3.pem > cas.pem`.

`SSLCACertificatePath /usr/local/apache/ssl/cacerts`

Die Direktive dient ebenfalls dazu, dem Server Zugriff auf CA-Zertifikate zu geben. Der Unterschied zu `SSLCACertificateFile` liegt in der Verwaltung der CA-Zertifikate. Für `SSLCACertificatePath` müssen alle Zertifikate *einzeln* in das benannte Verzeichnis kopiert werden (hier: `.../cacerts`). Anschließ werden sie dann über ihre Hash-Werte durch die Ausführung des Shell-Skripts `c_rehash` (aus dem OpenSSL-Paket) verlinkt. Ohne diese Links findet der Apache die CA-Zertifikate nicht:

- `cd $(APACHE)/ssl/cacerts`
- `chmod 640 $(APADIR)/ssl/cacerts/*`
- `c_rehash`

15.2.2 Zertifizierung durch eine CA

Bei dieser Methode wird zunächst ein Request erzeugt, welcher anschließend durch eine CA signiert wird. Diese CA kann natürlich auch vom Betreiber des Servers realisiert werden; hierzu kann ebenfalls das OpenSSL-Paket eingesetzt werden. Auf diese Variante der Realisierung einer CA wird hier nicht weiter eingegangen.

Zur Erzeugung eines Requests muß zunächst ein Schlüsselpaar generiert werden. Mit folgendem Befehl wird ein 1024 Bit-Schlüssel erzeugt (bei Bedarf ist der OpenSSL-Pfad zu ergänzen):

```
openssl genrsa -out serverKey.pem -rand file1:file2:... 1024
```

Vorher sollte allerdings die Umgebungsvariable `RANDFILE` gesetzt und der Zufallszahlen-Status initialisiert worden sein. Dazu kann irgendeine Datei mit (Pseudo-)Zufallsdaten nach `$(RANDFILE)` kopiert werden. Die nach der Option `-rand` angegebenen Dateien dienen als zusätzliche Zufallsdaten für die Schlüsselerzeugung.

Das `genrsa`-Kommando erzeugt ein unverschlüsseltes Schlüsselpaar, der Private-Key ist also von jedem verwendbar. Ist eine Verschlüsselung des Private-Keys gewünscht, kann bei obigem Kommando zusätzlich die Option `-des3` angegeben werden.

Die Erzeugung des Requests erfolgt anschließend durch folgenden Befehl:

```
openssl req -new -inkey serverKey.pem -out severReq.pem
```

Nach Eingabe des Befehls müssen folgende Angaben gemacht werden (beispielhafte Eingaben sind in doppelten Anführungsstrichen):

```
Using configuration from /usr/local/openssl/openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:"DE"
State or Province Name (full name) [Some-State]:"Schleswig-Holstein"
Locality Name (eg, city) []:"Kiel"
Organization Name (eg, company) [Internet Widgits Pty Ltd]:"WWW UnLtd"
Organizational Unit Name (eg, section) []:"Abt. Web-Server"
Common Name (eg, YOUR name) []:"www.www-unltd.de"
Email Address []:"wwwadmin@www-unltd.de"

Please enter the following "extra" attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

(Für den genauen Inhalt muß die zertifizierende CA bzw. deren Policy konsultiert werden.)

Achtung:
Soll der signierte Request später als SSL-Server-Zertifikat verwendet werden, muß als `Common Name` unbedingt der Server-Name angegeben werden!

Die beiden letzten Angaben (`challenge password` und `optional company name`) tauchen als Klartext im Attribut-Bereich des Requests auf. Soll später ein Zertifikat zurückgerufen werden, kann sich der Inhaber gegenüber der CA, auch bei Verlust des Private-Keys, durch Angabe dieser Felder als Inhaber „ausweisen“. Die Verwaltung dieser Angaben kann von der unterzeichnenden CA durchgeführt werden. Ob die signierende CA diese Felder unterstützt, sollte mit der CA geklärt werden.

Der so erzeugte Request (die Datei `serverReq.pem`) kann dann einer CA zum Signieren vorgelegt werden. Alternativ kann auch eine eigene (Test-)CA installiert und der Request signiert werden. Auf diese Möglichkeit wird hier nicht weiter eingegangen. Liegt das Server-Zertifikat später vor, werden das Zertifikat und der Private-Key nach `$(APADIR)/ssl` kopiert.

Für die Konfiguration der Mod-SSL-Direktiven gelten die im Abschnitt 15.2.1 gemachten Anmerkungen.

15.3 Prüfung von Client-Zertifikaten

In diesem Abschnitt werden die folgenden Mod-SSL-Direktiven beschrieben (siehe Dokumentation des Mod-SSL-Pakets):

- `SSLVerifyClient`
- `SSLVerifyDepth`
- `SSLRequire`
- `SSLOptions`
- `SSLRequireSSL`

Ist zusätzlich zu der obligatorischen Server-Authentifizierung (siehe Kapitel 15) noch eine Client-Authentifizierung erforderlich, schickt der Client sein Zertifikat während des SSL-Handshakes an den Server. Der Server sendet dazu dem Client eine zufällige Nachricht, die dieser dann signiert zusammen mit seinem Zertifikat an den Server sendet. Dieser kann dann die Signatur mit dem Public-Key aus dem Client-Zertifikat überprüfen. Das Client-Zertifikat selbst wird dann ebenfalls überprüft, wozu das Zertifikat der Herausgeber-CA erforderlich ist.

SSLVerifyClient

Der SSL-Server kann in Bezug auf die Client-Authentifizierung auf vier verschiedene Arten konfiguriert werden (Schlüsselwort `SSLVerifyClient`):

`none`

Es ist kein Zertifikat erforderlich, es erfolgt also keine Client-Authentisierung.

`optional`

Ein gültiges Zertifikat wird akzeptiert, ist aber nicht erforderlich.

`require`

Ein gültiges Zertifikat muß vorgelegt werden. Außerdem muß es überprüfbar sein, der Server muß also das Zertifikat der Unterzeichner-CA vorliegen haben.

`optional_no_ca`

Ein gültiges Zertifikat kann vorgelegt werden, muß aber nicht überprüfbar sein.

Für einen realen Betrieb kommen `none` oder `require` in Frage. Die anderen Werte können für einen Test-Server interessant sein.

Durch Setzen der `SSLVerifyClient`-Direktive auf `require` erhält ein Client erst nach erfolgter Authentisierung Zugriff auf die HTML-Seiten eines Servers. Es ist aber auch möglich, dem Client Zugriff auf das Document-Root-Verzeichnis ohne Zertifikat zu gewähren (`SSLVerifyClient none`) und dann den Zugriff auf Seiten in einem Unterverzeichnis erst nach erfolgreicher Client-Authentisierung zuzulassen. Dazu wird eine zweite `SSLVerifyClient`-Direktive innerhalb einer `Directory`-Anweisung für dieses Unterverzeichnis auf `require` gesetzt. Der Ablauf ist dann folgender: Der Client kontaktiert den Server, und es wird zunächst eine Server-Authentifizierung durchgeführt. Danach hat der Client Zugriff auf das Document-Root-Verzeichnis. Versucht dann der Client, von den nicht geschützten Seiten auf die Seiten in dem durch `SSLVerifyClient require` geschützten Verzeichnis zuzugreifen, werden die Bedingungen für die SSL-Verbindung erneut ausgehandelt (*Renegotiation*). Der Server fordert dann vom Client ein Zertifikat und gestattet den Zugriff erst nach erfolgreicher Authentisierung des Clients.

SSLVerifyDepth

Zusammen mit der Direktive `SSLVerifyClient` wird die Direktive `SSLVerifyDepth` *n* eingesetzt. Durch die Zahl *n* wird die maximale Länge der zu prüfenden Zertifikatkette festgelegt. Ein Wert von *n*=`1` bedeutet, daß der Client direkt von einer Root-CA signiert wurde. Ein Wert von 3 erlaubt die Prüfung einer Zertifikatkette aus drei CA-Zertifikaten und Client-Zertifikat.

SSLRequire, SSLOptions, SSLRequireSSL

Wie das Client-Zertifikat auszusehen hat, kann sehr fein durch die `SSLRequire`-Direktive kontrolliert werden. Der SSL-Server setzt, wenn die Direktive `SSLOptions` die Option `+ExportCertData` enthält, mehrere Umgebungsvariablen, deren Werte aus den einzelnen Zertifikatfeldern bestehen. Beispielweise kann die Variable `SSL_CLIENT_S_DN_L` den Wert `Kiel` haben, also den Inhalt des Locality-Feldes (*L*) aus dem Distinguished Name (*DN*) des Zertifikat-Subjects (*S*). Die Bezeichnungen der anderen Variablen bauen sich analog auf. Ein vollständige Liste der Namen der gesetzten Variablen kann der Mod-SSL-Dokumentation entnommen werden.

Das oben bei der `SSLVerifyClient`-Direktive angeführte Beispiel des durch Renegotiation geschützten Unterverzeichnisses kann durch Verwendung dieser Variablen weiter verfeinert werden: z.B. sollen nur Clients mit überprüfbarem Zertifikat Zugriff auf das Unterverzeichnis bekommen, wenn das Zertifikat im Locality-Feld des DN die Zeichenkette `Kiel` enthält. Die entsprechende Direktive zur Prüfung des Locality-Feldes sieht so aus:

```
SSLOptions  +ExportCertData +OptRenegotiate +StrictRequire
SSLRequire  %{SSL_CLIENT_S_DN_L} eq "Kiel"
```

Die Option `+StrictRequire` zu setzen, hat eine Sicherungsfunktion. Wenn unter den nicht-SSL-Direktiven des Apache die Direktive `Satisfy` auf `any` gesetzt ist, kann ein Client einen ungewollten Zugriff auf das Unterverzeichnis bekommen. Bei gleichzeitiger Verwendung von `SSLRequire` und `SSLRequireSSL` erhält ein Client Zugang, sobald eine der Direktiven den Zugang zuläßt, die andere Direktive aber nicht. Durch Setzen von `+StrictRequire` wird der Zugriff verweigert, auch wenn eine der beiden Direktiven es zulassen würde. Die `Satisfy any`-Direktive ist somit in diesem Fall unwirksam.

15.4 Widerrufslisten (CRLs)

In diesem Abschnitt beschriebene Mod-SSL-Direktiven:

- `SSLCARevocationPath`
- `SSLCARevocationFile`

Das `mod_ssl`-Paket unterstützt die Verwendung von Widerrufslisten (CRL) mit dem Apache. Eine CRL enthält die Seriennummern zurückgerufener Zertifikate und den Namen der CA, die die CRL herausgegeben hat. Wird eine der folgenden Direktiven benutzt, werden Client-Zertifikate bei aktivierter Client-Authentisierung anhand einer CRL geprüft. Ist ein Client-Zertifikat widerrufen worden, wird der Zugriff dieses Clients abgelehnt. CRLs werden in regelmäßigen Abständen (z.B. wöchentlich oder monatlich) von den CAs herausgegeben. CRLs sind keine differentiellen Listen, sondern enthalten immer alle von einer CA widerrufenen Zertifikate. Die alte CRL einer CA kann also durch die neu herausgegebene CRL ersetzt werden. Trotzdem kann es sinnvoll sein, alte CRLs aufzubewahren (s.u.). Für jede unter den `SSLCACertificate`...-Direktiven (s.o.) angegebene CA sollte die entsprechende CRL dem Server zur Verfügung stehen.

SSLCARevocationPath, **SSLCARevocationFile**

Die CRLs werden dem Apache über eine der folgenden Direktiven (analog den `SSLCACertificate`-Direktiven) zur Verfügung gestellt, z.B.:

```
SSLCARevocationFile /usr/local/apache/ssl/crl.crl
```

Die Direktive weist auf eine Datei mit der CRL. Liegen CRLs von verschiedenen CAs vor, können alle CRLs in *eine Datei* geschrieben werden: z.B. `cat ca1.crl ca2.crl ca3.crl > crl.crl`.

`SSLCARevocationPath /usr/local/apache/ssl/crls`

Die Direktive hat dieselbe Funktion wie `SSLCARevocationFile`. Die Unterschied liegt in der Verwaltung der CRLs. Für `SSLCARevocationPath` müssen alle Zertifikate *einzeln* in das benannte Verzeichnis kopiert werden (hier: `.../ssl/crls`). Anschließend müssen die Listen dann über ihre Hash-Werte verlinkt werden. Bei mehreren Listen läßt sich das am einfachsten durch das unten stehende Shell-Skript erledigen.

CAs geben üblicherweise CRLs heraus, die immer sämtliche aktuell widerrufenen Zertifikate enthalten. Daher kann die alte CRL einer CA gelöscht (oder besser für Archivierungszwecke umbenannt) werden. Das folgende Shell-Skript löscht zunächst den Hash-Link auf die alte CRL und benennt dann die alte CRL um, indem das aktuelle Datum angehängt wird. Anschließend wird die neue CRL verschoben und verlinkt. Im Beispiel wird angenommen, daß die neue CRL unter `/tmp/bsp.crl` vorliegt und im Verzeichnis `/usr/local/apache/ssl/crls` installiert werden soll.

```
#!/bin/sh
cd /usr/local/apache/ssl/crls
rm `openssl crl -in /tmp/bsp.crl -hash -noout`.r0
mv bsp.crl bsp.crl.`date '+%m%d%y'`
mv /tmp/bsp.crl .
ln -s $i `openssl crl -in bsp.crl -hash -noout`.r0
chmod 644 bsp.crl.*
```

Die Verwaltung der CRLs ist mit einem gewissen Aufwand verbunden, da CRLs von CAs u.U. monatlich aktualisiert werden und entsprechend die Listen auf dem SSL-Server zu warten sind. Möglicherweise läßt sich der Vorgang durch ein entsprechendes Skript per Cron-Job automatisieren. Sollte die CRL nicht (wie im Skript angenommen) im PEM-Format vorliegen, sondern im DER-Format, muß in dem Skript in der zweiten Zeile (nach `openssl crl`) noch die Option `-inform der` eingefügt werden.

15.5 Starten und Stoppen des Apache

Der Server kann über das Skript `apachectl` oder durch Eingabe eines der folgenden Kommandos gestartet bzw. gestoppt werden:

- `/usr/local/bin/httpd`
  In der Beispiel-Konfigurationsdatei im Anhang G sind die SSL-bezogenen Teile der Apache-Konfiguration über `<IfDefine SSL>...</IfDefine SSL>` gekapselt. Somit startet dieses Kommando den Apache *ohne* SSL-Funktionalität.

- `/usr/local/bin/httpd -DSSL`
  Mit diesem Kommando wird der Apache mit zusätzlicher *mit* SSL-Funktionalität gestartet.
- `` kill -TERM `cat /usr/local/apache/logs/httpd.pid` ``
  Mit diesem Kommando wird der Server gestoppt.

Nach Eingabe des zweiten Kommandos wird der Schlüssel für den SSL-Server geladen. Liegt dieser verschlüsselt vor, wird nach einer Passphrase gefragt. Nach Eingabe der Passphrase startet der Server dann. Das impliziert, daß beim Starten des SSL-Servers jemand zugegen sein muß. Alternativ kann die Passphrase in einer – wie auch immer beschaffenen – Datei abgelegt werden. Wichtig ist nur, daß die Passphrase von dieser Datei bzw. dem Programm auf der Standardausgabe ausgegeben wird. Dazu muß die SSL-Direktive `SSLPassPhraseDialog` auf den Pfad zu dieser Datei gesetzt werden:

```
SSLPassPhraseDialog /usr/local/bin/gibpassphrase
```

Voreinstellung der Direktive ist `builtin`, was für die Eingabe der Passphrase am Prompt steht. Bei einem unbeaufsichtigten Start des Servers (z.B. nach einem Reboot) kann mit dem obigen ersten Kommando (per init-Skript) bei verschlüsselter Passphrase zumindest der Nicht-SSL-Server wieder in Betrieb genommen werden. Oder es kann alternativ die Passphrase-Datei eingesetzt werden.

Das Skript `apachectl` nutzt letztlich die obigen drei Kommandos. Die Funktion der obigen Kommandos würde mit dem Skript durch folgende Aufrufe erreicht:

- `/usr/local/bin/apachectl`
- `/usr/local/bin/apachectl startssl`
- `/usr/local/bin/apachectl stop`

Das Skript bietet noch eine Reihe weiterer Funktionen, allerdings z.T. in Abhängigkeit von der Server-Konfiguration. Für weitere Informationen wird auf das kommentierte Skript verwiesen.

Anhang

Anhang A

Empfehlenswerte Lektüre

A.1 Online-Dokumente

OpenSSL- und SSL-Apache-Handbuch der DFN-PCA

regelmäßig aktualisierte Versionen von Teil II und III dieses Handbuches

`http://www.pca.dfn.de/dfnpca/certify/ssl/handbuch/`

zu digitalen Signaturen, Public-Key-Verschlüsselung und ihren kryptographischen Grundlagen

Online-Fassung der Diplomarbeit *Digitale Signaturen* von ROBERT GEHRING [Geh98]

`http://ig.cs.tu-berlin.de/ap/rg/002/`

zur Krypto-Kontroverse

ULF MÖLLERS Webseite

`http://www.fitug.de/ulf/krypto/verbot.html`

zu Kryptographie und Recht weltweit

B-J. KOOPS Crypto Law Survey

`http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm`

zu Datenschutz allgemein

WWW-Angebot des Berliner Datenschutzbeauftragten

`http://www.datenschutz.de`

zur Exportkontrolle

Homepage des Wassenaar Sekretariats

`http://www.wassenaar.org`

zu ASN.1

BURTON KALISKIS *Layman's Guide to a Subset of ASN.1, BER, and DER* [Kal93]

`ftp://ftp.rsasecurity.com/pub/pkcs/ps/layman.ps`

OSS Nokalvas ASN.1 Ressourcen-Seite

`http://www.oss.com/asn1/index.html`

zu X.509 und ASN.1

Peter Gutmanns *X.509 Style Guide*

`http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt`

zur IT-Grundsicherung

BSI Grundschutzhandbuch-Homepage

`http://www.bsi.bund.de/gshb/_start.htm`

„Maximalforderungen“ beim Betrieb einer Zertifizierungsstelle

Maßnahmenkatalog für digitale Signaturen von RegTP und BSI (als Anregung, was unternommen werden kann bzw. müßte)

`http://www.bsi.bund.de/aufgaben/projekte/pbdigsig/download/kat.pdf`

zu TEMPEST (elektromagnetische Abstrahlung)

JOEL MCNAMARAS “Complete, Unofficial TEMPEST Information Page”

`http://www.eskimo.com/~joelm/tempest.html`

zu Angriffen auf die PGP-Verschlüsselung

BILL UNRUHS “PGP Attacks”

`http://axion.physics.ubc.ca/pgp-attack.html`

JOEL MCNAMARAS “Practical Attacks on PGP”

`http://www.eskimo.com/~joelm/pgpatk.html`

zum geplanten europäischen Überwachungsprogramm ENFOPOL [SH98b]

Dokumentation ENFOPOL-Papiere des Netzmagazin *TELEPOLIS*[1] (Heise Verlag)

`http://www.heise.de/tp/deutsch/special/enfo/default.html`

A.2 Linksammlungen

zu PGP-Versionen/-Integration/-Plugins/-Dokumentation

PGP international Homepage

`http://www.pgpi.com`

zu Kryptographie

ULF MÖLLERS Krypto-Seite

`http://www.fitug.de/ulf/krypto/`

zur Krypto-Kontroverse

Webseite der Projektgruppe verfassungsverträgliche Technikgestaltung e.V. (provet)

`http://www.provet.org/kk/kkindex.htm`

[1] `http://www.heise.de/tp/`

Gesetzestexte

Juristisches Internetprojekt der Universität des Saarlandes, Institut für Rechtsinformatik

`http://www.jura.uni-sb.de`

zum Telekommunikationsgesetz

NICOLAS REICHELTS Webseite

`http://www.crypto.de/tkg`

zu CAs, PKI und verwandten Gebieten

Die “Comprehensive List of Public-Key Infrastructure Links” der DFN-PCA

`http://www.pca.dfn.de/dfnpca/pki-links.html`

MARCH BRANCHAUDS PKI References Page

`http://www.xcert.com/~marcnarc/PKI/`

zu Zufallszahlen-Erzeugung

DAVID WAGNERS “online resources for collecting randomness”

`http://www.cs.berkeley.edu/~daw/rnd/index.html`

zu Computersicherheit allgemein

Uni Siegen Security-Server

`http://www.uni-siegen.de/security/`

zu Linux auf tragbaren Computern

KENNETH E. HARKERS “Linux on Laptops”-Webseite

`http://www.cs.utexas.edu/users/kharker/linux-laptop/mirrors.html`

A.3 Mailinglisten

win-pca Die PCA-Diskussions- und -Mitteilungsliste des DFN (unmoderiert)

`win-pca-request@pca.dfn.de`

ietf-pkix Die Diskussionsliste der Internet Engineering Task Force (IETF), in der die Public-Key Infrastructure on X.509 basis (PKIX) diskutiert und entwickelt wird (englisch)

`ietf-pkix-request@imc.org`

cert-talk Die Diskussionsliste für alle Fragen rund um X.509-Zertifikate und -Zertifizierung (englisch)

`majordomo@mail.structuredarts.com`

A.4 Bücher, Aufsätze

zur Geschichte der (herkömmlichen, symmetrischen) Kryptographie

DAVID KAHN: *The Codebreakers. The Comprehensive History of Secret Communication from Ancient Times to the Internet.* 2. Aufl., Dezember 1996. Scribner Book Company; ISBN 0-6848-3130-9.

zur US-National Security Agency (NSA)

über den US-Geheimdienst, der für die Auslandsüberwachung zuständig und zugleich der weltgrößte Arbeitgeber für Kryptologen ist

JAMES BAMFORD: *The Puzzle Palace. A Report on America's Most Secret Agency.* Penguin Books, New York, 1983. ISBN 0-1400-6748-5.

zur Frage der Haftung von Zertifizierungsstellen

BIRTE TIMM: *Signaturgesetz und Haftungsrecht*, Datenschutz und Datensicherheit (DuD) 9/97, S. 525 ff.

Anhang B

Bezugsquellen

Die Hinweise zu Hardware sind nicht als Empfehlung zu verstehen, sondern nur beispielhaft gemeint. Andere Hersteller haben möglicherweise vergleichbare oder ähnliche Geräte in ihrem Sortiment.

B.1 Software

PGP2.6.3in: `ftp://ftp.individual.net/doc/IN/IN-CA/pgp/pgp263in/`

GnuPG: `http://www.gnupg.org`
`ftp://ftp.gnupg.org/pub/gcrypt/`
`ftp://ftp.guug.de/pub/gcrypt/` (Mirror)

OpenSSL: `http://www.openssl.org`
`ftp://ftp.cert.dfn.de/pub/tools/crypt/openssl/source/`
(Mirror)

pkcs12 (obsolet), pfx:
`http://www.drh-consultancy.demon.co.uk/pkcs12faq.html`

ca-fix (obsolet):
`http://www.drh-consultancy.demon.co.uk/ca-fix.html`

Apache-SSL:
`http://www.apache-ssl.org`
`ftp://ftp.pca.dfn.de/pub/tools/net/sslapache/` (Mirror)

mod_ssl: `http://www.modssl.org`
`ftp://ftp.pca.dfn.de/pub/tools/net/mod_ssl/` (Mirror)

Fortify: FARRELL MCKAYS starke Verschlüsselung für Netscape-Browser

`http://www.fortify.net`
`ftp://ftp.pca.dfn.de/pub/tools/net/Fortify/` (Mirror)

Opera: Shareware-Browser mit starker Verschlüsselung

`http://www.operasoftware.com/download.html`

Netscape Certificate Server Extras:
`http://developer.netscape.com/tech/security/certs/certs.html`

MS CertMgr:
`http://www.microsoft.com/security/tech/certificates/formats.asp`
`http://msdn.microsoft.com/downloads/tools/authcodeie4/authcodeie4.asp?`
`http://www.microsoft.com/security/downloads/certinst.exe`

dumpasn1: `http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c`
`http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.cfg`

Secude: Ein von der GMD entwickeltes Krypto-Toolkit

`http://www.darmstadt.gmd.de/secude/`

Tripwire: `ftp://ftp.cert.dfn.de/pub/tools/admin/Tripwire/Tripwire-1.2.tar.gz`

CryptoEx Security Suite:
Glück & Kanja GmbH
Christian-Pleß-Str. 11–13
D-63069 Offenbach
Telefon: 0 69 / 80 07 06 - 10
Telefax: 0 69 / 80 07 06 - 66
`info@glueckkanja.de`
`http://www.glueckkanja.de`

ascom MAIL / IDEA@exchange:
Ascom Systec AG
Gewerbepark
CH-5506 Mägenwil
Telefon: (+41) 62 / 889 - 52 11
Telefax: (+41) 62 / 889 - 59 90
`SEMS@ascom.ch`
`http://www.ascom.ch/systec/`

B.2 Notebooks

Panasonic Toughbook CF-71 robust, SuperDisk Drive LS-120 optional möglich
Panasonic Toughbook 45NJ48 robust, “all-in-one”
Panasonic Ruggby CF-25 spritzwasserbeständig, LS-120 optional möglich

Panasonic Deutschland GmbH
Winsbergring 15
D-22525 Hamburg
Telefon: 040 / 85 49 - 27 76
Telefax: 040 / 85 49 - 25 00
`http://www.panasonic.de`

Rocky II – „das Outdoor Notebook“ schlagfest, internes CD-ROM und Floppy-LW

roda computer GmbH
Landstr. 6
D-77839 Lichtenau
Telefon: 0 72 27 / 95 79 - 0
Telefax: 0 72 27 / 95 79 - 20
E-Mail: `mail@roda-computer.com`
`http://www.roda-computer.com`

Apple G3 PowerBooks schnell[1], SCSI-Anschluß integriert[2], gleichzeitiger Betrieb von internem CD-ROM und Floppy möglich

Apple Response Center
Dornacher Str. 3 d
D-85622 Feldkirchen
Telefon: 0 18 05 / 00 06 22
Telefax: 0 18 05 / 00 06 23
E-Mail: `de.response@euro.apple.com`
`http://www.apple.com/de/powerbook/`

B.3 Spezial-Hardware

Krypto-Beschleuniger

“nFast”-Crypto-Accelerator-Familie, Testberichte u.a. in [Edw98, Mac97]

nCipher Corporation Ltd.
Andrew Newton
Mühlenkoppel 27
D-22889 Tangstedt
Telefon: 0 41 09 / 25 08 51
Telefax: 0 41 09 / 25 08 53
`andrew@ncipher.com`
`http://www.ncipher.com/international/german.html`

Biometrie

„Biomouse“/„Biomouse PLUS“ Fingerprintscanner[3] – einer der wenigen Scanner, die auch mit Treibern bzw. einem Developer Toolkit für Unix/Linux erhältlich sind.

[1] Geschwindigkeitsvergleich zwischen Apple PowerBooks und Notebook-PCs siehe [MR98]

[2] bei den neuesten PowerBooks nicht mehr

[3] Vorteil von Fingerabdruck-Scan (und Iris-Scan) gegenüber Stimm-, Gesichts- oder Unterschriftenerkennung: Sie weisen eine um den Faktor 10^3–10^4 geringere *false acceptance rate* auf, bei ihnen liegt also der Anteil der fälschlich akzeptierten Personen um Größenordnungen unter denen anderer biometrischer Verfahren [Pet97, S. 52]. Nachteil z.B. gegenüber Handschriften-basierten Systemen: Man kann maximal zehn Mal den „Schlüssel“ (das Merkmal) wechseln...

Preis: ca. 600 DM ohne bzw. 700 DM mit integriertem Kartenleser (Besprechung in [Lan98])

Importeur:

AlphaNet Online GmbH
Konrad-Adenauer-Ufer 37
D-50668 Köln
Telefon: 02 21 / 37 00 - 0
Telefax: 02 21 / 37 00 - 370
E-Mail: `security@alphanet.de`
`http://www.alphanet.de/biomouse.htm`

Hardware-Zufallszahlengenerator

Protego SG100 Security Generator
Kleiner Zufallszahlengenerator für serielle Schnittstelle (Abb. B.1 und B.2), würde wegen der kleinen Maße besonders gut zu einem Laptop als CA-Rechner passen und wäre, da er ohne eigene Stromversorgung auskommt, ebenso mobil. Der in Aussicht gestellte Linux-Treiber für den SG 100 ist allerdings vorerst nicht erhältlich. (*"The Linux driver project has been discontinued. We will restart this project only if a high-volume customer requests a Linux driver."*) (US$140, Mengenrabatt ab 10 Stück)

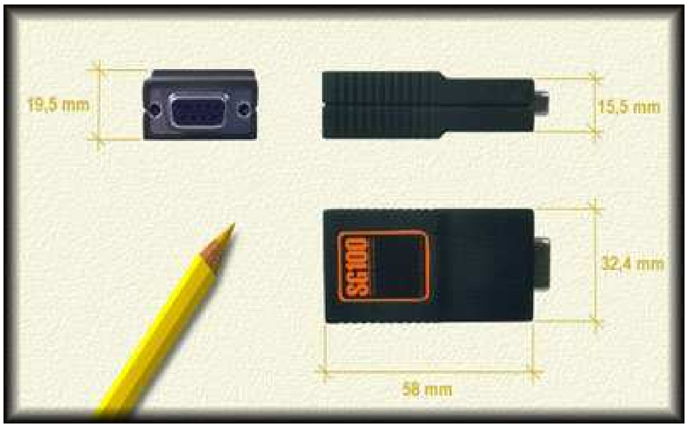


Abb. B.1: Protego SG100 Security Generator (Aufmaß)

Protego Information AB
Gamma-Paviljongen IDEON
Sölvegatan 41
SE - 223 70 Lund
SCHWEDEN
Telefon: +46 (0)46 286 36 30
Telefax: +46 (0)46 286 36 40
`sales@protego.se`
`http://www.protego.se/sg100_en.htm`

Siehe auch [SU97] mit weiteren Anbietern von HW-Zufallszahlengeneratoren.

Abb. B.2: Protego SG100 Security Generator im Einsatz

TEMPEST-Hardware

TEMPEST-PCs und Zubehör

Siemens AG
Bereich Automatisierungstechnik
Kombinationstechnik
AUT 7 B3
Postfach 23 55
D-90713 Fürth
Telefon: 0911 / 750 - 93 68
Telefax: 0911 / 750 - 98 98

TEMPEST-Abschirmgerät „SecuDat“ / „Data Safety Device“

unterbindet/überlagert weitgehend die elektromagnetische Abstrahlung [Kre99a] (ca. 2500 DM)

Teller Bau- und Entwicklungs GmbH
Hr. Wolff
Friedrichstr. 95
D-10117 Berlin
Telefon: 0 30 / 20 96 - 27 41
Telefax: 0 30 / 20 96 - 27 43

oder Kontaktaufnahme mit Herrn Wolff über ANDY MÜLLER-MAGUHN vom Chaos Computer Club `<andy@ccc.de>`

Anhang C

PGP-Beispielkonfiguration `config.txt`

Das nachfolgende Beispiel zeigt, wie eine Konfigurationsdatei “config.txt” beim Einsatz von PGP2.6.3in für die UNI-Zertifizierungsstelle aussehen könnte. Erläuterungen sind in die Kommentare (vom ’#’ bis zum jeweiligen Zeilenende) eingefügt.

```
#
# Beispiel-Konfigurationsdatei "config.txt" fuer PGP 2.6.3in
# fuer die UNI-CA
#
# Blank lines are ignored, as is anything following a '#'.
# Keywords are not case-sensitive.
# Whatever appears in here can be overridden on the command line,
# by specifying (for example) "+armor=on".

# The language we will be using for displaying messages to the user.

Language = en  # 'Language = de', falls deutsche PGP-Meldungen
               # bevorzugt werden (entsprechende Sprach-Meldungsdatei
               # "language.txt" mit Eintraegen fuer die gewuenschte
               # Sprache muss installiert sein; anderenfalls wird die
               # Meldung auf Englisch ausgegeben)

# Character set for displaying messages and for conversion of text files.
# If you set this variable to anything else than latin1, koi8 or noconv,
# PGP will do character set conversions if TextMode = on or if you specify
# the -t option when encrypting or signing a file.
#
# Available character sets:
#
#   latin1    - ISO-Latin/1      (ISO 8859/1)
#   cp850     - IBM codepage 850 (International)
#   cp852     - IBM codepage 852 (Eastern Europe)
#   cp860     - IBM codepage 860 (Portuguese)
#   cp866     - IBM codepage 866 (Russian)
#   alt_codes - Cyrillic         (Russian)
#   koi8      - Cyrillic         (Russian)
#   keybcs2   - KEYBCS 2         (Eastern Europe)
#   next      - NeXTSTEP
#   mac       - Macintosh
#   ascii     - 7-bit ASCII
```

```
CharSet = latin1    # z.B. fuer typische Linux-Installationen

# TMP is the directory name for PGP scratch files, usually a RAM disk.
# It can be overridden by the environment variable TMP.

TMP = "/tmp"
                   # sollte auf eine ggf. einzurichtende RAM-Disk zeigen
                   # alternativ auf ein Verzeichnis auf dem Wechselmedium,
                   # auf dem der geheime Schluessel gespeichert wird
                   # (damit keine sensiblen temporaeren Dateien -- auch
                   # nicht voruebergehend -- auf der *Festplatte* des
                   # CA-Rechners gespeichert werden)

# Pager is the file viewing program used for viewing messages with -m
# If not set or set to "pgp", a built-in pager will be used.  The pager set
# in config.txt will override the environment variable PAGER.

Pager = "more"     # ...beispielsweise, falls ein externer Pager bevorzugt wird

# ArmorLines is the maximum number of lines per packet when creating a
# transport armored file.  Set to 0 to disable splitting in parts.

ArmorLines = 0     # sollte auf '0' gesetzt sein, damit nicht lange
                   # Ausgabe-Dateien von PGP in viele kleine "Haeppchen"
                   # zerlegt werden (diese Option ist historisch bedingt
                   # und diente dazu, grosse PGP-Nachrichten per Mail ueber
                   # Systeme senden zu koennen, die ein Groessenlimit fuer
                   # einzelne E-Mails hatten)


Armor = on         # Use -a flag for ASCII armor whenever applicable

                   # keine Binaer-Ausgaben erzeugen, sondern ASCII-"armor"
                   # (Format aehnlich dem BASE64-Encoding von MIME oder
                   # dem Output, den UUENCODE erzeugt)


TextMode = on      # Attempt to use -t option where applicable


Clearsig = on      # Use ASCII armor even for unencrypted signed messages

                   # Texte so signieren, dass sie auch fuer Empfaenger ohne
                   # PGP lesbar bleiben; es wird dann nur eine Signatur
                   # angehaengt - der zu signierende Text selber wird quasi
                   # im Klartext lesbar "eingebettet" in den resultierenden Text


#Verbose = 2       # Verbose diagnostic messages

                   # aktivieren, falls gewuenscht, wird aber
                   # etwas unuebersichtlich

# Verbose = 0      # Be quiet

                   # Fuer interaktives Arbeiten nicht empfehlenswert;
                   # kann eventuell beim Einsatz von PGP in Skripten
                   # sinnvoll sein (und sollte dann mit dem Kommando-
                   # zeilen-Switch '+batchmode' kombiniert werden)


# Compress = off   # Suppress compression to aid debugging
```

```
                   # sollte ausgeschaltet bleiben, da die Kompression
                   # vor dem Verschluesseln Redundanz aus dem Klartext
                   # entfernt und so einem Angreifer die Arbeit erschwert


# ShowPass = on    # Echo password when user types it

                   # SOLLTE *NIE* AKTIVIERT SEIN -- anderenfalls waere z.B.
                   # die Kombination zweier Teil-Passsaetze zwecks Umsetzung
                   # des Vier-Augen-Prinzips nach dem ersten Aufruf obsolet,
                   # da ja dann beide beteiligten den Teil-Passsatz des jeweils
                   # anderen lesen koennten


Interactive = on   # Interactively prompt the user when adding keys (-ka)

                   # empfehlenswert, damit Schluessel nicht automatisch in den
                   # Public-Keyring uebernommen werden, sondern nur auf
                   # Nachfrage und ausdrueckliche Bestaetigung


EncryptToSelf = on # Encrypt all messages with your own public key


# MyName is substring of default userid for secret key to make signatures.
# If not set, PGP will use the first key on your secret keyring (the last
# key you created) if you don't specify the user with -u

MyName = "UNI-CA SoSe 1999 Certification Key"

                   # eindeut. Teilstring der UserID des CA-SIGN-Keys,
                   # alternativ die hexadezimale KeyID dieses Schluessels


# MyEtsId is substring of default userid for public key to encrypt to self
# If not set, PGP will use the first key on your secret keyring (the last
# key you created) if you don't specify the user with -u

MyEtsId = "Communication Key <ca@UNI.de>"

                   # eindeutiger Teilstring der UserID
                   # des CA-KOMMUNIKATIONSschluessels, alternativ die
                   # hexadezimale KeyID dieses Schluessels


# AutoSign = off   # Don't sign new userids

                   # sollte deaktiviert bleiben, damit neu-generierte eigene
                   # Schluessel sofort nach dem Generieren selbst-signiert
                   # und damit vor unbemerkten Manipulationen geschuetzt werden


Unknown_Certs = off
                   # unterdrueckt die Anzeige unbekannter Zertifikate
                   # macht die Ausgaben von PGP uebersichtlicher, zumal
                   # bei User- oder CA-Schluesseln, die von vielen unbekannten
                   # Dritten signiert sind
                   # andernfalls hat man -zig Meldungen der Form
                   #
                   # sig     <KeyID>      (Unknown signator, can't be checked)
                   #
                   # zwischen den wichtigen Ausgaben von PGP (bei 'pgp -kv')
```

```
# Number of completely trusted signatures needed to make a key valid.

Completes_Needed = 1
                      # z.B. die DFN-PCA und die von ihr zertifizierten
                      # Schluessel anderer Zertifizierungsinstanzen


# Number of marginally trusted signatures needed to make a key valid.

Marginals_Needed = 9999
                         # in einer CA-Umgebung sollten "Introducer"
                         # (in diesem Falle ausschliesslich CAs)
                         # entweder vertrauenswuerdig sein oder nicht,
                         # aber nicht "ein bisschen". Daher duerfte
                         # es sinnvoll sein zu verhindern, dass
                         # mehrere "halb-vertrauenswuerdige"
                         # Zertifizierungen anderer Stellen einen
                         # Schluessel als gueltig erscheinen lassen koennten


# How many levels of introducers may introduce other introducers.

Cert_Depth = 2  # Eine Indirektionsstufe, d.h. die DFN-PCA
                         # koennte beispielsweise die oeffentlichen Schluessel
                         # anderer CAs, die sie zertifiziert hat, "introducen"
                         # Bei `Cert_Depth = 2' koennten diese anderen CAs
                         # selber wieder oeffentliche Schluessel Dritter
                         # "einfuehren", also beispielsweise die ihrer Nutzer
                         # oder einer anderen cross-zertfizierten CA (oder auch
                         # einer eigenen Sub-CA oder einer eigenen
                         # Registrierungsstelle)
                         # Eine Beschraenkung auf `Cert_Depth = 1' ist in jedem
                         # Fall die `konservativere', vorsichtigere Variante


# TZFix is hours to add to time() to get GMT, for GMT timestamps.
# Since MSDOS assumes local time is US Pacific time, and pre-corrects
# Pacific time to GMT, make TZFix=0 for California, -1 for Colorado,
# -2 for Chicago, -3 for NY, -8 for London, -9 for Amsterdam.
# However, if your MSDOS environmental variable TZ is properly defined
# for your timezone, you can leave TZFix=0.  Unix systems probably
# shouldn't need to worry about setting TZFix.

# TZFix = 0              # i.d.R. nur unter DOS/Windows relevant


# BakRing is the path to a backup copy of your secret keyring, usually
# on floppy disk.  Your secret keys will be compared with the backup copy
# when doing a keyring check (pgp -kc)

# BakRing = "a:\secring.pgp"

# Da der geheime Schluessel bereits auf einem externen Datentraeger
# gespeichert ist, ausserdem ein Backup des geheimen Signierschluessels
# nicht sinnvoll ist (vgl. entsprechende Ausfuehrungen im Hauptteil
# der Arbeit unter `Archivierung'), duerfte diese Option nicht gebraucht
# werden


# Paths to keyrings and seed file for random generator.

SecRing = "/mountpoint_keymedium/secring.pgp"
```

```
                         # ‘/mountpoint_keymedium’ sollte
                         # das Verzeichnis sein, als das
                         # das Wechselmedium mit den
                         # geheimen Schluesseln gemountet
                         # wurde


PubRing = "/mountpoint_keymedium/pubring.pgp"
                         # falls auch die Public-Keys auf dem
                         # Wechselmedium gespeichert werden
# PubRing = "/pfad_zum_PGP-Verzeichnis/pubring.pgp"
                         # falls die Public-Keys auf der
                         # Festplatte abgelegt werden
                         # ‘/pfad_zum_PGP-Verzeichnis’ muß
                         # der vollstaendige Pfad sein, oder
                         # aber "pubring.pgp" muß in dem
                         # Verzeichnis liegen, auf das die
                         # Umgebungsvariable PGPPATH verweist


# RandSeed = "randseed.bin"
                       # hier koennte eventuell "/dev/random" o.ae. eingetragen
                       # werden, falls der CA-Rechner mit einem kryptographisch
                       # starken (Hardware-)Zufallszahlengenerator ausgestattet
                       # ist
                       # Ansonsten empfiehlt es sich, auch diese Datei auf dem
                       # Wechselmedium mit den geheimen Schluesseln unterzu-
                       # bringen, was aber nur moeglich ist, wenn dieses
                       # schreibbar ist (randseed.bin wird des oefteren
                       # mit neuen Zufallszahlen aufgefuellt) -- hier duerfte
                       # es also zu einer Abwaegung zwischen dem Schutz des
                       # geheimen Schluessels und dem Zufallszahlenvorrat fuer
                       # Session-Keys kommen muessen, die bei einer CA im
                       # Zweifel zugunsten der secret keys ausfallen sollte


# Add optional comment to ASCII armor output.

Comment = "Zertifizierungsstelle UNI"
                               # Hier koennte IHRE Werbung stehen! ;-)
                               # der Kommentar sollte kuerzer als
                               # 66 Zeichen sein, da ihm noch ein
                               # ’Comment: ’ vorangestellt wird und
                               # es sonst oft Probleme mit zu langen
                               # Textzeilen gibt, die dann z.B. vom
                               # Mail-UA umgebrochen werde -- was
                               # beim Empfaenger zu Problemen bei der
                               # Verarbeitung der entsprechenden Mail
                               # mit PGP fuehrt
# An dieser Stelle laesst sich ein Hinweis auf die Homepage der
# Zertifizierungsstelle unterbringen, oder ein Aufruf, doch bitte oefter
# zu Verschluesselung zwecks Schutz der Privatsphaere zu greifen, oder
# der Appell, seine Schluessel von der UNI-CA zertifizieren zu lassen...
#
# Diese Kommentarzeile wird bei allen von PGP generierten Ausgaben mit
# eingefuegt, so beispielsweise in den Vorspann von extrahierten Schluesseln
# oder von Signaturen:
#
#     -----BEGIN PGP MESSAGE-----
#     Version: 2.6.3in
#     Comment: Zertifizierungsstelle der UNI
#
#     hQEMA+CY00q2yzDZAQgAxAZcglv36InvtHFVbdvmBXGRmzKPn8LBuX7vXWpReeBM
```

```
#      T9eekuhWEoUoTwXwpC9HAXH6nyzI/5URdNM0qUtJJf4w3OZM9vTbU1ric3UAVsi1
#      ytCVoQM8faeElu/8FvFonl9ezcgACBgcEYQoiA+NELmy3GJZpmr1O90fWkMwfOSP
#      [...]
#
# oder
#
#      -----BEGIN PGP SIGNATURE-----
#      Version: 2.6.3in
#      Charset: next
#      Comment: Use PGP for e-mails like envelopes for letters!
#
#      mQBtAzHIH3AAAAEDANcnGb6IZYKgkwyLFmiXMMvGPK+n2YSev0Bwg64x4/0XMuyo
#      [...]
#
#
# EOF
```

Die speziell für den Einsatz in Zertifizierungsumgebungen (weiter-)entwickelte Version PGP263i**n** bietet einige Optionen, die in PGP2.6.3 noch nicht vorgesehen sind. Auf die entsprechende zusätzliche Funktionalität müßte, wenn eine andere PGP2.6-Version als 2.6.3in eingesetzt wird, verzichtet bzw. diese müßten auf andere Weise, nötigenfalls manuell, nachgebildet werden: **MyEtsId** und **Unknown_Certs** existieren nur in der “in”-Version. Weitere Vorzüge dieser Version sind in [CD98, Cam98c] beschrieben.

Anhang D

Undokumentierte PGP-Optionen

Undokumentierte Features und Optionen von PGP 2.6.x (nach einer Liste von CHRISTIAN SCHNEIDER `<100542.2132compuserve.com>`, dokumentiert auf `http://www.uni-siegen.de/security/krypto/pgp-udf.html`):

-km ‘pgp -km’ oder auch (falls der Schlüsselbund sehr groß ist und man sich alles in Ruhe anschauen will) ‘pgp -km > output.txt’ zeigt eine Kette von Signaturen aus dem Public-Key Ring an. Ausgehend vom “ultimately trusted”-Key (der eigene, zu dem sich der dazugehörige Private Key im Secret Key Ring befindet) werden alle Schlüssel angezeigt, die mit diesem Key signiert sind; dann werden alle Schlüssel angezeigt, die von einem dieser Keys signiert sind usw. – quasi eine Art „Lawine“ oder Kettenreaktion.

-L Dieser zusätzliche Parameter (kann auch kleingeschrieben werden) kann bei allen PGP-Optionen mit angegeben werden und bewirkt, daß PGP Debug-Informationen anzeigt. Statt ‘pgp -L...’ kann man aber auch ‘pgp +verbose=2’ eingeben, das hat dieselbe Wirkung.

-kg <keysize> <expsize>
Der zusätzliche Parameter <*expsize*> gibt bei der Erzeugung eines neuen Schlüsselpaares an, wie groß der sogenannte “public exponent” werden soll.

-da <ciphertext>
Diese Option wandelt eine Datei in PGP-ASCII-Armor-Codierung [ASZ96] (vergleichbar etwas dem MIME-BASE64-Encoding) in eine Datei mit dem entsprechenden Binär-Inhalt um (vergleichbar mit dem MIME-8bit-Encoding)

+makerandom=<number> <filename>
Dieser Parameter erzeugt eine Datei <*filename*>, die <*number*> Zufallsbytes enthält. So kann man sich z.B. eine *Challenge* erzeugen lassen oder einen Sitzungsschlüssel. Es wird dabei der interne Pseudo-Zufallszahlengenerator von PGP benutzt.

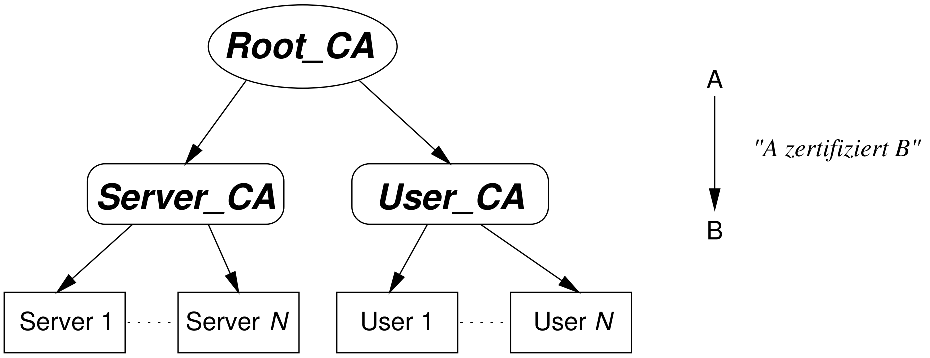
+encrypttoself
Dieser Parameter (der auch in der Konfigurationsdatei von PGP, “config.txt”, auftauchen kann – dann z.B. in der Form ‘encrypttoself = ON’) bewirkt, daß bei einer

Verschlüsselung die zu verschlüsselnde Nachricht nicht nur an den oder die angegebenen Empfänger sondern zusätzlich auch immer an den mit dem Eintrag "MYNAME" in der Konfigurationsdatei vorgegebenen Empfänger (beispielsweise einen selbst) verschlüsselt wird. Wenn man diese Option benutzt, braucht man sich beim Verschlüsseln nicht mehr explizit als Verschlüsselungsempfänger mit anzugeben und kann trotzdem die resultierende Mail (z.B. im Postausgangs-Ordner) entschlüsseln.

Anhang E

`openssl.cnf` – Beispiel-Datei

Die folgende Beispiel-Konfigurationsdatei enthält drei Abschnitte für verschiedene CA-Konfigurationen. Der erste Abschnitt `[ Root_CA ]` enthält eine Konfiguration zur Herausgabe von CA-Zertifikaten, entsprechend `[ Server_CA ]` zur Herausgabe von SSL-Server-Zertifikaten (*Server 1 bis N*) und `[ User_CA ]` für die Herausgabe von Benutzer-Zertifikaten (*User 1 bis N*). Damit kann die unten dargestellte Zertifizierungshierarchie realisiert werden.



Die einzelnen CA-Abschnitte unterscheiden sich vor allem in der Angabe zum Extension-Abschnitt, der beim Schlüsselwort `x509_extensions` im jeweiligen CA-Abschnitt festgelegt ist. Über den Extension-Abschnitt wird bestimmt, welche Extensions die herausgegebenen Zertifikate enthalten und somit ihr Verwendungszweck. Die CA-Abschnitte werden bei der der Zertifizierung von Requests durch das `ca`-Kommando mit der Option `-name` angewählt (siehe Abschnitt Signieren (10.3)).

Für die drei CA-Abschnitte gemeinsame Werte können auch am Anfang der Konfigurationsdatei vor dem ersten Abschnitt (hier `[ new_oids ]` festgelegt werden.

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# RANDFILE              = $ENV::HOME/.rnd
# oid_file              = $ENV::HOME/.oid
# oid_section           = new_oids
```

```
# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions            =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

pfad                    = /usr/local/etc/ssl

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

####################################################################

[ ca ]

default_ca      = Server_CA             # The default ca section

####################################################################


[ Root_CA ]         # Abschnitt fuer eine Root CA

dir             = $pfad/PCA               # Where everything is kept
certs           = $dir/certs              # Where the issued certs are kept
crl_dir         = $dir/crl                # Where the issued crl are kept
database        = $dir/index.txt          # database index file.
new_certs_dir   = $dir/newcerts           # default place for new certs.

certificate     = $dir/PCAcert.pem        # The CA certificate
serial          = $dir/serial             # The current serial number
crl             = $dir/crl.pem            # The current CRL
private_key     = $dir/private/PCAkey.pem # The private key
RANDFILE        = $dir/private/.rand      # private random number file

x509_extensions = PCA_ext                 # The extentions to add to the cert

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
#crl_extensions = crl_ext                 # Extensions to add to CRL

default_days    = 730                     # how long to certify for
default_crl_days= 30                      # how long before next CRL
default_md      = md5                     # which md to use.
preserve        = no                      # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_match

######################################################################

[ Server_CA ]         # Abschnitt fuer eine Server CA

dir             = $pfad/SCA               # Where everything is kept
certs           = $dir/certs              # Where the issued certs are kept
crl_dir         = $dir/crl                # Where the issued crl are kept
```

```
database        = $dir/index.txt          # database index file.
new_certs_dir   = $dir/newcerts           # default place for new certs.

certificate     = $dir/SCAcert.pem        # The CA certificate
serial          = $dir/serial             # The current serial number
crl             = $dir/crl.pem            # The current CRL
private_key     = $dir/private/SCAkey.pem # The private key
RANDFILE        = $dir/private/.rand      # private random number file

x509_extensions = SCA_ext                 # The extentions to add to the cert
#crl_extensions = crl_ext                 # Extensions to add to CRL
default_days    = 365                     # how long to certify for
default_crl_days= 30                      # how long before next CRL
default_md      = md5                     # which md to use.
preserve        = no                      # keep passed DN ordering

policy          = policy_anything

######################################################################

[ User_CA ]         # Abschnitt fuer eine User CA

dir             = $pfad/UCA               # Where everything is kept
certs           = $dir/certs              # Where the issued certs are kept
crl_dir         = $dir/crl                # Where the issued crl are kept
database        = $dir/index.txt          # database index file.
new_certs_dir   = $dir/newcerts           # default place for new certs.

certificate     = $dir/UCAcert.pem        # The CA certificate
serial          = $dir/serial             # The current serial number
crl             = $dir/crl.pem            # The current CRL
private_key     = $dir/private/UCAkey.pem # The private key
RANDFILE        = $dir/private/.rand      # private random number file

x509_extensions = UCA_ext                 # The extentions to add to the cert
#crl_extensions = crl_ext                 # Extensions to add to CRL
default_days    = 365                     # how long to certify for
default_crl_days= 30                      # how long before next CRL
default_md      = md5                     # which md to use.
preserve        = no                      # keep passed DN ordering

policy          = policy_anything

######################################################################


# For the CA policy
# Auch hier gilt:
# ... you must list all acceptable 'object' types.

[ policy_match ]

countryName            = match
stateOrProvinceName    = supplied
localityName           = optional
organizationName       = supplied
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
```

```
[ policy_anything ]

countryName            = match
stateOrProvinceName    = optional
localityName           = optional
organizationName       = optional
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional

####################################################################

[ req ]

default_bits        = 1024
default_keyfile     = privkey.pem
distinguished_name  = req_distinguished_name
attributes          = req_attributes
x509_extensions     = v3_ca     # The extentions to add to the self signed cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix   : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # The extensions to add to a certificate request

######################################################################

[ req_distinguished_name ]

countryName                     = Country Name (2 letter code)
countryName_default             = DE
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
#stateOrProvinceName_default    = Schleswig-Holstein

localityName                    = Locality Name (eg, city)
#localityName_default           = Kiel

0.organizationName              = Organization Name (eg, company)
#0.organizationName_default     = Universitaet Kiel

# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg, section)
#organizationalUnitName_default = Studis

commonName                      = Common Name (eg, YOUR name)
```

```
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 60

# SET-ex3                       = SET extension number 3

######################################################################

[ req_attributes ]

# Das Challenge Password dient dazu, sich bei Verlust des geheimen Schluessels
# gegenueber der Herausgeber-CA fuer einen Zertifikatwiderruf auszuweisen.
# Wird bei Erstellung der Zertifikat-Anforderung erfragt.

challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

unstructuredName                = An optional company name

######################################################################

[ PCA_ext ]

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.
basicConstraints         = critical, CA:TRUE

# Moeglich: digitalSignature, nonRepudiation, keyEncipherment,
#           dataEncipherment, keyAgreement, keyCertSign,
#           cRLSign, encipherOnly, decipherOnly
keyUsage                 = cRLSign, keyCertSign

# PKIX recommendations
subjectKeyIdentifier     = hash
authorityKeyIdentifier   = keyid,issuer:always

# Import the email address.
subjectAltName           = email:copy

# Copy subject details
issuerAltName            = issuer:copy

crlDistributionPoints    = URI:http://mystic.pca.dfn.de/PCA.crl

# Moeglich: client, server, email, objsign, reserved, sslCA, emailCA, objCA
nsCertType               = sslCA, emailCA, objCA

# Hier kann der den folgenden Url's gemeinsame Url-Stamm angeben werden.
nsBaseUrl                = https://mystic.pca.dfn.de/

# Die Seite mit der CA-Policy
nsCaPolicyUrl            = http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html

# Ein Text, der von Netscape-Browsern angezeigt wird
nsComment                = This certificate was issued by a PCA\n\
just for testing.

# Hier kann eine Online-Zertifikatspruefung stattfinden, indem auf die
# Url in der Form ../foo.cgi?aaaa zugegriffen wird. "aaaa" ist dabei
# die ASCII-kodierte Seriennummer des Zertifikats. Dann kann das Zertifikat
# per OpenSSL geprueft werden. Wird vermutlich nur in Serverzertifikaten und
```

```
# durch Netscape-Browser unterstuetzt
# Zurueckgegeben wird eine dezimale 0 oder 1
# nsRevocationUrl        = cgi/non-CA-rev.cgi?

# Nur gueltig in CA-Zertifikaten. Bedeutung nicht ganz klar.
# nsCaRevocationUrl      = cgi/CA-rev.cgi?

# Wird verwendet um einem Benutzer die Erneuerung seines Zertifikats zu
# erleichtern. Ueblicherweise steckt dahinter ein CGI-Script, auf das per
# HTTP GET in der Form ../foo.cgi?aaaa zugegriffen wird. "aaaa" ist wieder
# Seriennummer. Zurueckgegeben werden kann ein Antrags-Formular zur Erneuerung
# des Zertifikats.
# nsRenewalUrl           = cgi/check-renw.cgi?

####################################################################

[ SCA_ext ]

# basicConstraints       = critical, CA:FALSE
keyUsage                 = digitalSignature, keyEncipherment
subjectKeyIdentifier     = hash
authorityKeyIdentifier   = keyid,issuer:always
subjectAltName           = email:copy
issuerAltName            = issuer:copy
crlDistributionPoints    = URI:http://mystic.pca.dfn.de/SCA.crl
nsCertType               = server
nsBaseUrl                = https://mystic.pca.dfn.de/
nsCaPolicyUrl            = http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html
nsComment                = This certificate was issued by a Server CA
nsRevocationUrl          = cgi/non-CA-rev.cgi?
# nsRenewalUrl           = cgi/check-renw.cgi?

####################################################################

[ UCA_ext ]

# basicConstraints       = critical, CA:FALSE
keyUsage                 = digitalSignature, keyEncipherment, keyAgreement
subjectKeyIdentifier     = hash
authorityKeyIdentifier   = keyid,issuer:always
subjectAltName           = email:copy
issuerAltName            = issuer:copy
crlDistributionPoints    = URI:http://mystic.pca.dfn.de/UCA.crl
nsCertType               = client, email
nsBaseUrl                = https://mystic.pca.dfn.de/
nsCaPolicyUrl            = http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html
nsComment                = This certificate was issued by a User CA
nsRevocationUrl          = cgi/non-CA-rev.cgi?
# nsRenewalUrl           = cgi/check-renw.cgi?

####################################################################

[ v3_ca ]

basicConstraints         = critical, CA:TRUE
subjectKeyIdentifier     = hash
authorityKeyIdentifier   = keyid:always,issuer:always
keyUsage                 = cRLSign, keyCertSign
nsCertType               = sslCA, emailCA, objCA
subjectAltName           = email:copy
issuerAltName            = issuer:copy
crlDistributionPoints    = URI:http://mystic.pca.dfn.de/PCA.crl
nsBaseUrl                = https://mystic.pca.dfn.de/
```

```
nsCaPolicyUrl            = http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html
nsComment                = This certificate is a Root CA Certificate



# RAW DER hex encoding of an extension: beware experts only!
# 1.2.3.5=RAW:02:03
# You can even override a supported extension:
# basicConstraints       = critical, RAW:30:03:01:01:FF

######################################################################

[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.

issuerAltName            = issuer:copy
authorityKeyIdentifier   = keyid:always,issuer:always
```

Anhang F

Aufrufparameter und Optionen von `openssl`

openssl

```
Standard commands
asn1parse      ca             ciphers        crl            crl2pkcs7
dgst           dh             dhparam        dsa            dsaparam
enc            errstr         gendh          gendsa         genrsa
nseq           passwd         pkcs12         pkcs7          pkcs8
req            rsa            s_client       s_server       s_time
sess_id        smime          speed          spkac          verify
version        x509

Message Digest commands (see the 'dgst' command for more details)
md2            md5            mdc2           rmd160         sha
sha1

Cipher commands (see the 'enc' command for more details)
base64         bf             bf-cbc         bf-cfb         bf-ecb
bf-ofb         cast           cast-cbc       cast5-cbc      cast5-cfb
cast5-ecb      cast5-ofb      des            des-cbc        des-cfb
des-ecb        des-ede        des-ede-cbc    des-ede-cfb    des-ede-ofb
des-ede3       des-ede3-cbc   des-ede3-cfb   des-ede3-ofb   des-ofb
des3           desx           idea           idea-cbc       idea-cfb
idea-ecb       idea-ofb       rc2            rc2-40-cbc     rc2-64-cbc
rc2-cbc        rc2-cfb        rc2-ecb        rc2-ofb        rc4
rc4-40         rc5            rc5-cbc        rc5-cfb        rc5-ecb
rc5-ofb
```

asn1parse

```
asn1parse [options] <infile
where options are
 -inform arg   input format - one of DER TXT PEM
 -in arg       input file
 -out arg      output file
 -noout arg    don't produce any output
 -offset arg   offset into file
 -length arg   length of section in file
```

```
 -i            indent entries
 -oid file     file of extra oid definitions
 -strparse offset
               a series of these can be used to 'dig' into multiple
               ASN1 blob wrappings
 -out filename output DER encoding to file
```

ca

```
usage: ca args

 -verbose        - Talk alot while doing things
 -config file    - A config file
 -name arg       - The particular CA definition to use
 -gencrl         - Generate a new CRL
 -crldays days   - Days is when the next CRL is due
 -crlhours hours - Hours is when the next CRL is due
 -startdate YYMMDDHHMMSSZ  - certificate validity notBefore
 -enddate YYMMDDHHMMSSZ    - certificate validity notAfter (overrides -days)
 -days arg       - number of days to certify the certificate for
 -md arg         - md to use, one of md2, md5, sha or sha1
 -policy arg     - The CA 'policy' to support
 -keyfile arg    - PEM private key file
 -key arg        - key to decode the private key if it is encrypted
 -cert file      - The CA certificate
 -in file        - The input PEM encoded certificate request(s)
 -out file       - Where to put the output file(s)
 -outdir dir     - Where to put output certificates
 -infiles ....   - The last argument, requests to process
 -spkac file     - File contains DN and signed public key and challenge
 -ss_cert file   - File contains a self signed cert to sign
 -preserveDN     - Don't re-order the DN
 -batch          - Don't ask questions
 -msie_hack      - msie modifications to handle all those universal strings
 -revoke file    - Revoke a certificate (given in file)
 -extensions ..  - Extension section (override value in config file)
 -crlexts ..     - CRL extension section (override value in config file)
```

ciphers

Nach Aufruf von `ciphers` werden die von OpenSSL unterstützten Cipher-Suites angezeigt.

crl

```
usage: crl args

 -inform arg     - input format - default PEM (DER or PEM)
 -outform arg    - output format - default PEM
 -text           - print out a text format version
 -in arg         - input file - default stdin
 -out arg        - output file - default stdout
 -hash           - print hash value
 -issuer         - print issuer DN
 -lastupdate     - lastUpdate field
 -nextupdate     - nextUpdate field
```

```
 -noout          - no CRL output
 -CAfile  name   - verify CRL using certificates in file "name"
 -CApath  dir    - verify CRL using certificates in "dir"
```

crl2pkcs7

```
crl2pkcs7 [options] <infile >outfile
where options are
 -inform arg    input format - DER or PEM
 -outform arg   output format - DER or PEM
 -in arg        input file
 -out arg       output file
 -certfile arg  certificates file of chain to a trusted CA
                (can be used more than once)
 -nocrl         no crl to load, just certs from '-certfile'
```

dgst

```
options are
-c   to output the digest with separating colons
-d   to output debug info
-md5 to use the md5 message digest algorithm (default)
-md2 to use the md2 message digest algorithm
-sha1 to use the sha1 message digest algorithm
-sha to use the sha message digest algorithm
-mdc2 to use the mdc2 message digest algorithm
-ripemd160 to use the ripemd160 message digest algorithm
```

dh

```
dh [options] <infile >outfile
where options are
 -inform arg   input format - one of DER PEM
 -outform arg  output format - one of DER PEM
 -in arg       input file
 -out arg      output file
 -check        check the DH parameters
 -text         print a text form of the DH parameters
 -C            Output C code
 -noout        no output
```

dhparam

```
dhparam [options] [numbits]
where options are
 -inform arg   input format - one of DER PEM
 -outform arg  output format - one of DER PEM
 -in arg       input file
 -out arg      output file
 -check        check the DH parameters
 -text         print a text form of the DH parameters
```

```
 -C            Output C code
 -2            generate parameters using  2 as the generator value
 -5            generate parameters using  5 as the generator value
 numbits       number of bits in to generate (default 512)
 -rand file:file:...
               - load the file (or the files in the directory) into
               the random number generator
 -noout        no output
```

dsa

```
dsa [options] <infile >outfile
where options are
 -inform arg     input format - DER or PEM
 -outform arg    output format - DER or PEM
 -in arg         input file
 -passin arg     input file pass phrase
 -out arg        output file
 -passout arg    output file pass phrase
 -des            encrypt PEM output with cbc des
 -des3           encrypt PEM output with ede cbc des using 168 bit key
 -idea           encrypt PEM output with cbc idea
 -text           print the key in text
 -noout          don't print key out
 -modulus        print the DSA public value
```

dsaparam

```
dsaparam [options] [bits] <infile >outfile
where options are
 -inform arg   input format - DER or PEM
 -outform arg  output format - DER or PEM
 -in arg       input file
 -out arg      output file
 -text         print the key in text
 -C            Output C code
 -noout        no output
 -rand         files to use for random number input
 number        number of bits to use for generating private key
```

enc

```
options are
-in <file>     input file
-out <file>    output fileencrypt
-e             encrypt
-d             decrypt
-a/-base64     base64 encode/decode, depending on encryption flag
-k             key is the next argument
-kfile         key is the first line of the file argument
-K/-iv         key/iv in hex is the next argument
-[pP]          print the iv/key (then exit if -P)
-bufsize <n>   buffer size
Cipher Types
```

```
des     : 56 bit key DES encryption
des_ede :112 bit key ede DES encryption
des_ede3:168 bit key ede DES encryption
idea    :128 bit key IDEA encryption
rc2     :128 bit key RC2 encryption
bf      :128 bit key Blowfish encryption
 -rc4   :128 bit key RC4 encryption
 -des-ecb      -des-cbc      -des-cfb      -des-ofb      -des  (des-cbc)
 -des-ede      -des-ede-cbc  -des-ede-cfb  -des-ede-ofb  -desx -none
 -des-ede3     -des-ede3-cbc -des-ede3-cfb -des-ede3-ofb -des3 (des-ede3-cbc)
 -idea-ecb     -idea-cbc     -idea-cfb     -idea-ofb     -idea (idea-cbc)
 -rc2-ecb      -rc2-cbc      -rc2-cfb      -rc2-ofb      -rc2  (rc2-cbc)
 -bf-ecb       -bf-cbc       -bf-cfb       -bf-ofb       -bf   (bf-cbc)
 -cast5-ecb    -cast5-cbc    -cast5-cfb    -cast5-ofb    -cast (cast5-cbc)
 -rc5-ecb      -rc5-cbc      -rc5-cfb      -rc5-ofb      -rc5  (rc5-cbc)
```

errstr

```
usage: errstr [-stats] <errno> ...
```

gendh

```
usage: gendh [args] [numbits]
 -out file - output the key to 'file
 -2    use 2 as the generator value
 -5    use 5 as the generator value
 -rand file:file:...
          - load the file (or the files in the directory) into
            the random number generator
```

gendsa

```
usage: gendsa [args] dsaparam-file
 -out file - output the key to 'file'
 -des      - encrypt the generated key with DES in cbc mode
 -des3     - encrypt the generated key with DES in ede cbc mode (168 bit key)
 -idea     - encrypt the generated key with IDEA in cbc mode
 -rand file:file:...
          - load the file (or the files in the directory) into
            the random number generator
 dsaparam-file
          - a DSA parameter file as generated by the dsaparam command
```

genrsa

```
usage: genrsa [args] [numbits]
 -des            encrypt the generated key with DES in cbc mode
 -des3           encrypt the generated key with DES in ede cbc mode (168 bit key)
 -idea           encrypt the generated key with IDEA in cbc mode
 -out file       output the key to 'file
 -passout arg    output file pass phrase
 -f4             use F4 (0x10001) for the E value
```

```
 -3             use 3 for the E value
 -rand file:file:...
                load the file (or the files in the directory) into
                the random number generator
```

nseq

```
Netscape certificate sequence utility
Usage nseq [options]
where options are
-in file  input file
-out file output file
-toseq    output NS Sequence file
```

passwd

```
Usage: passwd [options] [passwords]
where options are
-crypt             standard Unix password algorithm (default)
-apr1              MD5-based password algorithm
-salt string       use provided salt
-in file           read passwords from file
-stdin             read passwords from stdin
-quiet             no warnings
-table             format output as table
-reverse           switch table columns
```

pkcs12

```
Usage: pkcs12 [options]
where options are
-export       output PKCS12 file
-chain        add certificate chain
-inkey file   private key if not infile
-certfile f   add all certs in f
-name "name"  use name as friendly name
-caname "nm"  use nm as CA friendly name (can be used more than once).
-in  infile   input filename
-out outfile  output filename
-noout        don't output anything, just verify.
-nomacver     don't verify MAC.
-nocerts      don't output certificates.
-clcerts      only output client certificates.
-cacerts      only output CA certificates.
-nokeys       don't output private keys.
-info         give info about PKCS#12 structure.
-des          encrypt private keys with DES
-des3         encrypt private keys with triple DES (default)
-idea         encrypt private keys with idea
-nodes        don't encrypt private keys
-noiter       don't use encryption iteration
-maciter      use MAC iteration
-twopass      separate MAC, encryption passwords
-descert      encrypt PKCS#12 certificates with triple DES (default RC2-40)
-certpbe alg  specify certificate PBE algorithm (default RC2-40)
-keypbe alg   specify private key PBE algorithm (default 3DES)
```

```
-keyex        set MS key exchange type
-keysig       set MS key signature type
-password p   set import/export password (NOT RECOMMENDED)
-passin p     input file pass phrase
-passout p    output file pass phrase
-rand file:file:...
              load the file (or the files in the directory) into
              the random number generator
```

pkcs7

```
pkcs7 [options] <infile >outfile
where options are
 -inform arg   input format - DER or PEM
 -outform arg  output format - DER or PEM
 -in arg       input file
 -out arg      output file
 -print_certs  print any certs or crl in the input
 -text         print full details of certificates
 -noout        don't output encoded data
```

pkcs8

```
Usage pkcs8 [options]
where options are
-in file        input file
-inform X       input format (DER or PEM)
-passin arg     input file pass phrase
-envpassin arg  environment variable containing input file pass phrase
-outform X      output format (DER or PEM)
-out file       output file
-passout arg    output file pass phrase
-topk8          output PKCS8 file
-nooct          use (nonstandard) no octet format
-embed          use (nonstandard) embedded DSA parameters format
-nsdb           use (nonstandard) DSA Netscape DB format
-noiter         use 1 as iteration count
-nocrypt        use or expect unencrypted private key
-v2 alg         use PKCS#5 v2.0 and cipher "alg"
-v1 obj         use PKCS#5 v1.5 and cipher "alg"
```

req

```
req [options] <infile >outfile
where options  are
 -inform arg    input format - DER or PEM
 -outform arg   output format - DER or PEM
 -in arg        input file
 -out arg       output file
 -text          text form of request
 -noout         do not output REQ
 -verify        verify signature on REQ
 -modulus       RSA modulus
 -nodes         don't encrypt the output key
```

```
 -key file      use the private key contained in file
 -keyform arg   key file format
 -keyout arg    file to send the key to
 -newkey rsa:bits generate a new RSA key of 'bits' in size
 -newkey dsa:file generate a new DSA key, parameters taken from CA in 'file'
 -[digest]      Digest to sign with (md5, sha1, md2, mdc2)
 -config file   request template file.
 -new           new request.
 -x509          output a x509 structure instead of a cert. req.
 -days          number of days a x509 generated by -x509 is valid for.
 -newhdr        output "NEW" in the header lines
 -asn1-kludge   Output the 'request' in a format that is wrong but some CA's
                have been reported as requiring
 -extensions .. specify certificate extension section (override value in config file)
 -reqexts ..    specify request extension section (override value in config file)
```

rsa

```
rsa [options] <infile >outfile
where options are
 -inform arg     input format - one of DER NET PEM
 -outform arg    output format - one of DER NET PEM
 -in arg         input file
 -passin arg     input file pass phrase
 -in arg         input file
 -out arg        output file
 -passout arg    output file pass phrase
 -des            encrypt PEM output with cbc des
 -des3           encrypt PEM output with ede cbc des using 168 bit key
 -idea           encrypt PEM output with cbc idea
 -text           print the key in text
 -noout          don't print key out
 -modulus        print the RSA key modulus
 -check          verify key consistency
 -pubin          expect a public key in input file
 -pubout         output a public key
```

s_client

```
usage: s_client args

 -host host     - use -connect instead
 -port port     - use -connect instead
 -connect host:port - who to connect to (default is localhost:4433)
 -verify arg   - turn on peer certificate verification
 -cert arg     - certificate file to use, PEM format assumed
 -key arg      - Private key file to use, PEM format assumed, in cert file if
                 not specified but cert file is.
 -CApath arg   - PEM format directory of CA's
 -CAfile arg   - PEM format file of CA's
 -reconnect    - Drop and re-make the connection with the same Session-ID
 -pause        - sleep(1) after each read(2) and write(2) system call
 -showcerts    - show all certificates in the chain
 -debug        - extra output
 -nbio_test    - more ssl protocol testing
 -state        - print the 'ssl' states
 -nbio         - Run with non-blocking IO
```

```
 -crlf         - convert LF from terminal into CRLF
 -quiet        - no s_client output
 -ssl2         - just use SSLv2
 -ssl3         - just use SSLv3
 -tls1         - just use TLSv1
 -no_tls1/-no_ssl3/-no_ssl2 - turn off that protocol
 -bugs         - Switch on all SSL implementation bug workarounds
 -cipher       - preferred cipher to use, use the 'openssl ciphers'
                 command to see what is available
```

s_server

```
usage: s_server [args ...]

 -accept arg   - port to accept on (default is 4433)
 -context arg  - set session ID context
 -verify arg   - turn on peer certificate verification
 -Verify arg   - turn on peer certificate verification, must have a cert.
 -cert arg     - certificate file to use, PEM format assumed
                 (default is server.pem)
 -key arg      - Private Key file to use, PEM format assumed, in cert file if
                 not specified (default is server.pem)
 -dcert arg    - second certificate file to use (usually for DSA)
 -dkey arg     - second private key file to use (usually for DSA)
 -dhparam arg  - DH parameter file to use, in cert file if not specified
                 or a default set of parameters is used
 -nbio         - Run with non-blocking IO
 -nbio_test    - test with the non-blocking test bio
 -crlf         - convert LF from terminal into CRLF
 -debug        - Print more output
 -state        - Print the SSL states
 -CApath arg   - PEM format directory of CA's
 -CAfile arg   - PEM format file of CA's
 -nocert       - Don't use any certificates (Anon-DH)
 -cipher arg   - play with 'openssl ciphers' to see what goes here
 -quiet        - No server output
 -no_tmp_rsa   - Do not generate a tmp RSA key
 -ssl2         - Just talk SSLv2
 -ssl3         - Just talk SSLv3
 -tls1         - Just talk TLSv1
 -no_ssl2      - Just disable SSLv2
 -no_ssl3      - Just disable SSLv3
 -no_tls1      - Just disable TLSv1
 -no_dhe       - Disable ephemeral DH
 -bugs         - Turn on SSL bug compatibility
 -www          - Respond to a 'GET /' with a status page
 -WWW          - Respond to a 'GET /<path> HTTP/1.0' with file ./<path>
```

s_time

```
usage: s_time <args>

-connect host:port - host:port to connect to (default is localhost:4433)
-nbio         - Run with non-blocking IO
-ssl2         - Just use SSLv2
-ssl3         - Just use SSLv3
-bugs         - Turn on SSL bug compatibility
```

```
-new          - Just time new connections
-reuse        - Just time connection reuse
-www page     - Retrieve 'page' from the site
-time arg     - max number of seconds to collect data, default 30
-verify arg   - turn on peer certificate verification, arg == depth
-cert arg     - certificate file to use, PEM format assumed
-key arg      - RSA file to use, PEM format assumed, key is in cert file
                file if not specified by this option
-CApath arg   - PEM format directory of CA's
-CAfile arg   - PEM format file of CA's
-cipher       - preferred cipher to use, play with 'openssl ciphers'
```

sess_id

```
usage: sess_id args

 -inform arg     - input format - default PEM (DER or PEM)
 -outform arg    - output format - default PEM
 -in arg         - input file - default stdin
 -out arg        - output file - default stdout
 -text           - print ssl session id details
 -cert           - output certificate
 -noout          - no CRL output
 -context arg    - set the session ID context
```

smime

```
Usage smime [options] cert.pem ...
where options are
-encrypt       encrypt message
-decrypt       decrypt encrypted message
-sign          sign message
-verify        verify signed message
-pk7out        output PKCS#7 structure
-des3          encrypt with triple DES
-des           encrypt with DES
-rc2-40        encrypt with RC2-40 (default)
-rc2-64        encrypt with RC2-64
-rc2-128       encrypt with RC2-128
-nointern      don't search certificates in message for signer
-nosigs        don't verify message signature
-noverify      don't verify signers certificate
-nocerts       don't include signers certificate when signing
-nodetach      use opaque signing
-noattr        don't include any signed attributes
-binary        don't translate message to text
-certfile file other certificates file
-signer file   signer certificate file
-recip  file   recipient certificate file for decryption
-in file       input file
-inkey file    input private key (if not signer or recipient)
-out file      output file
-to addr       to address
-from ad       from address
-subject s     subject
-text          include or delete text MIME headers
-CApath dir    trusted certificates directory
```

```
-CAfile file   trusted certificates file
-rand file:file:...
               load the file (or the files in the directory) into
               the random number generator
cert.pem       recipient certificate(s) for encryption
```

speed

```
bad value, pick one of
md2      mdc2   md5      hmac      sha1    rmd160
idea-cbc rc2-cbc  rc5-cbc  bf-cbc
des-cbc  des-ede3 rc4
rsa512   rsa1024  rsa2048  rsa4096

dsa512   dsa1024  dsa2048
idea     rc2      des      rsa    blowfish
```

spkac

```
spkac [options]
where options are
 -in arg        input file
 -out arg       output file
 -key arg       create SPKAC using private key
 -passin arg    input file pass phrase
 -challenge arg challenge string
 -spkac arg     alternative SPKAC name
 -noout         don't print SPKAC
 -pubkey        output public key
 -verify        verify SPKAC signature
```

verify

```
usage: verify [-verbose] [-CApath path] [-CAfile file] cert1 cert2 ...
recognized usages:
        sslclient       SSL client
        sslserver       SSL server
        nssslserver     Netscape SSL server
        smimesign       S/MIME signing
        smimeencrypt    S/MIME encryption
        crlsign         CRL signing
```

version

```
usage:version -[avbofp]
```

x509

```
usage: x509 args
 -inform arg     - input format - default PEM (one of DER, NET or PEM)
 -outform arg    - output format - default PEM (one of DER, NET or PEM)
 -keyform arg    - private key format - default PEM
 -CAform arg     - CA format - default PEM
 -CAkeyform arg  - CA key format - default PEM
 -in arg         - input file - default stdin
 -out arg        - output file - default stdout
 -passin arg     - private key password
 -envpassin arg  - read private key password from environment variable "arg"
 -serial         - print serial number value
 -hash           - print hash value
 -subject        - print subject DN
 -issuer         - print issuer DN
 -startdate      - notBefore field
 -enddate        - notAfter field
 -purpose        - print out certificate purposes
 -dates          - both Before and After dates
 -modulus        - print the RSA key modulus
 -pubkey         - output the public key
 -fingerprint    - print the certificate fingerprint
 -alias          - output certificate alias
 -noout          - no certificate output
 -trustout       - output a "trusted" certificate
 -clrtrust       - clear all trusted purposes
 -clrreject      - clear all rejected purposes
 -addtrust arg   - trust certificate for a given purpose
 -addreject arg  - reject certificate for a given purpose
 -setalias arg   - set certificate alias
 -days arg       - How long till expiry of a signed certificate - def 30 days
 -signkey arg    - self sign cert with arg
 -x509toreq      - output a certification request object
 -req            - input is a certificate request, sign and output.
 -CA arg         - set the CA certificate, must be PEM format.
 -CAkey arg      - set the CA key, must be PEM format
                   missing, it is assumed to be in the CA file.
 -CAcreateserial - create serial number file if it does not exist
 -CAserial       - serial file
 -text           - print the certificate in text form
 -C              - print out C code forms
 -md2/-md5/-sha1/-mdc2 - digest to use
 -extfile        - configuration file with X509V3 extensions to add
 -extensions     - section from config file with X509V3 extensions to add
```

Anhang G

`httpd.conf` – Beispielkonfiguration für Apache-Webserver

```
# The configuration directives are grouped into three basic sections:
#  1. Directives that control the operation of the Apache server process as a
#     whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or 'default' server,
#     which responds to requests that aren't handled by a virtual host.
#     These directives also provide default values for the settings
#     of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be sent to
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.



# ServerType legt fest, wie der Server gestartet wird. Moeglich sind
# "inetd" oder "standalone". Mit inetd wird der Server von inetd ueber einen
# Eintrag in inetd.conf gestartet. Wird der Server auf diese Weise gestartet,
# wird fuer jeden HTTP-Request ein neuer Server gestartet und anschliessend
# beendet. Die Folge ist, dass der Rechner stark belastet werden kann. Und
# ausserdem:
# SSL Servers MUST be standalone, currently.

ServerType standalone


# ServerRoot: The directory the server's config, error, and log files
# are kept in

ServerRoot "/usr/local/apache"


# PidFile: The file the server should log its pid to

PidFile logs/httpd.pid


# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this.  But if yours does (you'll know because
# this file will be  created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.

# ScoreBoardFile logs/apache_runtime_status
```

```
# Saemtliche Server-Direktiven befinden sich in httpd.conf. Daher:

ResourceConfig /dev/null
AccessConfig /dev/null


# Nach wieviel Sekunden die Verbindung geschlossen wird.

Timeout 300

# Ermoeglicht es, mehrere Anfragen ueber eine TCP-Verbindung abzusetzen.

KeepAlive On


# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.

MaxKeepAliveRequests 100


# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.

KeepAliveTimeout 15


# Server-pool size regulation.  Rather than making you guess how many
# server processes you need, Apache dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).
#
# It does this by periodically checking how many servers are waiting
# for a request.  If there are fewer than MinSpareServers, it creates
# a new spare.  If there are more than MaxSpareServers, some of the
# spares die off.  The default values are probably OK for most sites.

MinSpareServers 5
MaxSpareServers 10


# Number of servers to start initially --- should be a reasonable ballpark
# figure.

StartServers 5


# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# the system with it as it spirals down...

MaxClients 150


# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.  The child will exit so
# as to avoid problems after prolonged use when Apache (and maybe the
```

```
# libraries it uses) leak memory or other resources.  On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries.
#
# NOTE: This value does not include keepalive requests after the initial
#       request per connection. For example, if a child process handles
#       an initial request and 10 subsequent "keptalive" requests, it
#       would only count as 1 request towards this limit.
#

MaxRequestsPerChild 10000


# Die Port-Direktive legt fest, an welchem Port der Server horcht. Sind
# zusaetzlich
# noch Listen oder BindAddress gesetzt, hat die Direktive keine Wirkung. Soll
# allerdings der Default-Port ein anderer als 80 sein, muss der Port ueber
# die Port-Anweisung gesetzt sein.
# Port setzt ausserdem die Umgebungs-Variable "SERVER_PORT", die fuer CGI und
# SSI benoetigt wird.
# The default port for SSL is 443...

Port 80



############################################################################
############################################################################
#
# Die folgenden Zeilen mit den LoadModule- und AddModule-Direktiven sind fuer
# einen DSO-Apache gedacht. Kommt ein Nicht-DSO-Server zum Einsatz, sollten die
# Zeilen bis zu der Zeile   "# Ende der DSO-relevanten Direktiven"
# auskommentiert oder geloescht werden.
#
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Please read the file README.DSO in the Apache 1.3 distribution for more
# details about the DSO mechanism and run `httpd -l' for the list of already
# built-in (statically linked and thus always available) modules in your httpd
# binary.
#
# Note: The order is which modules are loaded is important.  Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so
LoadModule mmap_static_module   libexec/mod_mmap_static.so
LoadModule vhost_alias_module   libexec/mod_vhost_alias.so
LoadModule env_module           libexec/mod_env.so
# LoadModule define_module      libexec/mod_define.so
LoadModule config_log_module    libexec/mod_log_config.so
# LoadModule agent_log_module   libexec/mod_log_agent.so
# LoadModule referer_log_module libexec/mod_log_referer.so
# LoadModule mime_magic_module  libexec/mod_mime_magic.so
LoadModule mime_module          libexec/mod_mime.so
LoadModule negotiation_module   libexec/mod_negotiation.so
# LoadModule status_module      libexec/mod_status.so
# LoadModule info_module        libexec/mod_info.so
# LoadModule includes_module    libexec/mod_include.so
```

```
# LoadModule autoindex_module   libexec/mod_autoindex.so
LoadModule dir_module           libexec/mod_dir.so
LoadModule cgi_module           libexec/mod_cgi.so
# LoadModule asis_module        libexec/mod_asis.so
# LoadModule imap_module        libexec/mod_imap.so
LoadModule action_module        libexec/mod_actions.so
# LoadModule speling_module     libexec/mod_speling.so
# LoadModule userdir_module     libexec/mod_userdir.so
LoadModule alias_module         libexec/mod_alias.so
# LoadModule rewrite_module     libexec/mod_rewrite.so
LoadModule access_module        libexec/mod_access.so
# LoadModule auth_module        libexec/mod_auth.so
# LoadModule anon_auth_module   libexec/mod_auth_anon.so
# LoadModule dbm_auth_module    libexec/mod_auth_dbm.so
# LoadModule digest_module      libexec/mod_digest.so
# LoadModule proxy_module       libexec/libproxy.so
# LoadModule cern_meta_module   libexec/mod_cern_meta.so
# LoadModule expires_module     libexec/mod_expires.so
# LoadModule headers_module     libexec/mod_headers.so
# LoadModule usertrack_module   libexec/mod_usertrack.so
# LoadModule example_module     libexec/mod_example.so
# LoadModule unique_id_module   libexec/mod_unique_id.so
LoadModule setenvif_module      libexec/mod_setenvif.so

<IfDefine SSL>
    LoadModule ssl_module           libexec/libssl.so
</IfDefine>



#  Reconstruction of the complete module list from all available modules
#  (static and shared ones) to achieve correct module execution order.
#  [WHENEVER YOU CHANGE THE LOADMODULE SECTION ABOVE UPDATE THIS, TOO]

ClearModuleList
AddModule       mod_mmap_static.c
AddModule       mod_vhost_alias.c
AddModule       mod_env.c
# AddModule     mod_define.c
AddModule       mod_log_config.c
# AddModule     mod_log_agent.c
# AddModule     mod_log_referer.c
# AddModule     mod_mime_magic.c
AddModule       mod_mime.c
AddModule       mod_negotiation.c
# AddModule     mod_status.c
# AddModule     mod_info.c
# AddModule     mod_include.c
# AddModule     mod_autoindex.c
AddModule       mod_dir.c
AddModule       mod_cgi.c
# AddModule     mod_asis.c
# AddModule     mod_imap.c
AddModule       mod_actions.c
# AddModule     mod_speling.c
# AddModule     mod_userdir.c
AddModule       mod_alias.c
# AddModule     mod_rewrite.c
AddModule       mod_access.c
# AddModule     mod_auth.c
# AddModule     mod_auth_anon.c
# AddModule     mod_auth_dbm.c
# AddModule     mod_digest.c
```

```
# AddModule     mod_proxy.c
# AddModule     mod_cern_meta.c
# AddModule     mod_expires.c
# AddModule     mod_headers.c
# AddModule     mod_usertrack.c
# AddModule     mod_example.c
# AddModule     mod_unique_id.c
AddModule       mod_so.c
AddModule       mod_setenvif.c

<IfDefine SSL>
    AddModule mod_ssl.c
</IfDefine>

# Ende der DSO-relevanten Direktiven
#
##############################################################################
##############################################################################



# Hat der Server mehrere Interfaces, horcht er auf jedem Interface, auf den bei
# "Port" angegebenen Port. Es ist auch moeglich, verschiedene IP-Adressen
# anzugeben (soweit der Rechner entsprechend konfiguriert ist). Ebenso sind
# unterschiedliche Hostnames moeglich. Listen hat hoehere
# Prioritaet als Port und BindAdress.

Listen 127.0.0.42:80


<IfDefine SSL>
    Port   443
    Listen 127.0.0.42:443
</IfDefine>


# User- und Group-Direktive setzen die UserID bzw. Group-ID, unter der der
# Server laeuft. Am besten einen eigenen User und Group einrichten und Server
# als root starten. Nach dem Start wechselt der Server auf den weniger
# priviligierten User-/Group-ID.

User www
Group wwwadm


# An wen geht eine Meldung im Fehlerfall, z.B. steht diese Adresse
# auf vom Server generierten Seiten mit Fehlermeldungen.

ServerAdmin wwwadmin@name.de


# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
ServerName www.name.de
```

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.

DocumentRoot "/var/www/www80"


# Durch folgende Anweisung wird Default-Zugriff auf das gesamte Datei-System
# des Rechners abgeschaltet, sowie alle Includes und .htaccess-overrides.
# Bei Bedarf kann der Zugriff fuer einzelne Verzeichnisse durch eine neue
# Directory-Anweisung wieder eingeschaltet werden.
# Directory bezieht sich auf den absoluten Pfad.
# Die Anweisungen Directory, Files, Location, bilden eine Hirarchie mit in
# dieser Reihenfolge zunehmender Prioritaet. Die Angabe von
# Verzeichnissen kann fuer alle drei Direktiven auch durch Wildcards und
# Regulaere Ausdruecke erfolgen.

<Directory />
    order deny,allow
    #  Zugriff aus allen Domains verbieten
    deny from all
    #  ExecCGI, FollowSymLinks, Includes, Indexes, Multiview abschalten
    Options none
    #  .htaccess-Dateien ignorieren
    AllowOverride none
</Directory>


# Wegen der oben erwaehnten Prioritaets-Hierarchie muss hier eine Directory-D
# verwendet werden, obwohl eine "Location /" dasselbe bezeichnet.

<Directory /var/www/www80>

    # Nach "deny from all" in der "Directory /"-Anweisung wird hier wieder der
    # Zugriff gestattet. Der Zugriff sollte auch gestattet werden, damit
    # bei einer Standard-Anfrage der Default "index.html" zurueckgeliefert
    # werden kann. In darauffolgenden "Directory"-Anweisungen kann der
    # Zugriff wieder auf HTML-Dateien beschraenkt werden.

    Order allow,deny
    allow from all

</Directory>


# Fuer alle Unterverzeichnisse Zugriff verbieten.
# Muss Directory statt Location sein, da sonst die Beschraenkung mit deny
# nicht mehr ueber Files fuer HTML-docs gemildert werden koennte (Location hat
# hoehere Prioritaet als Files).

<Directory /var/www/www80/*/>
    order deny,allow
    deny from all
</Directory>


# Durch folgende Anweisungen wird der Name der Index-Datei festgelegt

DirectoryIndex index.html
```

```
# UseCanonicalName:  (new for 1.3)  With this setting turned on, whenever
# Apache needs to construct a self-referencing URL (a URL that refers back
# to the server the response is coming from) it will use ServerName and
# Port to form a "canonical" name.  With this setting off, Apache will
# use the hostname:port that the client supplied, when possible.  This
# also affects SERVER_NAME and SERVER_PORT in CGI scripts.

UseCanonicalName On


# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
# Die neuen, SSL-bezogenen MIMI-Types werden weiter unten im SSL-Teil
# des Servers mit der Direktive "AddType" hinzugefuegt. Somit ist
# keine Aenderung von "mime.types" noetig.

TypesConfig conf/mime.types


# Default Mime-Type, wenn kein passender Eintrag gefunden wurde.

DefaultType text/plain


# "Off" in der folgenden Direktive schreibt die IP-Nummern der Clients statt
# deren Namen in die Log-Dateien. Andernfalls werden die Namen per
# DNS-Lookup geholt, was sich negativ auf die Performanz des Servers
# auswirkt.

HostnameLookups Off


# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.

ErrorLog logs/80error.log


# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.

LogLevel warn


# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
# Custom logging
# Datum/Zeit (t), 1. Zeile der Client-Anfrage(r), Http-Statuscode(s), Groesse
# der gelieferten Datei (b), Protokollierung von Http-Headern (i)

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent


# If you would like to have agent and referer logfiles, uncomment the
# following directives.
```

```
#
#CustomLog logs/referer_log referer
#CustomLog logs/agent_log agent


# If you prefer a single logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.

CustomLog logs/80access.log combined


# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail

ServerSignature Off


# Alias fuer die Icons

Alias /icons/ /usr/local/apache/icons/


# Folgendes kann als Location-Direktive gesetzt werden (da innerhalb
# DocumentRoot und hoehere Prioritaet als Directory).

<Location /icons/*.gif>
    Order deny,allow
    allow from all
</Location>


<Location /icons/*.jpg>
    Order deny,allow
    allow from all
</Location>


# CGI-Skripte sollten nicht im Dokument-Root stehen, werden aber relativ dazu
# verwendet. Daher

ScriptAlias /cgi/ /usr/local/apache/cgi-bin/


<Files sample.cgi>

    # Setzen des cgi-Handlers auf genau eine Datei. Durch mehrere
    # Files-Anweisungen entsprechend auf mehrere Dateien. Etwas Paranoid,
    # alternativ kann jede auf cgi endende Datei "freigeschaltet" werden.

    SetHandler cgi-script
    order deny,allow
    allow from all

</Files>


# Die beiden Borwser-Typen (Mozilla/2 gilt sowohl fuer Navigator 2 wie
# auch fuer aeltere MSIE-Versionen, die sich als Navigator ausgegeben
# haben) haben Probleme mit persistenten Verbindungen. Daher keine
# persistenten Verbindungen fuer diese Browser ("noekeepalive").
```

```
# Ausserdem hat der MSIE Probleme mit HTTP/1.1-Antworten. Die Anfragen
# kommen zwar in HTTP/1.1 vom MSIE; durch die "downgrade-1.0"-Option wird dem
# Apache allerdings eine HTTP/1.0-Anfrage vorgetaeuscht, so dass die
# Antwort in HTTP/1.0 erfolgt und die Probleme mit MSIE vermieden werden.

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

# Schwierigkeiten mit dem von mod_rewrite eingefuegten
# "Vary"-Headern. Daher keine solche Header bei folgendem Browser-Typ
# einfuegen.

# BrowserMatch "MSIE 4\.0" force-no-vary

# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
# Der Apache signalisiert in seinen Antworten, dass er ein HTTP/1.1
# faehiger Server ist. Einige Clients interpretieren dass
# faelschlicherweise Angabe zum Protokoll der Anworten und verweigern
# die Annahme der Antwort. Daher wird diesen Clients mit folgenden
# Direktiven ein HTTP/1.0 angeboten...
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0


################################################################################
################################################################################
##
##  SSL Global Context
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts.
##

<IfDefine SSL>

    # Neue SSL-relevante Mime-Types hinzufuegen.

    AddType application/x-pkcs7-crl        crl
    AddType application/x-x509-ca-cert     cacrt
    AddType application/x-x509-email-cert  emailcrt
    AddType application/x-x509-user-cert   usercrt

</IfDefine>


<IfModule mod_ssl.c>

    #   Pass Phrase Dialog:
    #   Configure the pass phrase gathering process.
    #   The filtering dialog program (‘builtin’ is a internal
    #   terminal dialog) has to provide the pass phrase on stdout.

    SSLPassPhraseDialog      builtin


    #   Inter-Process Session Cache:
    #   Configure the SSL Session Cache: First either ‘none’
    #   or ‘dbm:/path/to/file’ for the mechanism to use and
    #   second the expiring timeout (in seconds).
    #
```

```
    #   "shm" benoetigt die MM-Bibliothek.

    # SSLSessionCache        none
    SSLSessionCache          shm:logs/ssl_scache(512000)
    # SSLSessionCache        dbm:logs/ssl_scache
    SSLSessionCacheTimeout   300


    #   Semaphore:
    #   Configure the path to the mutual explusion semaphore the
    #   SSL engine uses internally for inter-process synchronization.

    # SSLMutex   file:logs/ssl_mutex
    SSLMutex     sem

    #   Pseudo Random Number Generator (PRNG):
    #   Configure one or more sources to seed the PRNG of the
    #   SSL library. The seed data should be of good random quality.
    #
    #   Leider kein /dev/random unter Solaris. Es gibt aber die Moeglichkeit,
    #   ueber externe Programme Zufallsdaten zuzufuehren (z.B. "truerand" im
    #   Mod-SSL-Quell-Verzeichnis unter "pkg.contrib")
    # SSLRandomSeed startup exec:/usr/local/bin/truerand 16
    #   Ohne "truerand":

    SSLRandomSeed startup builtin
    SSLRandomSeed connect builtin


    #   Logging:
    #   The home of the dedicated SSL protocol logfile. Errors are
    #   additionally duplicated in the general error log file.  Put
    #   this somewhere where it cannot be used for symlink attacks on
    #   a real server (i.e. somewhere where only root can write).
    #   Log levels are (ascending order: higher ones include lower ones):
    #   none, error, warn, info, trace, debug.

    SSLLog      logs/ssl_engine.log
    SSLLogLevel info

</IfModule>



<IfDefine SSL>

    ##
    ## SSL Virtual Host Context
    ##

    #  Grundsaetzlich gilt, dass jeder virtuelle Host die Werte der im
    #  "Hauptserver" gesetzten Direktiven "erbt".


    <VirtualHost www.name.de:443>

        #  Im Document-Root-Verzeichnis wird auf eine
        #  Client-Authentifizierung verzichtet.

        #  General setup for the virtual host

        DocumentRoot    /var/www/www443
        ServerName      www.name.de
```

```
ServerAdmin     wwwadm@name.de
ErrorLog        logs/443error.log
CustomLog       logs/443access.log \
                "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

SSLVerifyClient none
SSLEngine       on
SSLProtocol     all -SSLv2

#  SSL Cipher Suite: Ueber diese Direktive kann festgelegt werden,
#  welche Krypto-Algorithmen und in welcher Prioritaet diese verwendet
#  werden.

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

SSLCertificateFile        /usr/local/apache/ssl/serverCert.pem
SSLCertificateKeyFile     /usr/local/apache/ssl/serverKey.pem
# SSLCertificateChainFile /usr/local/apache/ssl/cacerts/cachain.pem

SSLCACertificatePath      /usr/local/apache/ssl/cacerts
# SSLCACertificateFile    /usr/local/apache/ssl/CAcerts.pem

SSLCARevocationPath       /usr/local/apache/ssl/crls
# SSLCARevocationFile     /usr/local/apache/ssl/crls.pem


<Directory /var/www/www443>

    # Nach "deny from all" in der "Directory /"-Anweisung wird hier
    # wieder der Zugriff gestattet.

    allow from all

</Directory>


<Directory /var/www/www443/*/>
    order deny,allow
    deny  from all
</Directory>


<Directory /var/www/www443/restricted>

    # Auf Dateien in diesem Verzeichnis sollen nur Clients mit
    # gueltigem Zertifikat zugreifen koennen. Das erfordert u.U.
    # ein erneutes SSL-Handshake ("Renegotiation"), da eine SSL-Session
    # besteht, der Client aber noch nicht authentifiziert wurde.

    SSLVerifyClient  require
    SSLVerifyDepth   5
    SSLOptions       +ExportCertData +StrictRequire +OptRenegotiate


    # Verfeinerte Zugriffssteuerung ueber Zertifikat-Attribute.
    # Hat nur indirekt etwas mit der x509-basierten Zugriffskontrolle
    # zutun, da nicht nur die Gueltigkeit des Client-Zertifkats
    # Vorraussetzung ist, sondern das Zertifikat gewisse
    # (DN-)Eigenschaften besitzen muss.
    # Vom Prinzip her eher mit einer Passwort-gesteuerten
    # Zugriffskontrolle vergleichbar.

    SSLRequire       %{SSL_CLIENT_S_DN_C} eq "DE" \
```

```
                             and %{SSL_CLIENT_S_DN_L} eq "Kiel" \
                             and %{SSL_CLIENT_S_DN_O} eq "WWW UnLtd"

            allow from all

        </Directory>

        # Wieder mal Browser-"Macken" abfangen.
        SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown

    </VirtualHost>

</IfDefine>
```

Anhang H

Zertifikat-Analyse-Tools

H.1 Werkzeuge für X.509-Zertifikate

H.1.1 `md5` und `md5sum`

Bei der Ausstellung von X.509-Zertifikaten durch eine CA können Werkzeuge hilfreich sein, die eine kryptographisch starke Prüfsumme (einen *Message Digest* oder auch *Hash*-Wert) über einer Datei berechnen können. Ein sehr bekannter Message Digest ist das Verfahren „MD5“. Mit seiner Hilfe kann eine Prüfsumme z.B. über einem PEM-codierten Zertfizierungs-Request ermittelt werden, also über einem Zertifizierungsantrag, der in einer ASCII-Darstellung etwa dieser Art vorliegt:


```
-----BEGIN CERTIFICATE-----
MIIFfDCCBGSgAwIBAgIBAjANBgkqhkiG9w0BAQQFADCBlTELMAkGA1UEBhMCREUx
ITAfBgNVBAoTGERldXRzY2hlcyBGb3JzY2h1bmdzbmV0ejEQMA4GA1UECxMHREZO
...
MsX23YbB+MAVsMaChV0lzsIbh6lSPPEif6o0pkEJM1JhU0I+TCuaSrAXs3S7BRbS
uRyOjKjcfvw6WqGG8lrN+WMfjyjdQDEUjn0lCIO+S/RdjVQ8WnEdQK44VGFInL5i
5atnqN+GcTY+lco5TjfsAA==
-----END CERTIFICATE-----
```


Befindet sich ein solcher Request z.B. in der Datei “certreq.pem”, so kann eine MD5-Prüfsumme mit einem der folgenden Tools daraus berechnet werden:

md5sum

Teil der PGP-Source-Distribution (Verzeichnis “contrib”):

```
$ md5sum certreq.pem
3b7dad5e5e62a4b7d2782dc703205b30 certreq.pem
```

md5

Eines der Programme aus dem OpenSSL-Paket (ggf. als ‘openssl md5 ...’ aufrufen, falls es sonst nicht gefunden wird):

```
$ openssl md5 certreq.pem
MD5(certreq.pem)= 3b7dad5e5e62a4b7d2782dc703205b30
```

(Ein MD5-Programm kann auch als Teil einer Betriebssystem-Installation ohnehin schon auf einem Rechner vorhanden sein.)

Mit einem solchen MD5-Tool läßt sich also eine eindeutige, kurze Prüfsumme eines Zertifizierungs-Requests, ähnlich dem PGP-Fingerprint bei PGP-Schlüsseln, errechnen. Derjenige, der den entsprechenden Zertifizierungsantrag stellt, hat damit die Möglichkeit, auch bei einer X.509-Zertifizierung seinen Schlüssel anhand dieses eindeutigen Merkmals verwechslungsfrei zu identifizieren, indem er die MD5-Prüfsumme auf dem Antrag mit angibt.

H.1.2 OpenSSLs `x509`

OpenSSL stellt nicht nur Routinen bzw. Programme für die SSL-Kommunikation und zur Zertifizierung bereit, sondern bietet darüber hinaus auch einige Werkzeuge zur Handhabung und Untersuchung von X.509-Zertifikaten. Eines davon ist **x509** (vgl. 12.3 / Übersicht der x509-Optionen in Anhang F), mit dem sich der Inhalt von Zertifikaten im X.509-Format anzeigen läßt.

Es liege beispielsweise ein Zertifikat der DFN-PCA in einer Datei “dfnpca.pem” in BASE64-Codierung vor, wie sie in [BF93] definiert ist:

```
$ cat dfnpca.pem
-----BEGIN CERTIFICATE-----
MIIFfDCCBGSgAwIBAgIBAjANBgkqhkiG9w0BAQQFADCBlTELMAkGA1UEBhMCREUx
ITAfBgNVBAoTGERldXRzY2hlcyBGb3JzY2h1bmdzbmV0ejEQMA4GA1UECxMHREZO
LVBDQTEuMCwGA1UEAxMlREZOIFRvcCBMZXZlbCBDZXJ0aWZpY2F0aW9uIEF1dGhv
cml0eTEhMB8GCSqGSIb3DQEJARYSY2VydGlmeUBwY2EuZGZuLmRlMB4XDTk4MTEw
MzE2NDcyM1oXDTAwMTEwMjE2NDcyM1owgZIxCzAJBgNVBAYTAkRFMSEwHwYDVQQK
ExhEZXV0c2NoZXMgRm9yc2NodW5nc25ldHoxEDAOBgNVBAsTB0RGTi1QQ0ExKzAp
...
MsX23YbB+MAVsMaChV0lzsIbh6lSPPEif6o0pkEJM1JhU0I+TCuaSrAXs3S7BRbS
uRyOjKjcfvw6WqGG8lrN+WMfjyjdQDEUjn0lCIO+S/RdjVQ8WnEdQK44VGFInL5i
5atnqN+GcTY+lco5TjfsAA==
-----END CERTIFICATE-----
```

Dann läßt sich mittels des Programms **x509** die Struktur des Zertifikates in dieser Datei und der Inhalt seiner Bestandteile sichtbar machen:

```
$ x509 -text -noout < dfnpca.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=DE, O=Deutsches Forschungsnetz, OU=DFN-PCA, CN=DFN Top Level Certifica-
tion Authority/Email=certify@pca.dfn.de
        Validity
            Not Before: Nov  3 16:47:23 1998 GMT
            Not After : Nov  2 16:47:23 2000 GMT
        Subject: C=DE, O=Deutsches Forschungsnetz, OU=DFN-PCA, CN=DFN Server Certificati-
on Authority/Email=certify@pca.dfn.de
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:be:63:c9:0a:5a:57:0e:1b:d3:b3:04:90:00:f6:
```

```
                    68:96:69:2a:35:b3:49:4e:8e:7c:06:c7:5a:b3:11:
                    d5:54:fc:b6:21:34:83:c0:15:0b:12:63:1b:11:f5:
                                        ...
                    13:c6:bb:df:90:95:d1:3e:df:21:cd:9d:56:8a:84:
                    5c:93
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape Cert Type:
                SSL CA
            Netscape Base Url:
                https://mystic.pca.dfn.de/
            Netscape CA Policy Url:
                http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html
            Netscape Comment:
                This certificate was issued by the DFN-PCA, the Top Level
Certification Authority of the German Research Network (Deutsches Forschungsnetz, DFN).
The key owner’s identity was authenticated in accordance with the DFN World Wide Web Poli-
cy, Version 0.90
            Netscape Revocation Url:
                cgi/check-rev.cgi?
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage:
                Certificate Sign, CRL Sign
    Signature Algorithm: md5WithRSAEncryption
        a2:ad:51:c0:66:89:0c:7e:52:3e:6e:60:25:a2:45:0a:60:d1:
        ee:57:63:7e:a2:08:50:12:1f:f1:50:4f:08:4c:3f:db:b6:8d:
        74:32:60:aa:35:11:f6:ce:d3:a0:74:89:ce:ad:2b:05:c8:75:
                                ...
        8e:7d:25:08:83:be:4b:f4:5d:8d:54:3c:5a:71:1d:40:ae:38:
        54:61:48:9c:be:62:e5:ab:67:a8:df:86:71:36:3e:95:ca:39:
        4e:37:ec:00
```

x509 (das Programm, nicht der Standard!) weist eine Vielzahl von Optionen auf (siehe F), mit denen es z.B. auch möglich ist, sich gezielt nur eine einzelne Komponente eines Zertifikates anzeigen zu lassen (hilfreich beispielsweise bei der Weiterverarbeitung mit Shellskripten):

```
$ x509 -subject -noout < dfnpca.pem
subject=/C=DE/O=Deutsches Forschungsnetz/OU=DFN-PCA/CN=DFN Server Certificati-
on Authority/Email=certify@pca.dfn.de
```

Doch **x509** kann noch mehr: Mit seiner Hilfe lassen sich Zertifikate aus einem der drei Darstellungsformate DER, PEM oder NET in jedes der jeweils anderen umwandeln. Will man beispielsweise das Zertifikat in der oben genannten Datei “dfnpca.pem” in die DER-Codierung umwandeln, z.B. um es mit einem anderen Programm weiterzuverarbeiten, das nur mit DER-codierten Zertifikaten umgehen kann, so erreicht man das durch folgenden Aufruf:

```
x509 -inform PEM -outform DER < dfnpca.pem  > dfnpca.der
```

oder auch

```
x509 -inform PEM -outform DER -in dfnpca.pem  -out dfnpca.der
```

Anschließend liegt in der so erzeugten Datei “dfnpca.der” das Zertifikat in ASN.1-DER-Codierung vor.

H.1.3 `asn1parse`

Ein weiteres Tool aus dem SSLeay-Paket ist **asn1parse** (siehe auch F). Es dient dazu, die ASN.1-Codierung eines Zertifikates nachvollziehen oder überprüfen zu können.

Gehen wir wieder wie im obigen Beispiel von einem Zertifikat in DER-Codierung in der Datei "dfnpca.der" aus, so sähen Aufruf und Ausgabe von asn1parse (auszugsweise) so aus:

```
$  asn1parse -inform DER < der/dfnpca.der
    0:d=0  hl=4 l=1404 cons: SEQUENCE
    4:d=1  hl=4 l=1124 cons: SEQUENCE
    8:d=2  hl=2 l=   3 cons: cont [ 0 ]
   10:d=3  hl=2 l=   1 prim: INTEGER           :02
   13:d=2  hl=2 l=   1 prim: INTEGER           :02
   16:d=2  hl=2 l=  13 cons: SEQUENCE
   18:d=3  hl=2 l=   9 prim: OBJECT            :md5WithRSAEncryption
   29:d=3  hl=2 l=   0 prim: NULL
   31:d=2  hl=3 l= 149 cons: SEQUENCE
   34:d=3  hl=2 l=  11 cons: SET
   36:d=4  hl=2 l=   9 cons: SEQUENCE
   38:d=5  hl=2 l=   3 prim: OBJECT            :countryName
   43:d=5  hl=2 l=   2 prim: PRINTABLESTRING   :DE
   47:d=3  hl=2 l=  33 cons: SET
   49:d=4  hl=2 l=  31 cons: SEQUENCE
   51:d=5  hl=2 l=   3 prim: OBJECT            :organizationName
   56:d=5  hl=2 l=  24 prim: PRINTABLESTRING   :Deutsches Forschungsnetz
   82:d=3  hl=2 l=  16 cons: SET
   84:d=4  hl=2 l=  14 cons: SEQUENCE
   86:d=5  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
   91:d=5  hl=2 l=   7 prim: PRINTABLESTRING   :DFN-PCA
  100:d=3  hl=2 l=  46 cons: SET
  102:d=4  hl=2 l=  44 cons: SEQUENCE
  104:d=5  hl=2 l=   3 prim: OBJECT            :commonName
  109:d=5  hl=2 l=  37 prim: PRINTABLESTRING   :DFN Top Level Certification Authority
  148:d=3  hl=2 l=  33 cons: SET
  150:d=4  hl=2 l=  31 cons: SEQUENCE
  152:d=5  hl=2 l=   9 prim: OBJECT            :emailAddress
  163:d=5  hl=2 l=  18 prim: IA5STRING         :certify@pca.dfn.de
  183:d=2  hl=2 l=  30 cons: SEQUENCE
  185:d=3  hl=2 l=  13 prim: UTCTIME           :981103164723Z
  200:d=3  hl=2 l=  13 prim: UTCTIME           :001102164723Z
  [...]
  368:d=3  hl=2 l=  13 cons: SEQUENCE
  370:d=4  hl=2 l=   9 prim: OBJECT            :rsaEncryption
  381:d=4  hl=2 l=   0 prim: NULL
  383:d=3  hl=4 l= 271 prim: BIT STRING
  658:d=2  hl=4 l= 470 cons: cont [ 3 ]
  662:d=3  hl=4 l= 466 cons: SEQUENCE
  666:d=4  hl=2 l=  17 cons: SEQUENCE
  668:d=5  hl=2 l=   9 prim: OBJECT            :Netscape Cert Type
  679:d=5  hl=2 l=   4 prim: OCTET STRING
  685:d=4  hl=2 l=  41 cons: SEQUENCE
  687:d=5  hl=2 l=   9 prim: OBJECT            :Netscape Base Url
  698:d=5  hl=2 l=  28 prim: OCTET STRING
  728:d=4  hl=2 l=  65 cons: SEQUENCE
  730:d=5  hl=2 l=   9 prim: OBJECT            :Netscape CA Policy Url
  741:d=5  hl=2 l=  52 prim: OCTET STRING
  795:d=4  hl=4 l= 268 cons: SEQUENCE
  799:d=5  hl=2 l=   9 prim: OBJECT            :Netscape Comment
  810:d=5  hl=3 l= 254 prim: OCTET STRING
 1067:d=4  hl=2 l=  33 cons: SEQUENCE
```

```
 1069:d=5  hl=2 l=   9 prim: OBJECT            :Netscape Revocation Url
 1080:d=5  hl=2 l=  20 prim: OCTET STRING
 1102:d=4  hl=2 l=  15 cons: SEQUENCE
 1104:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 1109:d=5  hl=2 l=   1 prim: BOOLEAN           :255
 1112:d=5  hl=2 l=   5 prim: OCTET STRING
 1119:d=4  hl=2 l=  11 cons: SEQUENCE
 1121:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
 1126:d=5  hl=2 l=   4 prim: OCTET STRING
  [...]
```

Auch **asn1parse** ist wie **x509** in der Lage, Zertifikate in den drei Darstellungsformen DER, PEM und TXT zu verarbeiten. Es kann aber darüberhinaus auch die Struktur und den Inhalt beliebiger anderer Objekte, die ASN.1-codiert vorliegen, anzeigen.

H.1.4 `dumpasn1`

Von PETER GUTMANN gibt es ein kleines Tool namens **dumpasn1**, mit dessen Hilfe man sich ebenfalls die Bestandteile eines X.509-Zertifikates anzeigen lassen kann. (Quelle: `http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c`)

dumpasn1 stellt die einzelnen ASN.1-Strukturen dar, aus denen sich das jeweilige X.509-Zertifikat zusammensetzt. Dazu muß das Zertifikat in einer Datei in DER-Codierung (*distinguished encoding rules*, eine rechnerunabhängige, eindeutige Repräsentation für die Übertragung von Informationen in ASN.1-Struktur zwischen beliebigen Rechner- und Betriebssystemplattformen, daher auch als *network encoding* bezeichnet) vorliegen.

Der Aufruf und die Ausgaben von dumpasn1 mit demselben Zertifikat der DFN-PCA wie oben (das vorher lediglich mit **x509** in die DER-Darstellung umgewandelt wurde) würde dann auszugsweise so aussehen:

```
$ dumpasn1 dfnpca.der

[...]
   0 30 1404: SEQUENCE {
   4 30 1124:   SEQUENCE {
   8 A0    3:     [0] {
  10 02    1:       INTEGER 2
            :       }
  13 02    1:     INTEGER 2
  16 30   13:     SEQUENCE {
  18 06    9:       OBJECT IDENTIFIER
            :         md5withRSAEncryption (1 2 840 113549 1 1 4)
  29 05    0:       NULL
            :       }
  31 30  149:     SEQUENCE {
  34 31   11:       SET {
  36 30    9:         SEQUENCE {
  38 06    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
  43 13    2:           PrintableString 'DE'
            :           }
            :         }
  47 31   33:       SET {
  49 30   31:         SEQUENCE {
  51 06    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
```

```
  56 13   24:           PrintableString 'Deutsches Forschungsnetz'
            :           }
            :         }
  82 31   16:       SET {
  84 30   14:         SEQUENCE {
  86 06    3:           OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
  91 13    7:           PrintableString 'DFN-PCA'
            :           }
            :         }
 [...]
 364 30  290:     SEQUENCE {
 368 30   13:       SEQUENCE {
 370 06    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
 381 05    0:         NULL
            :         }
 383 03  271:       BIT STRING 0 unused bits
            :         30 82 01 0A 02 82 01 01 00 BE 63 C9 0A 5A 57 0E
            :         1B D3 B3 04 90 00 F6 68 96 69 2A 35 B3 49 4E 8E
            :         7C 06 C7 5A B3 11 D5 54 FC B6 21 34 83 C0 15 0B
            :         12 63 1B 11 F5 86 71 8D 1A DC 6F F2 BF FB C4 FA
            :         9E 90 BF 50 69 8F 84 61 EA 8A AB 68 21 C9 A6 79
            :         9F 46 6D 8F F1 DC 13 3E 66 F5 E4 F9 CE AB CC 65
            :         3F BC 25 FE D1 5A 5B D4 29 E9 70 F9 5E 24 6D 21
            :         DC 33 DD C0 5D 0F 15 E0 7C DC 9C 4A 5C 6A 74 58
            :                 [ Another 142 bytes skipped ]
            :       }
 658 A3  470:     [3] {
 662 30  466:       SEQUENCE {
 666 30   17:         SEQUENCE {
 668 06    9:           OBJECT IDENTIFIER
            :             netscape-cert-type (2 16 840 1 113730 1 1)
 679 04    4:           OCTET STRING
            :             03 02 00 04
            :           }
 685 30   41:         SEQUENCE {
 687 06    9:           OBJECT IDENTIFIER
            :             netscape-base-url (2 16 840 1 113730 1 2)
 698 04   28:           OCTET STRING
            :             16 1A 68 74 74 70 73 3A 2F 2F 6D 79 73 74 69 63
            :             2E 70 63 61 2E 64 66 6E 2E 64 65 2F
            :           }
 728 30   65:         SEQUENCE {
 730 06    9:           OBJECT IDENTIFIER
            :             netscape-ca-policy-url (2 16 840 1 113730 1 8)
 741 04   52:           OCTET STRING
            :             16 32 68 74 74 70 3A 2F 2F 77 77 77 2E 70 63 61
            :             2E 64 66 6E 2E 64 65 2F 64 66 6E 70 63 61 2F 70
            :             6F 6C 69 63 79 2F 77 77 77 70 6F 6C 69 63 79 2E
            :             68 74 6D 6C
            :           }
            :             72 69 74 79 20 6F 66 20 74 68 65 20 47 65 72 6D
            :             61 6E 20 52 65 73 65 61 72 63 68 20 4E 65 74 77
            :             6F 72 6B 0A 28 44 65 75 74 73 63 68 65 73 20 46
            :                     [ Another 126 bytes skipped ]
            :           }
[...]
            :   }
0 warnings, 0 errors.
```

Dank der separaten Konfigurationsdatei "dumpasn1.cfg" (oder einer per Kommandozeilenoption '-c*file*' anzugebenden Datei) lassen sich neuen numerischen Object-Identifiern (OIDs), wie sie beispielsweise durch neue Versionen von X.509 eingeführt werden oder aus privaten Zertifikaterweiterungen resultieren könnten, Klartextbezeichnungen zuordnen, so daß die Ausgaben von dumpasn1 leichter verständlich bleiben.

H.2 Werkzeuge für PGP-Zertifikate

H.2.1 `pgpacket`

Ein ähnliches Tool wie die unter H.1 genannten existiert auch für PGP bzw. dessen Schlüssel- und Zertifikatformat, inzwischen erfreulicherweise sogar für RSA- und für DSS/DH-PGP-Keys: **pgpacket** von MARK E. SHOULSON `<shoulson@cs.columbia.edu>`, zu finden u. a. auf `ftp://ftp.cert.dfn.de/pub/tools/crypt/pgp/utils/pgpacket3.1.pl.gz`.

Ein Beispiel soll die Arbeit mit pgpacket verdeutlichen. Im Public-Keyring befindet sich u.a. der Schlüssel von Vera Heinau mit den folgenden Signaturen:

```
$ pgp -kvv heinau
[...]
Key ring: 'pubring.pgp', looking for user ID "heinau".
Type Bits/KeyID    Date       User ID
pub  1024/00F0D5E9 1995/08/29 Vera Heinau <heinau@cis.fu-berlin.de>
sig*      4F570BA3             Ingmar Camphausen <ingmar@aurora.in-berlin.de>
sig       00F0D5E9             Vera Heinau <heinau@cis.fu-berlin.de>
                              Vera Heinau <heinau@zedat.fu-berlin.de>
sig*      84B3BC11             in-ca@in-berlin.de (SIGN EXPIRE:2000-01-01) Regiona-
le CA des Individual Network e.V.
sig       00F0D5E9             Vera Heinau <heinau@cis.fu-berlin.de>
[...]
1 matching key found.
```

Extrahiert man ihn nun (mittels 'pgp -kx heinau keyfile') in eine Datei "keyfile", so läßt er sich wie folgt mit pgpacket analysieren:

```
$ pgpacket -u="pubring.pgp" keyfile

---------------------------
Packet Type:    Public-Key Packet
Length: 141
Version Byte:   3
Key Created:    29 Aug 1995  23:10:02
Valid forever
Algorithm:      1 (RSA)
N:      0xC3DC0FB677B34F98C69DC96B1FC136615C4BA3487DB00524D74E1D35DC10CD17823190CFFEFFDA
93CC6651FB40E5DEC049E2A3B1688C61DB1CB21750B14D4EBD62637F8A457A86CC62DBB755279662
D391F6DA7CD64A175A693222E0FED17FF4A8DDBA046C2B8D4A37FCAA59925536F1E6772CC8CB4F84
D3F2A2F2F600F0D5E9
E:      0x13

---------------------------
Packet Type:    User ID Packet
Length: 37
User ID:        "Vera Heinau <heinau@cis.fu-berlin.de>"

---------------------------
Packet Type:    Secret-Key Encrypted Packet (signature)
```

```
Length: 149
Version:        3
Adding 5 bytes of header to digest
Positive ID Key certification (unsupported)
Signature Created:      17 Jan 1998  01:43:52
Signing Key ID: 0x144F2D454F570BA3
    User ID for above key: "Ingmar Camphausen <ingmar@aurora.in-berlin.de>"
Public-Key Algorithm:   1 (RSA)
Message Digest Algorithm:       1 (MD5)
Check bytes:    0x316D
128 bytes of data

---------------------------
Packet Type:    Secret-Key Encrypted Packet (signature)
Length: 149
Version:        3
Adding 5 bytes of header to digest
Generic Key certification
Signature Created:      29 May 1997  00:37:37
Signing Key ID: 0xF2A2F2F600F0D5E9
    User ID for above key: "Vera Heinau <heinau@cis.fu-berlin.de>"
    User ID for above key: "Vera Heinau <heinau@zedat.fu-berlin.de>"
Public-Key Algorithm:   1 (RSA)
Message Digest Algorithm:       1 (MD5)
Check bytes:    0x1219
128 bytes of data

---------------------------
Packet Type:    User ID Packet
Length: 39
User ID:        "Vera Heinau <heinau@zedat.fu-berlin.de>"
[...]
```

pgpacket läßt sich auch auf die geheimen PGP-Schlüssel anwenden; damit hat man also ein Werkzeug an der Hand, mit dessen Hilfe sich z.B. – Zugriff auf den Private-Key vorausgesetzt – bei Bedarf die Zahlen auslesen und anzeigen lassen, aus denen der geheime Schlüssel zusammengesetzt ist. Dafür muß zuerst die *Passphrase* durch einen Leerstring ersetzt (‘pgp -ke *keyid* secring.pgp’) und anschließend dieser so entstandene – **ungeschützte!** – geheime Schlüssel extrahiert werden. Danach läßt er sich ebenso wie ein öffentlicher Schlüssel mit pgpacket analysieren:

```
$ pgpacket2 -vau="./pubring.pgp" secuni

---------------------------
Packet Type:    Secret Key Packet
Length: 472
Version Byte:   3
Key Created:    20 Mar 1999  19:49:00
Valid for       195 days
Algorithm:      1 (RSA)
N:      0xBA8533C0055E5DB332AEF85A11AFF43143407011DC76B96F83E28D30E61D9230F529277...
E:      0x11
Protection Algorithm:   0 (None)
D:      0x2666B7D4B5CFA9E12105E7D64EEF8519337E3530DA90E9F14FDBD1C64D7E8F0A145B4BE...
P:      0xC9FD611381A598ADD6AF808497D5EE4A0710E5635B2FDF6020DFEDC970440C7EEB1D2C0...
Q:      0xEC64E84A223C6B8AC91BAD586CC7782712626894D67BE76CAFC948C35397D41835D94D1...
U:      0xA27E8EDE002F695BEB438AA8D6EFAD5D49B9C847E1CB81014C74354F2361835BF394ECE...
Checksum:       0xAB9E

---------------------------
Packet Type:    User ID Packet
```

```
Length: 91
User ID:        "UNI-CA, SoSe 1999 Certification Key <http://ca.UNI.de>...
```

H.2.2 `pgpsort`

Zur Handhabung und „Pflege“ der PGP 2.6-Keyring-Dateien – das könnte vor allem für eine CA wichtig werden, wenn ihr Public-Keyring sehr groß geworden ist – kann man sich eines Tools bedienen, das zu der PGP-Distribution dazugehört: Das Programm stammt von Ståle Schumacher, dem Administrator der „internationalen“ PGP-Homepage www.pgpi.com, heißt **pgpsort** und findet sich im Unterverzeichnis ‘contrib/pgpsort/’ der PGP 2.6-Distribution.

pgpsort sortiert ganze Keyrings neu, z.B. nach dem Erstellungsdatum der Schlüssel, deren Länge, der numerischen KeyID oder der (textuellen) UserID. Es kann nebenbei auch noch defekte Schlüssel aus dem Schlüsselbund entfernen, die PGP anderenfalls zu ständigen akustischen Warnmeldungen veranlassen.
Beispiel:

```
$ pgpsort +d pubring.pgp
Keyring 'pubring.pgp' sorted on date.

$ pgp -kv
Pretty Good Privacy(tm) 2.6.3in - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1997-10-07
International version - not for use in the USA. Does not use RSAREF.
Current time: 1999/02/23 00:18 GMT

Key ring: 'pubring.pgp'
Type Bits/KeyID    Date       User ID
pub   768/2246421D 1995/07/18 Heiko Schlichting <heiko@cis.fu-berlin.de>
                              Heiko Schlichting <heiko@chemie.fu-berlin.de>
                              Heiko Schlichting <heiko@fu-berlin.de>
pub  1024/00F0D5E9 1995/08/29 Vera Heinau <heinau@cis.fu-berlin.de>
                              Vera Heinau <heinau@zedat.fu-berlin.de>
                              Vera Heinau <heinau@chemie.fu-berlin.de>
                              Vera Heinau <heinau@fu-berlin.de>
pub  2048/FE93EAB9 1997/04/16 DFN-User-CA, CERTIFICATION ONLY KEY (Low Level: 1997-1998)
pub  2048/4EB02EDD 1998/07/30 ZIB-CA, Certification Only Key <http://www.zib.de/ca/>
pub  2048/C72929BD 1998/07/30 ZIB-CA, Encryption Key <ca@zib.de>
pub  2048/F7E87B9D 1998/12/29 DFN-PCA, CERTIFICATION ONLY KEY (Low-Level: 1999-2000)
pub  1024/06753A4D 1999/03/20 UNI-CA, SoSe 1999 Certification Key <http://ca.UNI.de>
           Expire: 1999/10/01 SIGNature only
pub  1024/A6A81269 1999/03/20 UNI-CA, SoSe 1999 Communication Key <ca@UNI.de>
           Expire: 1999/10/01 ENCRyption only
9 matching keys found.
```

Die Schlüssel sind nun aufsteigend nach ihrem Generierungsdatum (Spalte ‘Date’) sortiert und liegen in dieser Reihenfolge im “pubring.pgp” vor. – Diese Funktionalität könnte beispielsweise nützlich sein, wenn für eine Datenbank alle Schlüssel eines Keyrings in nach ihrer KeyID geordneter Reihenfolge benötigt werden, z.B. weil sie dann schneller in die Datenbank einsortiert werden können oder weil dann ein schnellerer Such-Zugriff möglich ist.

Anhang I

Hürden bei der Zertifizierung

I.1 Probleme mit X.509-Software

Trotz der Standardisierung des Zertifikatformates in der ITU-Recommendation X.509 bzw. dem ISO/IEC-Standard ISO/IEC 9594-8 verhalten sich Programme, die diesen Standard unterstützen, gelegentlich unerwartet oder uneinheitlich. Das hängt häufig nicht so sehr mit dem Zertifikatformat selbst, sondern mehr mit den komplexen Abläufen bei deren Verifikation zusammen. So ist beispielsweise vom Microsoft Internet Explorer bekannt, daß sich verschiedene Versionen unterschiedlich verhalten, wenn die Gültigkeitsdauer der Schlüssel bzw. Zertifikate übergeordneter CAs sich überlappen und nicht im Gültigkeitszeitraum des Schlüssels der jeweils nächsthöheren Zertifizierungsstelle enthalten sind.

Andere Probleme haben teilweise fast schon banale Ursachen, die aber für den Außenstehenden schwer bis gar nicht zu erkennen sind [Cam98b]. Die folgenden Screenshots mögen das belegen:

(a) NS-Navigator/Linux

(b) NS-Navigator 3.0/Windows 95

(c) NS-Navigator 2.02/Windows 95

(d) Microsoft Internet Explorer 3.0

Abb. I.1: Browser-Fehlermeldungen bei 2048 bit-Server-Schlüssel

Sie zeigen Fehlermeldungen verschiedener Versionen von Netscapes Navigator und Microsofts Internet Explorer, die durch einen zu *langen* CA-Schlüssel hervorgerufen wurden. Beide Programme verwenden in ihren Versionen kleiner als 4.0 eine Langzahl-Bibliothek, die nicht auf CA-Schlüssel von beispielsweise 2048 bit Länge ausgelegt ist und folglich einen Fehlercode zurückliefert, wenn ein solcher Schlüssel von einem Server übermittelt wird.

Im Benutzerinterface dieser Browser wird dann offenbar nur getestet, ob (irgend)ein Fehler aufgetreten ist, und wenn ja wird die – in diesem Fall völlig irreführende – Standard-Fehlermeldung ausgegeben. Das Phänomen kann mit entsprechenden alten Versionen der beiden Browser nachvollzogen werden, indem z.B. der Server `https://security.iks-jena.de` kontaktiert wird. Dieser Server verwendet einen 2048 bit-Schlüssel.

Dies ist gerade für Zertifizierungsstellen umso gravierender, als zumindest einige dieser Fehlermeldungen fälschlicherweise den Eindruck erwecken, mit dem Zertifikat des Servers wäre etwas nicht in Ordnung; hier wirkt sich also die Beschränktheit der in den Programmen verwendeten Langzahl-Bibliothek zu Lasten einer u.U. völlig ordnungsgemäß und sorgfältig arbeitenden Zertifizierungsstelle aus und könnte deren Reputation – obgleich in der Sache völlig unbegründet – schweren Schaden zufügen. Gerade in der Aufbauphase einer Zertifizierungsstelle könnte sich dies verheerend auswirken und die betreffende Stelle von vornherein in den Augen der Anwender diskreditieren!

WWW-Software (Browser- und sogar Server-Software) bietet dem Anwender oder Administrator häufig keinerlei Möglichkeit, sich den eigenen Public-Key oder dessen Fingerprint anzeigen zu lassen. Das macht es bei einem Zertifizierungsantrag für den betreffenden Schlüssel schwierig, diese Information mit anzugeben. Ohne diese Angabe ist aber die eindeutige Zuordnung zwischen einem Zertifizierungsantrag und dem elektronisch übermittelten *certificate request* mit darin enthaltenem Public-Key nicht möglich, so daß die Zertifizierungsstelle sich mit solchen vagen und unsicheren Anhaltspunkten wie der zeitlichen Koinzidenz behelfen muß. (Dies ist einer der Punkte, die die Ausstellung von X.509-Zertifikaten unnötig erschweren.) Ein *workaround* kann die Berechnung einer kryptographischen Prüfsumme sein, wenn zumindest eine lokale Kopie des Certificate Requests vorliegt (siehe H.1).

Die Erfahrung zeigt, daß bei *neuen* Browser-Versionen (oder auch allgemein neuen Versionen eines Programms) Vorsicht angebracht ist: Zwar werden oft bekannt Sicherheitslücken oder Fehler aus alten Versionen darin behoben, aber die neue Version enthält u.U. neuen, bislang noch nicht entdeckte Fehler. Dies trifft insbesondere auf den Bereich Zertifikat-Handhabung zu, in dem es immer wieder neue, überraschende Erfahrungen zu machen gibt.

I.2 Probleme mit PGP

Der gravierendste Fehler zuerst: PGP zeigt auch Zertifikate von solchen Schlüsseln weiterhin als „gültig" an, die widerrufen worden sind. Im nachfolgenden Beispiel ist der Key von Uta Ungültig widerrufen worden.

```
$ pgp -kv uta
[...]
```

```
Key ring: 'pubring.pgp', looking for user ID "uta".
Type Bits/KeyID    Date       User ID
pub   399/BD7054C9 1998/05/25 *** KEY REVOKED ***
                              Uta Ungueltig <uta@nil.org>
1 matching key found.
```

Uta hatte zuvor den Schlüssel von Ingmar Camphausen signiert. Ruft man nun nach dem Schlüsselwiderruf von Uta Ungültig die Zertifikatsprüfung für den Schlüssel von Ingmar auf, so sieht die Ausgabe von PGP so aus:

```
$ pgp -kc ingmar
[...]
Key ring: 'pubring.pgp', looking for user ID "ingmar@i".
Type Bits/KeyID    Date       User ID
pub  1024/4F570BA3 1993/06/18 Ingmar Camphausen <ingmar@in-berlin.de>
sig!      BD7054C9 1998/05/25  Uta Ungueltig <uta@nil.org>
[...]
```

Es wird also in keiner Weise davor gewarnt, daß der Schlüssel selber, mit dem die Unterschrift unter Ingmars Key erzeugt worden ist, widerrufen worden ist! (Diese Information wäre ja bei der Einschätzung der Signatur von Uta durchaus wichtig.)

Weiteres PGP-Fehlverhalten, u.a. die mangelhafte Plausibilitätsprüfung der in PGP-Schlüsseln, -Signaturen und -Zertifikaten vorliegenden Zeitstempel, ist in [CD98] dokumentiert. *Diese beiden Schwächen sind in der Version PGP2.6.3**in** behoben.*

PGP 2-Versionen kommen mit den Schlüsseln im neuen Format, das ab PGP 5 verwendet wird (d.h. DSS/ElGamal-Keys), nicht klar und geben die Fehlermeldung “unknown packet format” aus, wenn versucht wird, einen solchen Schlüssel dem Public-Keyring hinzuzufügen.

PGP 2.6.x gibt keine Fehlermeldung aus, wenn der Public-Keyring schreibgeschützt ist und versucht wurde, einen neuen Schlüssel hinzuzufügen oder einen bereits vorhandenen zu zertifizieren (also *schreibend* auf den Public-Keyring zuzugreifen). Es entsteht der Eindruck, man hätte erfolgreich einen neuen fremden Schlüssel zertifiziert und ihn dem Public-Keyring hinzugefügt, der Ablauf ist auch völlig identisch, die Änderungen konnten aber am Ende nicht abgespeichert werden.

Ein Problem, das mit PGP unter UNIX auftritt: die Diskette mit dem Secret-Keyring wird versehentlich aus dem Laufwerk entnommen, aber das darauf befindliche Dateisystem nicht ungemountet. Aufgrund der I/O-Buffer des UNIX-Filesystems macht sich das Fehlen des Mediums u.U. nicht gleich bemerkbar, es kommt aber zu Inkonsistenzen zwischen dem Abbild des Dateisystems in den Puffern im Hauptspeicher und dem tatsächlichen Aussehen auf dem Speichermedium.

Ähnliche Problem treten bei PGP 5 auf: Die Software (UNIX-Version) arbeitet nicht mit schreibgeschützten Secret-Keyringen. Das ist besonders ungünstig, da diese Datei so gut wie möglich und mit allen zur Verfügung stehenden Mitteln vor unbefugter Kenntnisnahme, aber auch vor Manipulationen geschützt werden sollte und man sie daher sinnvollerweise hardware-mäßig schreibschützen sollte, wenn das Speichermedium es zuläßt (und das ist ja bei Wechselmedien häufig der Fall).

Manche RSA-Schlüssel, die mit älteren PGP-Versionen (PGP 2.6) erzeugt worden sind, bereiten beim Import in die neuen PGP-Versionen Probleme und lassen sich nicht richtig einbinden. (Die

DFN-PCA-Mitarbeiter haben noch nicht herausgefunden, welche Eigenschaft alle problematischen Schlüssel gemeinsam haben.)

PGP 2.6 und PGP 6.5.1 verhalten sich auch beim Verifizieren mancher Signaturen unerwartet unterschiedlich: Eine Prüfung der separaten Signatur[1] zu der Archiv-Datei, die Version 2.6.0 des WU-FTP-Servers enthält[2], liefert bei PGP 6.5.1 ein “Good Signature” und bei PGP 2.6.2 und 2.6.3in ein “Bad Signature”.

[1] `ftp://ftp.cert.dfn.de/pub/tools/net/wuarchive-ftpd/wu-ftpd-2.6.0.tar.gz.asc`
[2] `ftp://ftp.cert.dfn.de/pub/tools/net/wuarchive-ftpd/wu-ftpd-2.6.0.tar.gz`

Anhang J

Aufwandsabschätzung

Schlimm sind die Schlüssel,
die nur schließen auf, nicht zu;
Mit solchem Schlüsselbund
im Haus verarmest Du.
— RÜCKERT, *Weisheit des Brahmanen,* nach [Zoo54, S. 682]

Eine wichtige Rolle bei der Entscheidung für oder gegen die Einrichtung einer Zertifizierungsstelle bzw. bei der Entscheidung, diese Dienstleistung bei einem externen Anbieter einzukaufen oder sie selber anzubieten, spielt – nicht nur bei Signaturgesetz-konformen Zertifizierungsstellen – der personelle und finanzielle Aufwand, der damit einhergeht.

An dieser Stelle sollen daher einige Vergleichszahlen aus entsprechenden Projekten genannt und der Versuch unternommen werden, zumindest eine erste grobe Schätzung des für eine UNI-Zertifizierungsstelle zu erwartenden Aufwandes vorzunehmen.

Eine Studie der Giga-Group für den Anbieter von Zertifizierungssoftware Entrust Technologies [Mac98] nennt für einen 5-Jahres-Zeitraum folgende *Enterprise Support Costs* für die Versorgung mit *Basic Certificates for Web Authentication*, also für Identitätszertifikate, wie sie auch die UNI-CA ausstellen soll:

Nutzer	prognostizierte Kosten
5 000	ca. 600 000 US-Dollar
20 000	ca. 850 000 US-Dollar (Inhouse-Lösung) bis 1,3 Mill. US-Dollar (Out-Sourcing)

Interessant dabei ist, wie sich diese Kosten aufteilen: Von den 600 000 US-Dollar für die Versorgung von 5 000 Nutzern mit Zertifikaten kalkuliert die Giga-Group-Studie für Lizenzen, Hardware und Software ca. 50 000 US-Dollar, für Installation, Wartung, Support, Zertifizierung und den Betrieb hingegen 550 000 US-Dollar.

Daß die in der Giga-Group-Studie genannten Zahlen durchaus realistisch sind, belegen die Erfahrungswerte aus der Einrichtung einer Zertifizierungsstelle für die ScotiaBank Inc., Toronto: Dort wurde mit einem Etat von zwei Millionen Dollar eine firmeneigene CA etabliert, die inzwischen 40 000 Zertifikate ausgestellt hat [Bru98].

Aufbau

Die Aufbauphase umfaßt, ggf. ausgehend vom vorliegenden Konzept, die Auswahl und Beschaffung geigneter Hardware, die Installation, Konfiguration und den Test von Betriebssystem, Treibern, zusätzlichen Peripheriegeräten und Zertifizierungssoftware. Weiterhin den Aufbau des WWW- und FTP-Servers für die UNI-CA sowie die Erstellung oder Beschaffung entsprechender Dokumentation, Vordrucke usw.

Zeitlicher Aufwand

Einarbeitung in das Konzept, Verfeinerung/Anpassung, Abstimmung	20 h
Hardware- und Software-Beschaffung, -Installation	90 h
Dokumentation (intern und extern), WWW-Server	120 h
FTP-Server, Zusammenstellung der Anwender-Software	30 h
Integration in die RZ-Arbeitsabläufe	10 h
Probeläufe, interne Schulung der Mitarbeiter	30 h
	300 h

Materialkosten

Laptop	4 000 – 15 000 DM
ZIP o. LS120-Laufwerk	600 DM
zweiter Akku	500 DM
zweite Festplatte	800 DM
SCSI-Controller	500 DM
Diebstahlschutz	200 DM
portabler Drucker	700 DM
Schutzschrank, -schlüssel	500 DM
Wechselmedien, Magnetbänder	250 DM
Papier, Schilder, Vordrucke, ...	250 DM
	ca. 8 – 19 TDM

Regelbetrieb Anlaufphase

Während der ersten Zeit nach der Einrichtung und offiziellen Bekanntgabe des neuen Services ‘Zertifizierung’ wird die Nachfrage vermutlich noch relativ gering ausfallen. Eventuell werden sich zunächst diejenigen melden, die schon jetzt intensive PGP-Nutzer sind, doch nach diesem möglichen ersten „Mini-Ansturm“ ist damit zu rechnen, daß es nur gelegentlich Nachfragen nach der Zertifizierung geben wird.

Einen nicht unerheblichen Teil der Zeit, der rund um die Zertifizierungstätigkeit aufgewendet werden muß, dürfte während dieser Phase durch das Einüben der Abläufe, Anpassen von Handlungsleitfäden und Anleitungen an die ersten praktischen Erfahrungen usw. beansprucht werden. Auch die Aktivitäten, die dem Ziel dienen, den neuen Service innerhalb der UNI bekanntzumachen, dürften in dieser Phase mehr Zeit in Anspruch nehmen als die Zertifizierungstätigkeit selbst.

Zeitlicher Aufwand

- ca. 5 Minuten bei der Entgegennahme jedes Zertifizierungsantrages nebst Identitätsprüfung
- nochmals etwa 10 bis 20 Minuten für die Zertifizierung des Schlüssels, das Veröffentlichen des Zertifikates und die Benachrichtigung des Zertifikatnehmers, wenn alle diese Schritte manuell und noch ohne die Unterstützung z.B. entsprechender Skripte oder Utilities ausgeführt werden (bei X.509-Schlüsseln eher etwas länger)

Bei einer geschätzten Nachfrage von zehn bis 20 Interessenten pro Woche wäre also mit einem Arbeitsaufwand von wöchentlich rund vier Stunden zu rechnen (zeitlichen „Leerlauf“ durch Wartezeiten aufgrund geringer Nachfrage z.B. während einer Nutzer-Sprechstunde nicht eingerechnet).

In den Leitlinien [ABR97, S. 21 f.] zum SPHINX-Projekt (siehe 3.4.1.2), das CAs in einigen Bundesbehörden vorsieht, rechnen die Autoren mit folgendem Aufwand für eine CA: bei kleinen CAs je eine halbe Stelle für allgemeine Administration und Organisation sowie für die technische Betreuung und Wartung. Für eine große CA wird mit bis zu zwei Stellen für Administration und Organisation und eine Stelle für die Technik gerechnet. Hinzu kommt jeweils ein Nutzer-abhängiger Aufwand von rund 30 Minuten pro Teilnehmer und Jahr bei nicht-automatisiertem Betrieb.

Materialkosten

In dieser Phase fallen so gut wie keine Materialkosten durch die Zertifizierungstätigkeit an, außer vielleicht den Kosten für Disketten, falls die eigenen Schlüssel auf diesem Weg an die Benutzer ausgegeben werden.

Vollausbau

Für eine „Vollversorgung“ aller UNI-Angehörigen mit Zertifizierungsdiensten und entsprechend breiter Nachfrage sieht die Kalkulation anders aus:

Es ist zum einen zu erwarten, daß diese Situation nicht von heute auf morgen eintritt, sondern daß die UNI-CA-Mitarbeiter bis dahin eine gewisse Routine in den üblichen Aufgaben und Abläufen entwickelt haben. Wenn die Nachfrage ein bestimmtes Maß übersteigt, werden sich die CA-Administratoren vermutlich auch damit befassen, wie die Arbeitsabläufe effizienter gestaltet werden können und ob nicht durch die Automatisierung eines Teils der Arbeit (Schlüssel vom Benutzer anfordern, falls er noch nicht vorliegt; Zusenden des Zertifikates und der UNI-CA- und

DFN-PCA-Schlüssel nach erfolgter Zertifizierung usw.) der menschliche Arbeitsaufwand pro Zertifizierung reduziert, fehlerträchtige oder wenig herausfordernde Tätigkeiten an Software delegiert werden können. Besonders interessant dürften in dieser Hinsicht auch die Ergebnisse der Diplomarbeit von SIMON FRISCHEISEN sein [Fri99], in der es um die Software-Unterstützung für den organisatorischen Teil der Arbeit einer Zertifizierungsstelle geht.

Die durchschnittliche Bearbeitungszeit pro Zertifizierungsfall sollte sich dadurch auf insgesamt etwa fünf Minuten senken lassen.

Die bisherigen Schätzungen bezogen sich auf die Zertifizierung von Personenschlüsseln. Bei der Ausstellung von Server-Zertifikaten für den Betrieb eines HTTPS-Servers liegt der Aufwand aufgrund der vielen zu prüfenden und anschließend auch einzugebenden Daten um einiges höher; Erfahrungswerte aus anderen Zertifizierungsstellen besagen, daß hierfür mit einem Arbeitszeitaufwand von etwa einer halben Stunde gerechnet werden muß.

Für die in diesem Stadium des Ausbaus der UNI-CA sicher ebenfalls verstärkt nachgefragten Schulungs- und Beratungsdienstleistungen ist demgegenüber mit einem nochmals erheblich höheren Zeitaufwand pro Fall zu rechnen (Aussage eines kommerziell tätigen Zertifizierers: drei bis vier Stunden Beratungsaufwand pro Kunde).

Zeitlicher Aufwand

Fiktive 10 000 Nutzer und UNI-Mitarbeiter – wenn Public-Key-Verfahren erst einmal ubiquitär oder sogar unverzichtbar sind im Internet, wird jede Nutzerin und jeder Nutzer seine persönlichen Schlüssel zertifizieren lassen wollen (oder müssen) – wollen mit Zertifikaten für ihre persönlichen Schlüssel versorgt werden. (Bei konstanter Größe der Studierenden- und Mitarbeiterzahlen dürfte irgendwann ein Gleichgewichtszustand erreicht sein, in dem sich die Zahl der Uni-Abgänger und ausscheidenden Mitarbeiter, deren Schlüssel zertifiziert waren, etwa die Waage hält mit der Zahl derjenigen, die neu an die UNI kommen und diesen Dienst in Anspruch nehmen.)

Da die Gültigkeit einer Zertifizierung zeitlich begrenzt ist, besteht wegen der erforderlichen Re-Zertifizierung selbst bei ansonsten völlig gleichbleibendem Studenten- und Personalbestand regelmäßig Bedarf nach Diensten der Zertifizierungsstelle.

Es ist zu hoffen, daß, bis dieser Nachfrage- bzw. Ausbauzustand der UNI-CA erreicht ist, die Gültigkeitsdauer bei allen Zertifikaten „gleitend“ und nicht an einen Stichtag gebunden festgelegt werden kann. Dadurch würde sich die Arbeit entzerren, und es wäre eine relativ gleichmäßige Verteilung der Arbeitsbelastung über das ganze Jahr möglich. (Anders heute: zumindest bei der PGP-Zertifizierung in der vorgeschlagenen Form würden zu ein oder zwei Stichtagen im Jahr die CA-Schlüssel und mit ihnen die Zertifikate ablaufen, so daß auf einen Schlag eine große Anzahl von Re-Zertifizierungen zu bewältigen wäre.) Doch selbst unter dieser Prämisse wären pro Monat durchschnittlich knapp 1 000 (Re-)Zertifizierungen abzuwickeln – allein für die Personen-Zertifikate (Server-Zertifizierung nicht mitgerechnet), d.h. pro Arbeitstag etwa 50 Anträge zu bearbeiten. Selbst unter der optimistischen Annahme, daß pro Antrag insgesamt nur etwa fünf Minuten Bearbeitungsaufwand anfielen, hieße das, daß nur für die Zertifizierung selbst bereits eine Teilzeit-Stelle eingeplant werden müßte (Schulungen, Wartungsaufwand und Operating der Zertifizierungsstelle einmal völlig außen vor). Setzt man hingegen die 30 Minuten pro Teilnehmer an, die beim SPHINX-Projekt zugrundegelegt wurden (s.o.), so wären alleine für die Zertifizierung drei Vollzeitkräfte erforderlich (zzgl. der dann sicher mindestens drei Mitarbeiter für Organisation und Technik). Exakter abschätzen lassen werden sich diese Zahlen erst, wenn konkrete Erfahrungen aus den ersten Monaten Betrieb der UNI-CA vorliegen und die unvermeidlichen Anlaufprobleme überwunden sind.

Bei dieser Abschätzung wurde vorausgesetzt, daß die Zertifizierung weiterhin zentral nur in der UNI-CA im Rechenzentrum erfolgen würde. Bei Nachfrage nach Zertifizierungsdiensten in dieser Größenordnung drängt es sich aber geradezu auf, die Möglichkeit von Registrierungs- oder eigenen Zertifizierungsstellen in einzelnen Fachbereichen oder Verwaltungseinheiten der Universität zu nutzen und so zu einer Entzerrung und besseren zeitlichen und räumlichen Verteilung der Nachfrage zu gelangen. Die Arbeit in jeder einzelnen der UNI-Zertifizierungsstellen ist dann entsprechend geringer, andererseits fällt zusätzliche Arbeit durch die erforderliche Kommunikation bzw. Abstimmung zwischen den beteiligten Stellen an, beispielsweise zwischen der UNI-CA und den Außenstellen, die die Registrierung und Identitätsprüfung vornehmen.

Kosten

Noch bevor die Nachfrage nach Zertifizierung ein solches Ausmaß erreicht haben wird, dürfte die in diesem Konzept vorgeschlagene initiale Geräteausstattung nicht mehr ausreichend sein. Zumal bei einer solchen Durchdringung aller anderen Dienste mit Public-Key-Verfahren auch ganz andere Anforderungen hinsichtlich der Verfügbarkeit als die für die Startphase der UNI-CA zugrundegelegten gestellt würden.

Wenn die Public-Key-Verfahren erst einmal so sehr Einzug in den Arbeitsalltag der Wissenschaftler und Verwaltungsmitarbeiter gehalten haben, würde bei einer „zentralistischen" Lösung (nur eine universitätsweite Zertifizierungsstelle) vermutlich nicht mehr mit tragbaren Computern gearbeitet, sondern dieser hohe Aufwand und die hohen Anforderungen an Sicherheit und Verfügbarkeit der Zertifizierungsdienste würden die Nutzung eines eigenen zugangskontrollierten CA-Rechnerraumes nahelegen, in dem der Zertifizierungsrechner betrieben würde. Dadurch entstünden mindestens entsprechende einmalige Kosten für die Einrichtung und Absicherung dieses Raumes und eventuell erforderliche neue Hardware.

Doch auch bei der Alternative, einem eher dezentralen Ansatz mit mehreren oder sogar vielen Zertifizierungs- oder Registrierungsstellen in den UNI-Einrichtungen würden Hardware-Kosten durch die Zertifizierungsrechner entstehen, die dann an jedem der Standorte gebraucht würden.

Noch ist eine so große Nachfrage nicht vorhanden, wenn es aber einmal soweit ist, dann existieren bis dahin vermutlich auch Firmen, die sich auf entsprechende Dienstleistungen spezialisiert haben. Bei Nutzerzahlen von mehreren Zehntausend Zertifikatnehmern könnte es sich dann vielleicht auch rechnen, die entsprechenden Dienste extern einzukaufen.

Anhang K

Low-Level-Policy (Mustertext)

(Letzte Änderung dieser Policy: ...)

Vorbemerkung zu dieser Version

Dies ist die Version ... der Low-Level-Policy der UNI-CA. Diese Version ist [ein Entwurf | vorläufig | ...] und beruht auf der Low-Level-Policy der DFN-PCA, Version 1.2. Sie soll den Anforderungen der Low-Level DFN-Policy an eine CA gerecht werden.

1 Einleitung

Dieses Dokument enthält die Low-Level Zertifizierungsrichtlinien (die sog. *Policy* der Zertifizierungsstelle der UNI (nachfolgend auch UNI-CA genannt). Dabei handelt es sich um Zertifizierung mit moderaten Sicherheitsanforderungen an die Zertifizierungsstelle, bei der Ausstellung der Zertifikate und an die Zertifikatnehmer.

Die UNI-CA wird betrieben als Teil der DFN-Zertifizierungshierarchie (siehe 3.2) im Sinne der Low-Level-Policy der DFN-PCA [DFN-PCA]. Die UNI-CA macht dabei Gebrauch von der in Abschnitt 5.1 „Regeln für die Zertifizierung von CAs“ der *DFN-PCA Low-Level Policy* einer Zertifizierungsstelle ausdrücklich zugestandenen Möglichkeit, über die DFN-Policy hinausgehende Richtlinien bei Bedarf in einer eigenen Policy festzulegen. Es gelten also für die Arbeit der Low-Level UNI-CA die DFN-PCA Low-Level Zertifizierungsrichtlinien, verschärft bzw. verfeinert durch die in diesem Dokument genannten zusätzlichen Vorgaben.

Der Sinn dieses Dokumentes ist es, Benutzern (*relying parties*) die Einschätzung der durch die Low-Level UNI-CA ausgestellten Zertifikate zu ermöglichen.

Die Policy der ebenfalls von der UNI-CA betriebenen Medium-Level Zertifizierungsstelle mit höheren Sicherheitsanforderungen als denen der Low-Level UNI-CA ist in einem separaten Dokument beschrieben.

Die in diesem Dokument getroffenen Aussagen sind für die Arbeit der Low-Level UNI-CA bindend. (Dies gilt nicht für diejenigen Teile, die gesetzlichen Vorschriften widersprechen. Derartige Widersprüche sind nicht beabsichtigt; sollten sie aber dennoch auftreten, so verlieren dadurch die übrigen Vorgaben dieser Policy nicht ihre Gültigkeit.) Die Low-Level UNI-CA zertifiziert ausschließlich nach den Richtlinien dieser Policy.

2 Identität der UNI-CA

Adresse UNI-CA
Zertifizierungsstelle der Universität ...
Zentrales Rechenzentrum
Musterstraße 77

D-99999 ....
Germany
Telefon: (+49) ...
Telefax: (+49) ...

E-Mail-Adresse

`ca@UNI.de`

Allgemeine Informationsdienste der UNI-CA

FTP-Server: `ftp://ca.UNI.de`

WWW-Server: `http://ca.UNI.de` bzw. `https://ca.UNI.de`

Auf diesen Servern finden Sie die Wurzelzertifikate der DFN-PCA, die Zertifikate der DFN-PCA für die UNI-CA, die Schlüssel zur vertraulichen Kommunikation mit den UNI-CA-Mitarbeitern sowie weitere Informationen zur UNI-CA und zum Projekt DFN-PCA.

Gültigkeit dieses Dokumentes

... bis ... [Alternativ: ab ...]

3 Zuständigkeitsbereich der UNI-CA

Der Zuständigkeitsbereich der UNI-CA umfaßt die Einrichtungen der UNI und deren Angehörige sowie deren Projekt-Partner. Das Ziel der UNI-CA besteht darin, den UNI-Angehörigen eine Infrastruktur zur Verfügung zu stellen, die sie bei der vertrauenswürdigen Kommunikation untereinander und mit den Angehörigen anderer DFN-Einrichtungen sowie mit Dritten unterstützt und umgekehrt Dritten die vertrauenswürdige Kommunikation mit UNI-Angehörigen möglich macht. Die Basis dafür bilden Dienste, die die Unverfälschtheit (Integrität), den Urheberschafts- bzw. Identitätsnachweis (Authentizität) und die Vertraulichkeit (Geheimhaltung des Inhaltes) von auf elektronischem Wege ausgetauschten Nachrichten gewährleisten.

Für die Zwecke der internationalen Kommunikation wird über die DFN-Zertifizierungshierarchie eine Anbindung an andere entsprechende Infrastrukturen nach Maßgabe der Möglichkeiten bereitgestellt.

Die Low-Level UNI-CA wird nur Zertifikate für Benutzer ausstellen, keine CA-Zertifikate.

3.1 Die DFN-Zertifizierungshierarchie

Das Projekt DFN-PCA sieht die Einrichtung einer Zertifizierungshierarchie innerhalb der DFN-Mitgliedseinrichtungen vor, an deren Spitze die DFN-PCA als Root-CA, also als oberste Zertifizierungsstelle steht. Die DFN-PCA ist erreichbar unter

DFN-PCA
Universität Hamburg
FB Informatik – RZ
Vogt-Kölln-Str. 30
D-22527 Hamburg

Telefon: (0 40) 428 83-2262
Telefax: (0 40) 428 83-2241
E-Mail: `dfnpca@pca.dfn.de` (PGP-Schlüssel erhältlich)

Die Zertifizierungshierarchie unterhalb der DFN-PCA besteht aus drei verschiedenen Einheiten (Zertifikatnehmern):

- Zertifizierungsinstanzen (CAs)
- optionalen Registrierungsinstanzen (RAs)
- Benutzern

Die internationale Anbindung der kompletten DFN-Zertifizierungshierarchie an andere Hierarchien kann durch eine gegenseitige Zertifizierung (Cross-Zertifizierung) der DFN-PCA mit anderen PCAs erfolgen. Die Eingliederung der DFN-Hierarchie in eine Internet-weite Infrastruktur kann erfolgen, sobald eine entsprechende Instanz (genannt *Internet PCA*

Registration Authority, IPRA) verfügbar ist. In jedem Fall werden die Teilnehmer der DFN-Hierarchie über internationale Anbindungen und Cross-Zertifizierungen informiert.

Das Ziel der DFN-Zertifizierungshierarchie ist es, pro wissenschaftlicher Einrichtung eine CA zu betreiben, welche direkt von der DFN-PCA zertifiziert wird. Für die UNI als DFN-Mitgliedseinrichtung ist dies die UNI-CA. Diese CAs haben die Möglichkeit, ihrerseits Zertifikate für Benutzer und untergeordnete Sub-CAs zu erteilen; das genaue Konzept zum Aufbau und Betrieb solcher CAs ist in [RFC 1422] (s. Literaturverzeichnis) erläutert.

Unterhalb der DFN-PCA operierende CAs haben weiterhin die Möglichkeit – durch den Vorgang der Cross-Zertifizierung mit anderen CAs –, eigene Verbindungen zu Zertifizierungsinstanzen bzw. -infrastrukturen von Einrichtungen herzustellen, welche nicht dem DFN-Verein angehören.

Der öffentliche Schlüssel der DFN-PCA ist in einem selbst-signierten Zertifikat (Wurzel-Zertifikat), ausgestellt durch die DFN-PCA, enthalten. Alle Teilnehmer der Infrastruktur erhalten dieses Wurzel-Zertifikat im Zuge der eigenen Zertifizierung und können somit die Authentizität und Gültigkeit aller unterhalb der DFN-PCA erteilten Zertifikate überprüfen.

Die Low-Level-PCA unterstützt auch den Einsatz des Programms PGP (“Pretty Good Privacy“). Obwohl in dem von PGP propagierten Vertrauensmodell (dem sog. “Web-of-trust”) der Einsatz von Zertifizierungsinstanzen nicht vorgesehen ist, sind die technischen Voraussetzungen zum Betrieb einer PGP-Zertifizierungsinstanz grundsätzlich gegeben, so daß derartige Dienste angeboten werden können.

Der Sinn von PGP-CAs ist die Einrichtung vertrauenswürdiger Instanzen, die durch ihre digitalen Signaturen die Bindung eines Public-Keys an einen Benutzer (bzw. dessen Benutzer-ID) herstellen. Die digitale Signatur einer PGP-CA soll auf diese Weise einem Public-Key ein größeres Maß an Vertrauenswürdigkeit geben, als durch die Signaturen beliebiger Benutzer erreicht werden kann.

3.2 Rechtliche Bedeutung

Eine Zertifizierung durch die UNI-CA ist keine Zertifizierung *im Sinne des Signaturgesetzes* [SigG]. Die UNI-CA erhebt nicht den Anspruch, eine Zertifizierungsstelle *im Sinne von § 2 Abs. 2 des Signaturgesetzes* zu sein.

Ein *Anspruch* auf die Erteilung eines Zertifikates durch die UNI-CA besteht nicht.

Die rechtliche Beweiskraft digitaler Signaturen ist derzeit noch offen. Der Sinn einer DFN-weiten Public-Key-Infrastruktur, deren Teil die UNI-CA ist, liegt in der Schaffung der technischen Voraussetzungen für eine gesicherte elektronische Kommunikation.

Insbesondere der DFN-Verein, die UNI sowie die Mitarbeiter der UNI-CA übernehmen keine Form der Gewährleistung. Alle Aufgaben werden von den UNI-CA-Mitarbeitern nach bestem Wissen und Gewissen durchgeführt.

4 Sicherheitsanforderungen

Durch die Teilnahme an einer Public-Key-Infrastruktur entstehen für alle Beteiligten bestimmte Anforderungen hinsichtlich der Sicherheit der eingesetzten Hard- und Software einerseits sowie des verantwortungsvollen Umgangs mit kryptographischen Schlüsseln andererseits. Die Anforderungen an die DFN-PCA und die CAs sind dabei höher als die an RAs oder Nutzer gestellten, da der Mißbrauch eines PCA-/CA-Schlüssels allen untergeordneten Zertifikaten die Vertrauenswürdigkeit entziehen würde und insofern viel gravierendere Auswirkungen hätte.

4.1 Sicherheitsanforderungen an die DFN-PCA

Die Vorgaben für die Low-Level DFN-PCA sind in den Low-Level Zertifizierungsrichtlinien der DFN-PCA [DFN-PCA] beschrieben.

4.2 Sicherheitsanforderungen an die UNI-CA

Die UNI-CA arbeitet nach folgenden Maßgaben:

- Für die Dienste der UNI-CA wird ein Rechner eingesetzt, der in geeigneter Weise vor mißbräuchlicher Benutzung geschützt ist. Der unbefugte Zugriff auf den CA-Rechner und eventuell gespeicherte Schlüsseldaten wird durch den Einsatz geeigneter Hard- und Software unterbunden. Es wird ein Rechner ohne jeglichen Netzwerkanschluß eingesetzt; dieser Rechner wird physikalisch geschützt aufbewahrt (Zugangs- / Zugriffskontrolle).
- Die UNI-CAs verwendet getrennte asymmetrische Schlüsselpaare zum Signieren und zum Entschlüsseln.
- Geheime Schlüssel der UNI-CA zum Erzeugen digitaler Signaturen müssen vor Mißbrauch durch Unbefugte geschützt werden und dürfen nicht weitergegeben werden. Die Verantwortung hierfür liegt bei den Administratoren der CA, die daher angehalten sind, vom CA-Rechner getrennte Speichermedien (z.B. SmartCard, Wechsel-Festplatte, Diskette) zur Speicherung der geheimen Schlüssel einzusetzen, soweit dies technisch möglich ist. Der Zugriff auf diese geheimen CA-Schlüssel wird in jedem Fall durch Paßworte bzw. PINs geschützt, die nur den CA-Administratoren bekannt sind und niemals im Klartext abgelegt oder notiert werden.
- Mit dem geheimen Signatur-Schlüssel der UNI-CA werden ausschließlich Benutzer-Schlüssel, Zertifikat-Widerrufe oder die Policy der UNI-CA unterschrieben. Der geheime Signatur-Schlüssel wird nicht für Kommunikationszwecke verwendet.
- Asymmetrische Schlüsselpaare der UNI-CA zur Erzeugung von Signaturen weisen eine Mindestlänge von 1024 Bits auf. Aus Kompatibilitätsgründen dürfen sie zugleich nicht länger als 2047 Bits sein
- In solchen Fällen, in denen die Low-Level UNI-CA asymmetrische Schlüsselpaare für die Nutzer erzeugt, müssen nach der Erzeugung und Schlüsselübergabe an den Nutzer auf Seiten der UNI-CA alle Kopien des geheimen Schlüssels des Nutzers oder Bestandteile oder Zwischenergebnisse aus der Schlüsselgenerierung, aus denen der geheime Nutzer-Schlüssel ermittelt werden könnte, endgültig gelöscht werden.
- Sämtliche persönlichen Daten der Zertifikatnehmer, die den UNI-CA-Mitarbeitern bei der Zertifizierung über die Schlüssel- und Zertifikatdaten hinaus bekannt werden (z.B. die Personalausweisnummer), werden von den UNI-CA-Mitarbeitern vertraulich behandelt und ausschließlich zu internen Dokumentationszwecken erhoben; sie werden nicht veröffentlicht.

4.3 Sicherheitsanforderungen an Benutzer

Benutzer im Sinne dieser Policy sind natürliche Personen, die die Zertifizierungsdienste der UNI-CA in Anspruch nehmen (Zertifikatnehmer). Folgende Anforderungen werden an die Zertifikatnehmer der UNI-CA gestellt:

- Der geheime Schlüssel des Benutzers muß ausreichend vor Mißbrauch durch Unbefugte geschützt und darf nicht weitergegeben werden; hierfür ist jeder Benutzer selbst verantwortlich. Werden keine SmartCards zum Speichern des geheimen Schlüssels eingesetzt, ist der Zugriff auf den geheimen Schlüssel des Benutzers durch ein Paßwort bzw. eine PIN zu schützen. Weder die optionale SmartCard noch das Paßwort bzw. die PIN dürfen an andere Benutzer oder CA-Administratoren weitergegeben werden. Der Benutzer wird hierauf bei der Zertifizierung ausdrücklich hingewiesen.
- Der zu zertifizierende Schlüssel des Benutzers muß eine Mindestlänge von 1024 Bits aufweisen. Abweichend davon dürfen auch Schlüssel zertifiziert werden, die kürzer als 1024 Bits, jedoch mindestens 768 Bits lang sind, *sofern* sie ausweislich des entsprechenden Zeitstempels im PGP-Public-Key bereits *vor* [dem Inkrafttreten dieser Policy | dem ...] erzeugt wurden. Ab dem 1.1.2001 werden keine Schlüssel mehr zertifiziert, die kürzer als 1024 Bits sind.

5 Zertifizierungsregeln

Die Low-level UNI-CA führt keine Cross-Zertifizierungen mit anderen Zertifizierungsstellen durch. Sie zertifiziert keine nachgeordneten CAs und auch keine Registrierungsstellen (RAs), sondern ausschließlich Nutzer, genauer: deren öffentliche Schlüssel. Die einzige Ausnahme von dieser Regel bilden der Zertifizierungs- und der Kommunikationsschlüssel der Low-Level UNI-CA selbst: Sie werden mit einem Selbst-Zertifikat versehen.

Dieser Abschnitt beschreibt technische und organisatorische Richtlinien und Prozeduren, die bei einer Zertifizierung von Benutzern zu beachten sind.

Anonyme oder pseudonyme Zertifikate werden von der UNI-CA *nicht* ausgestellt. Ebenso werden keine Zertifikate für Gruppenschlüssel ausgestellt (einzige Ausnahme: der Kommunikationsschlüssel der UNI-CA selbst; er ist nicht an eine einzige Person gebunden, sondern wird von allen UNI-CA-Mitarbeitern zum Lesen von verschlüsselt an die UNI-CA geschickten Nachrichten verwendet).

In keinem Fall werden Zertifizierungswünsche *automatisiert* bearbeitet; es muß sichergestellt sein, daß die Zertifizierung erst nach einer Interaktion durch einen der CA-Mitarbeiter erfolgt.

5.1 Unterstützte Schlüsselformate

Die Low-Level UNI-CA zertifiziert bis auf weiteres ausschließlich PGP-*RSA*-Schlüssel. DH/DSS-Schlüssel, wie sie z.B. von PGP 5 erzeugt werden, werden von der UNI-CA *nicht* zertifiziert.

5.2 Schlüsselgenerierung

Ein Benutzer, der zertifiziert werden möchte, generiert zunächst (oder besitzt schon) ein persönliches asymmetrisches Schlüsselpaar und übermittelt anschließend den selbst-signierten PGP-Public-Key) per E-Mail oder mittels eines Datenträgers an die UNI-CA. Wenn der Nutzer dies wünscht und die CA dies anbietet, wird das asymmetrische Schlüsselpaar des Benutzers auch von der CA erzeugt. Dabei sind von der UNI-CA die in Abschnitt 4.2 beschriebenen Sicherheitsanforderungen einzuhalten.

5.3 Namenswahl (Benutzerkennung)

PGP-Benutzer werden durch eine Benutzer-ID identifiziert, die aus einer beliebigen Kombination von ASCII-Zeichen bestehen kann. Bei der Benutzung von PGP kann zunächst jeder Teilnehmer die Benutzer-ID frei wählen. Um dennoch ein sinnvolles und korrekt funktionierendes Zertifizierungsmodell zu erreichen, muß die Bindung eines PGP-Public-Keys an einen Benutzer sichergestellt sein. Daher werden an zu zertifizierende PGP-Benutzer-IDs folgende Anforderungen gestellt:

Benutzer-IDs müssen eine eindeutige Bindung zwischen dem Public-Key und der Identität des Benutzers herstellen. Daher muß der Vor- und Zuname des Benutzers, wenn möglich auch eine E-Mail-Adresse, unter der er erreichbar ist, innerhalb der Benutzer-ID vorhanden sein. Weitere Bestandteile der Benutzerkennung sind zulässig, sofern sie nicht irreführend sind oder mit ihnen eine zusätzliche Eigenschaft oder Funktion des Schlüsselinhabers charakterisiert wird, die die UNI-CA ansonsten vor einer Zertifizierung nachprüfen müßte. (Eine Benutzerkennung wie `Hans Schmidt <hans@UNI.de>, Diplom-Biologe, Ute Durchschnitt, UNI-Praesidentin` oder `Prof. Dr. Y. Mustermensch <muh@xy-provider.de>` wäre also nicht zertifizierbar.)

Ergänzungen in der Benutzer-ID, die eine Gültigkeitsdauer oder eine Anwendungsbeschränkung des Schlüssels andeuten sollen (z.B. 'SIGN' für Schlüssel, die nur zur Unterschriftenprüfung, nicht aber zum Verschlüsseln benutzt werden sollen, oder EXPIRE wie oben für den UNI-CA-Signierschlüssel beschrieben), sind ebenfalls zulässig. Ein Verfallsdatum in der Benutzerkennung muß aber, wenn es verwendet wird und zugleich im PGP-Public-Key-Packet ein Verfallsdatum für den Schlüssel angegeben ist, mit diesem übereinstimmen; andernfalls darf dieser Schlüssel von der UNI-CA nicht zertifiziert werden.

Schlüssel mit abgelaufener Gültigkeitsdauer werden ebensowenig zertifiziert wie solche mit einem „Erstellungsdatum", das in der Zukunft liegt.

Beispiele für zertifizierbare PGP-Benutzer-IDs im Sinne dieser Policy sind:

```
Arnold J. Rimmer, Uni Hamburg <ajr@uni-hamburg.de>
<dave.lister@inf.fu-berlin.de>
"Katharina Benutzer" <kathi123@aol.com> (SIGN)
Joe Juhser, 21. Dezember 1967, Hamburg (EXPIRE:2000-01-18)
```

5.4 Identitätsprüfung

Um unerlaubte Zertifizierungswünsche zu erkennen, hat sich die UNI-CA vor jeder Zertifizierung in geeigneter Weise von der Identität desjenigen Schlüsselinhabers zu überzeugen, welcher eine Zertifizierung wünscht. Dieser Vorgang kann nur durch persönlichen Kontakt vor der Zertifizierung erfolgen.

Der Benutzer muß sich persönlich vorstellen, um der CA die Prüfung seiner Identität zu ermöglichen. Für den Prozeß der Verifikation ist die Vorlage eines gültigen Personalausweises oder Reisepasses bzw. eines entsprechenden amtlichen Dokumentes mit Lichtbild erforderlich.

5.5 Anforderungen an den Schlüssel

Zertifikate werden ausschließlich dann erteilt, wenn der zu zertifizierende Public-Key über die in Abschnitt 4 festgelegten Mindest- und Höchstlängen verfügt, die zu zertifizierende Benutzerkennung die Anforderungen in Abschnitt 8 erfüllt und sich die UNI-CA in geeigneter Weise von der Identität des Schlüsselinhabers überzeugt hat. Schlüssel, die ein Verfallsdatum enthalten, werden von der UNI-CA nur vor diesem Datum zertifiziert; abgelaufene Schlüssel oder abgelaufene Benutzerkennungen werden nicht von der UNI-CA zertifiziert.

Im Zertifizierungsantrag sind die kryptographische Prüfsumme (der sog. „Fingerprint") des öffentlichen Schlüssels, der zertifiziert werden soll, sowie dessen Schlüssellänge und sein Erstellungsdatum anzugeben. Die UNI-CA hat sich vor der Zertifizierung anhand dieser Angaben zu vergewissern, daß der richtige Public-Key vorliegt. Der Public-Key muß in jedem Fall eine Selbst-Signatur des Inhabers aufweisen; solange diese nicht vorhanden ist, darf er nicht zertifiziert werden. (Eine fehlende Selbst-Signatur kann vom Schlüsselinhaber nachgereicht werden.)

Die Low-Level UNI-CA führt vor der Zertifizierung eines Schlüssels *keine* Prüfung durch, ob der Schlüsselinhaber auch über den Private-Key verfügt, der mit dem zu zertifizierenden Public-Key korrespondiert (sog. *proof of possession*, PoP). Ebenso prüft die Low-Level UNI-CA vor einer Zertifizierung nicht, ob der Schlüsselinhaber tatsächlich unter der Mailadresse erreichbar ist, die eventuell in der zu zertifizierenden Benutzerkennung enthalten ist. Das Zertifikat der Low-Level UNI-CA bezieht sich also ausschließlich auf den *Namen* des Schlüsselinhabers.

5.6 Gültigkeitsdauer

Ein PGP-Zertifikat enthält grundsätzlich den Public-Key sowie die Benutzer-ID. Da eine Gültigkeitsdauer digitaler Signaturen von vielen gebräuchlichen PGP-Versionen nicht unterstützt wird, begrenzt alleine die Gültigkeit des Zertifizierungsschlüssels (implizit) die Gültigkeitsdauer eines PGP-Zertifikats. Neben den für eine Gültigkeitsdauer vorgesehenen Datenstrukturen im PGP-Public-Key-Packet wird auch in der Benutzerkennung des Low-Level UNI-CA-Schlüssels durch Angabe der Zeichenfolge ‘EXPIRE:*jjjj-mm-tt*’ das entsprechende Verfallsdatum des betreffenden Signierchlüssels angegeben. (*jjjj* ist dabei durch die entsprechende vierstellige Jahreszahl, *mm* durch die zweistellige numerische Darstellung des Monats, ggf. mit führender Null, und *tt* entsprechend durch die Nummer des Tages im Monat, ggf. ebenfalls mit führender ‘0’, zu ersetzen.) Dadurch wird erreicht, daß auch PGP-Anwender, deren PGP-Version die Gültigkeitsdauer von PGP-Schlüsseln oder -Zertifikaten nicht automatisch auswertet, erkennen können, bis wann ein UNI-CA-Schlüssel gültig ist und bis wann folglich auch die damit ausgestellten Zertifikate höchstens gültig sind.

Die Zertifikate der Low-Level UNI-CA haben eine Gültigkeitsdauer von sieben Monaten. Sie werden 15 Tage vor Beginn jedes Hochschulsemesters erzeugt und gelten jeweils bis zum 15. Tag des folgenden Semsters.

5.7 Verlängerung von Zertifikaten

Zertifikate werden nicht automatisch durch die UNI-CA erneuert oder verlängert. Eine Re-Zertifizierung erfolgt nicht, es ist stattdessen ein neuer normaler Zertifizierungsantrag (wie bei der erstmaligen Zertifizierung) bei der Low-Level UNI-CA zu stellen.

6 Management von Zertifikaten

Die PGP-Zertifikate, die die UNI-CA ausstellt, werden von ihr auf den PGP-Keyservern des internationalen PGP-Keyserver-Verbundes (http://www.pgp.net/) veröffentlicht.

Alle Zertifikatnehmer der UNI-CA erklären sich mit der Veröffentlichung ihres Zertifikates einverstanden. Sie werden auf die Publikation der Daten auf den Zertifizierungsanträgen ausdrücklich hingewiesen. Ohne diese Einwilligung kann keine Zertifizierung erfolgen.

7 Widerruf von Zertifikaten

Die UNI-CA behält sich vor, von ihr erteilte Zertifikate jederzeit vor Ablauf der Gültigkeitsdauer ohne öffentliche Nennung expliziter Gründe zu widerrufen. Dem Schlüsselinhaber wird die UNI-CA über den Widerruf informieren und ihm auf Anfrage die Gründe mitteilen, die sie dazu veranlaßt haben.

Jeder Zertifikatnehmer der UNI-CA kann von ihr ohne Angabe von Gründen verlangen, daß sie das entsprechende Zertifikat für seinen Schlüssel widerruft. Die UNI-CA hat diesem Verlangen nachzukommen, sobald sie sich durch geeignete Schritte davon überzeugt hat, daß der Antrag vom Zertifikatnehmer selbst stammt bzw. von ihm autorisiert ist.

Wird der eigene geheime Schlüssel Dritten bekannt, so hat der betroffene Nutzer unverzüglich die UNI-CA zu benachrichtigen und den Widerruf des eigenen Zertifikates veranlassen, sobald er Kenntnis von diesem Umstand hat.

Die Low-Level DFN-Policy macht für den Widerruf von PGP-Zertifikaten keine konkreten Vorgaben für die CA; die dort gemachte Aussage, eine CA könne ein PGP-Zertifikat nicht widerrufen, ist technisch dank neuer PGP-Versionen nicht mehr zutreffend. Daneben wird in der DFN-Policy lediglich die Möglichkeit eines Schlüsselwiderrufs durch den *Schlüsselinhaber* selbst erwähnt und darauf hingewiesen, daß an der Verteilung solcher Widerrufe mittels eines X.500-Verzeichnisdienstes geforscht wird.

Die UNI-CA wird daher eigene Zertifikate auf folgende Weise widerrufen:

Viele PGP-Versionen (PGP2.6.3in, PGP 5.x, PGP 6.x) unterstützen inzwischen die Möglichkeit eines *Zertifikat*-Widerrufs. Da die bislang praktizierte Vorgehensweise zum Widerruf eines PGP-Zertifikates durch eine CA, die Veröffentlichung des betreffenden Zertifikates auf einer Sperr- oder Widerrufsliste (engl. *certificate revocation list*, CRL), für PGP in keiner Weise standardisiert war und daher auch von keiner der PGP-Versionen unterstützt wurde, wird die UNI-CA nicht mit derartigen Sperrlisten arbeiten. Stattdessen wird sie Zertifikat-Widerrufe einsetzen und diese über die PGP-Keyserver allen PGP-Anwendern zugänglich machen.

Es besteht darüber hinaus für alle Benutzer die Möglichkeit, einen Schlüsselwiderruf, ein sog. *key revocation certificate*, zu erzeugen und dieses Zertifikat im Falle des Widerrufs über die PGP-Keyserver zu verteilen, um die Ungültigkeit des eigenen Public-Keys bekanntzumachen.

Wichtiger Hinweis für relying parties:

Eine Person, die sich auf das Zertifikat der UNI-CA für den PGP-Schlüssel einer anderen Person verlassen will, muß zweierlei tun, um sichergehen zu können, daß ein eventuell existierendes Zertifikat der UNI-CA für den betreffenden Schlüssel nicht mittlerweile von der UNI-CA widerrufen wurde:

- *Es muß eine PGP-Version benutzt werden, die PGP-Zertifikatwiderrufe verarbeiten kann und sie anzeigt*
- *Es muß der betreffende PGP-Schlüssel einschließlich aller anhängigen Zertifikate von einem der Server des PGP-Keyserver-Verbundes www.pgp.net heruntergeladen und anschließend geprüft werden, ob nicht ein Widerruf des UNI-CA-Zertifikates vorliegt.*

(Da die Server des PGP-Keyserver-Verbundes jeweils untereinander ihre Schlüssel- und Zertifikatdaten synchronisieren, kann es dabei zu einem „Unsicherheitszeitraum“ von in der Regel maximal einem Tag kommen, in dem zwar die Zertifizierungsstelle eventuell ihr Zertifikat für einen Schlüssel widerrufen hat, dieser Widerruf aber noch nicht alle Server des Verbundes erreicht hat.)

8 Verschiedenes

Es wird keine Haftung für die Korrektheit, Vollständigkeit oder Anwendbarkeit der enthaltenen Informationen und der vorgeschlagenen Maßnahmen übernommen. Ferner kann keine Haftung für eventuelle Schäden, entstanden durch die Inanspruchnahme der Dienste der UNI-CA oder die Nutzung eines oder mehrerer von ihr ausgestellter Zertifikate, übernommen werden. Die Verantwortung für die Verwendung der oben beschriebenen Verfahren und Programme liegt allein bei den die Installation durchführenden Administratoren und Benutzern.

Die UNI-CA behält sich vor, Zertifizierungswünschen nicht nachzukommen. Ferner kann keine Garantie für die Verfügbarkeit der UNI-CA-Dienste übernommen werden. Es besteht derzeit keine Möglichkeit, diese Dienste auf einer 24-Stunden-Basis anzubieten.

Die UNI-CA fragt *niemals* Nutzer nach ihrer geheimen *Passphrase*, dem Code-Wort oder -satz, mit dem der geheime Nutzerschlüssel (*private key*) vor unbefugtem Zugriff geschützt ist!

Datenschutz

Alle Zertifikatnehmer stimmen mit dem Antrag auf Zertifizierung der Speicherung und Verarbeitung ihrer bei der Zertifizierung anfallenden Daten, die auch maschinell erfolgen wird, durch die zertifizierende Instanz zu.

Erklärung der Teilnehmer

Alle Teilnehmer der DFN-Hierarchie haben vor ihrer Zertifizierung handschriftlich eine Erklärung zu unterzeichnen, in der sie über ihre Rechte und Pflichten sowie über die Risiken und Gefahren beim Einsatz von Public-Key-Systemen aufgeklärt wurden. Diese Erklärung, die im Einzelfall auch vom Teilnehmer an die zertifizierende CA gefaxt werden kann, wird von der zertifizierenden Instanz verwahrt und beinhaltet in erster Linie die Zustimmung zu den Richtlinien dieser Policy sowie gegebenenfalls eine Erklärung darüber, von welcher Partei das zu zertifizierende asymmetrische Schlüsselpaar erzeugt wurde. Außerdem enthält diese Erklärung auch die Einwilligung des Schlüsselinhabers in die Veröffentlichung seines Zertifikates.

Maßgebliche Fassung dieser Policy

Eventuell werden Übersetzungen dieser Policy in andere Sprachen verfügbar gemacht, um beispielsweise die internationale Zusammenarbeit mit anderen CAs zu ermöglichen und Anwendern von Public-Key-Verfahren weltweit die Möglichkeit zu geben, die Arbeitsweise der Low-Level UNI-CA nachzuvollziehen und so die Verläßlichkeit ihrer Zertifikate einschätzen zu können. Maßgeblich ist jedoch in jedem Fall die deutschsprachige Version in ihrer jeweils aktuellsten Fassung.

Gebühren

Für die Leistungen der UNI-CA werden keine Gebühren erhoben.

„Dienstaufsicht“

Nutzer, die mit der Arbeit der UNI-CA unzufrieden sind, weil sie *konkretes* Fehlverhalten festgestellt haben, werden gebeten, ihr Anliegen den UNI-CA-Mitarbeitern per E-Mail oder persönlich während der Sprechzeiten mitzuteilen.

Darüber hinaus steht den Betroffenen die Möglichkeit offen, sich bei Policy-Verstößen der UNI-CA an die DFN-PCA als übergeordnete Kontrollinstanz zu wenden (Adresse siehe 3.1).

Literaturverzeichnis

[DFN-PCA] S. Kelm, B. Liedtke: *DFN-PCA – Low-Level Policy. Zertifizierungsrichtlinien für das PCA-Projekt*, Version 1.2, 1. Januar 1999

[PGP] Philip Zimmermann: *The Official PGP User's Guide*, MIT Press, 1995

[RFC 1422] S. Kent: *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*, Februar 1993

[RFC 1875] N. Berge: *UNINETT PCA Policy Statements*, Dezember 1995

[SigG] *Gesetz zur digitalen Signatur (Signaturgesetz – SigG)* vom 22. Juli 1997, BGBl. I, S. 1870 ff.

Abkürzungsverzeichnis

CA:	Certification Authority (Zertifizierungsinstanz)
CRL:	Certificate Revocation List (Widerrufsliste)
DFN:	Verein zur Förderung eines Deutschen Forschungsnetzes e.V.
FTP:	File Transfer Protocol
ID:	Identifier
IPRA:	Internet PCA Registration Authority
PCA:	Policy Certification Authority
PIN:	Personal Identification Number
PGP:	Pretty Good Privacy
RA:	Registration Authority (Registrierungsinstanz)
RFC:	Request for Comment
SigG:	Signaturgesetz
WiN:	Wissenschaftsnetz

Anhang L

Standard-CA-Mails

Bestimmte Anfragen und Abläufe wiederholen sich für die Mitarbeiter einer Zertifizierungsstelle immer wieder. Dazu gehören auch Standardantworten bzw. -anschreiben zu den am häufigsten vorkommenden Fragen bzw. Zertifizierungsproblemen. Nachfolgend sollen einige Beispiele dafür gegeben werden, welche Situationen oft auftreten und wie eine entsprechende E-Mail-Reaktion eines CA-Mitarbeiters aussehen könnte.

Mail an die CA mit falschem Key verschlüsselt

```
Hallo ... ,

Du hast PGP-verschlüsselte Mail an die UNI-CA <ca@fu-berlin.de>
geschickt.

Leider hast Du für die Verschlüsselung den Key 0x06753A4D, UserID

UNI-Berlin CA, SoSe 1999 Certification Key <http://ca.UNI.de>

verwendet.

Dieser Key ist nicht zur Sicherung der Vertraulichkeit der Kommunikation
mit der UNI-CA vorgesehen, sondern er wird ausschließlich dazu verwendet,
Schlüssel zu zertifizieren und Schlüsselwiderrufe zu signieren (Private-
Key, Verwendung durch die UNI-CA) bzw. entsprechende Unterschriften der
UNI-CA zu verifizieren (entsprechender Public-Key). Dafür steht auch das
'SIGN' in der UserID dieses Keys.

Deine Mail konnte daher von den UNI-CA-Mitarbeitern nicht gelesen werden.

Bitte schicke die Mail(s) noch einmal und benutze dabei den Schlüssel

Type Bits/KeyID    Date       User ID
pub  1024/A6A81269 1999/03/20 UNI CA, SoSe 1999 Communication Key <ca@UNI.de>

wenn Du die Inhalte Deiner Kommunikation mit der UNI-CA  vor
unbefugter Kenntnisnahme durch Dritte schützen willst. (Unten ange-
```

```
hängt findest Du den kompletten Key in PGP-ASCII-Armor-Darstellung).

[Grußformel]
[angefügter Schlüssel]
```

Nutzerschlüssel ist nicht auf Keyserver

```
Hallo ....,

Du hast die Zertifizierung Deinen PGP-Public-Keys mit der KeyID
0xMMMMMMMM bei der UNI-CA beantragt.

Dieser PGP-Key ist bisher *nicht* auf den internationalen Keyservern
gespeichert, so daß wir ihn nicht von dort herunterladen können.

Damit wir den Schlüssel trotzdem zertifizieren können, schick' uns
bitte eine Kopie dieses Public-Keys per Mail an 'ca@UNI.de'.

[Grußformel]
```

Nutzerschlüssel weist keine Selbst-Signatur auf

```
Hallo xxxx,

Du hattest einen Antrag auf Zertifizierung Deines PGP-Keys

Type Bits/KeyID    Date       User ID
......

durch die UNI-CA gestellt.

Dieser Key bzw. die betreffende o.g. UserID ist nicht von Dir selber
unterschrieben (zumindest nicht die Kopie, die auf den internationalen
PGP-Keyservern liegt).

Das solltest Du umgehend nachholen - warum, das kannst Du im unten
angefügten Text aus der deutschsprachigen Übersetzung des
"Comp.security.pgp FAQ", übersetzt von Lutz Donnerhacke, nachlesen.

Solange der Schlüssel/die betreffende UserID nicht selbst-signiert
ist, kann er nicht von der UNI-CA zertifiziert werden; das
verbieten die Fu-CA-Zertifizierungsrichtlinien:

  5 Zertifizierungsregeln

   [...]
   Der Public-Key muß in jedem Fall eine Selbst-Signatur des
   Inhabers aufweisen; anderenfalls darf er nicht zertifiziert
   werden. (Eine fehlende Selbst-Signatur kann vom Schlüsselinhaber
   nachgereicht werden.) Liegt sie vor und sind alle anderen
   o.g. Voraussetzungen erfüllt, darf der Schlüssel von der CA
```

```
   zertifiziert werden.)


Bitte maile an 'ca@UNI.de' eine Kopie Deines o.g. PGP Public
Keys, nachdem Du den Key/die UserID selbst unterschrieben hast!

[Grußformel]
```

Keine DSS/DH-Schlüssel

```
[Anrede]

die von Dir registrierten persoenlichen Daten und den Public-Key haben
wir erhalten. Leider koennen wir aus Kompatibiliaetsgruenden derzeit noch
keine PGP-Schluessel der "neuen Generation" (DSS/DH-Keys von PGP 5.x und
6.x) zertifizieren.

Wir testen derzeit die neue PGP-Version und werden Deinen Schluessel
zertifizieren, sobald diese Version fuer den laufenden Betrieb der UNI-CA
verwendet werden kann. Wir werden Deinen zertifizierten Key anschliessend
per Email an Dich schicken.

Wir bitten um etwas Geduld!

[Grussformel]
```

Diskrepanz zwischen Schlüssel und Antrag

```
[Anrede]

Du hast die Zertifizierung Deinen PGP-Schlüssels 0xMMMMMMMM bei der UNI-CA
beantragt.

Leider stimmen die von Dir auf dem Zertifizierungsantrag gemachten Angaben
nicht mit den uns vorliegenden Schlüsselinformationen überein oder sind
unvollständig!
(Eventuell ist Dir einfach ein Zahlen- oder Buchstabendreher unterlaufen,
oder wir konnten Deine Handschrift nicht entziffern.)

Eine Zertifizierung kann aber nur erfolgen, wenn die Angaben im
Zertifizierungsantrag mit dem uns vorliegenden Schlüssel übereinstimmen.

Du hast jedoch die Möglichkeit, einen neuen Zertifizierungsantrag für
Deinen o.g. Schlüssel bei der UNI-CA zu stellen. Dieser Antrag kann von
uns aber nur von Dir persönlich entgegengenommen werden; Du müßtest
uns also gegebenenfalls zu unseren Sprechzeiten im Zertifizierungsbüro,
....
```

```
besuchen.

[Grußformel]
```

Zu kurzer Nutzerschlüssel

```
[Anrede]

Du hast die Zertifizierung Deinen PGP-Schlüssels 0xMMMMMMMM bei der UNI-CA
beantragt.

Dein Schlüssel ist jedoch fuer eine Zertifizierung durch die UNI-CA
nicht lang genug.

Unsere Zertifizierungspolicy schreibt eine Mindestlänge von ... Bits
vor, damit ein Schlüssel zertifiziert werden kann. In diesem Punkt
können, dürfen und wollen wir keine Ausnahmen machen.

Wenn Du gerne eine von der UNI-CA zertifizierten PGP-Schlüssel haben
möchtest, müßtest Du bitte einen neuen, ausreichend langen Key erzeugen
und dann für diesen neuen Schlüssel einen Zertifizierungsantrag bei uns
abgeben.

[Grußformel]
```

Fertiges Zertifikat

```
[Anrede]

nachstehend findest Du das von Dir beantragte Zertifikat der
Low-Level UNI-CA für Deinen PGP-Public-Key sowie die aktuellen
Zertifizierungsschlüssel der UNI-CA und der DFN-PCA. Diese Keys bzw.
deren Fingerprint, KeyID und Länge solltest Du auch beim persönlichen
Kontakt mit den Mitarbeitern der UNI-CA ausgehändigt bekommen haben.

Ansonsten findest Du den Fingerprint der UNI-CA-Schlüssel nebst
der übrigen zur Verifikation erforderlichen Angaben im aktuellen
UNI-Vorlesungsverzeichnis und im Impressum der "UNI-News".

Viel Spaß mit PGP!

[Grußformel]


[alternativ: Anschreiben und Schlüssel in separaten Mails, falls sie sich
 so leichter automatisch generieren lassen]

Key for user ID: Michaela Musterfrau <mm@UNI.de>
768-bit key, key ID MMMMMMM, created 1995/07/18
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3in
Comment: Zertifizierungsstelle der UNI

mQBtAzBMM5kAAAEDAOA3PTMQIjcHMRtkYq+WP2uGftMTlnzOoNaaYdOYX37jCdUC
ZRkbucVhA43XwXhewNipKsBZe+jzuSCR4znV1bMnkk38y2WhceDr8jo/lL0c7cf4
 ...
[zertifizierter Schlüssel]
 ...
4PiIIJvTzZYOnnDY3xUsb3VHIaM1jyTnqopZGDoCEODxcKcBVr76z4Fv5i/t/2gT
sZ1IXsKyhF0QhWYHeC3xcmoDYmC1y0uyb4jX6pYzqX6d
=F/5f
-----END PGP PUBLIC KEY BLOCK-----


Key for user ID: UNI CA, SoSe 1999 Certification Key
<http://ca.UNI.de> (SIGN EXPIRE:1999-10-15)
1024-bit key, key ID 06753A4D, created 1999/02/20
Key is a SIGNature only key.

-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----


Key for user ID: DFN-User-CA, CERTIFICATION ONLY KEY (Low Level: 1997-1998)
<http://www.pca.dfn.de/dfnpca/>
2048-bit key, key ID FE93EAB9, created 1997/04/16

-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----
```

Re-Zertifizierung eines Schlüssels

```
[Anrede]

Du hast vor einiger Zeit Deinen PGP-Key

Type Bits/KeyID    Date       User ID
pub  1024/00000000 1999/11/22 Erwin Mustermann <ermu@.UNI.de>

von der UNI-CA zertifizieren lassen. (Das Zertifikat wurde Dir per
E-Mail zugeschickt und auf den Keyservern des internationalen
Keyserververbundes www.pgp.net abgelegt.)

Der Schlüssel der UNI-CA, mit dem Dein Key von uns signiert wurde,
läuft demnächst ab, d.h. er ist dann nicht mehr gültig. Mit Ablauf
seiner Gültigkeitsdauer verfallen wegen Zeitablaufs auch die mit ihm
ausgestellten Zertifikate.

[alternativ:
```

```
Du hast die Re-Zertifizierung dieses Schlüssels durch die UNI-CA beantragt]


Die Re-Zertifizierung kann nach einfacherer Überprüfung als bei der
Erst-Zertifizierung und diesmal *ohne* persönlichen Kontakt zwischen Dir
und der UNI-CA erfolgen.

Diese Mail dient dazu, zweierlei zu prüfen:

1. Bist Du nach wie vor im Besitz des Private Keys, der zum o.g.
   Public-Key gehört?

2. Bist Du noch immer unter der oben in Deiner Key-UserID
   genannten Mailadresse erreichbar?

Wenn Du diese Nachricht erhalten hast und sie entschlüsseln kannst,
dann sind die beiden Bedingungen oben erfüllt. Und nur dann ist eine
Re-Zertifizierung Deines PGP-Keys möglich.

Bitte sende als Nachweis, daß Du diese Mail erhalten und entschlüsselt
hast und daß Du eine Re-Zertifizierung Deines o.g. PGP-Schlüssels durch
die UNI-CA wünschst, die folgende 'Challenge' (eine "Herausforderung",
die ein potentieller Angreifer nicht durch Raten herausbekommen kann)
in einer MIT DEINEM o.g. SCHLÜSSEL PGP-SIGNIERTEN und an die UNI-CA
VERSCHLÜSSELTEN Mail an die Mailadresse 'ca@UNI.DE'.
(Der UNI-CA Verschlüsselungs-Key ist unten angefügt.)


                        [........] {Zufallszahl}


Mit Deiner entsprechenden Antwort-Mail erhält die UNI-CA einen
Beleg dafür, daß Du offenbar unter der betreffenden Mailadresse er-
reichbar bist. (Adernfalls könntest Du die "Challenge" nicht kennen,
da diese zufällig gewählt wurde und somit nach menschlichem Ermessen
nicht erraten werden oder aus der verschlüsselten Mail von einem
Angreifer ersehen werden kann.)

Wenn Deine Antwort-Mail außerdem wie verlangt von Dir PGP-signiert ist,
dann kann die UNI-CA daran sehen, daß Du wirklich im Besitz des
privaten Schlüssels zum obigen öffentlichen Schlüssel bist. (Nur
dann kannst Du nämlich eine PGP-Signatur erzeugt haben, die sich mit
Deinem öffentlichen Schlüssel erfolgreich verifizieren läßt!)

Diese Challenge dient zugleich auch als sog. "gemeinsames Geheimnis"
("shared secret"), das nur Dir und der UNI-CA bekannt ist. Es ist
Dein "Berechtigungsnachweis", falls Du per E-Mail den WIDERRUF der
UNI-CA-Signatur unter diesem Key bzw. dieser Benutzerkennung Deines Keys
beantragen möchtest und z.B. wegen des Verlustes Deines Secret Keys oder
wegen einer vergessenen Passphrase für diesen Key nicht mehr in der
Lage bist, selber eine Key Revocation oder zumindest eine PGP-signierte
Widerrufs-Anforderung zu erzeugen und an die UNI-CA zu senden.

Du solltest die o.g. Challenge also unbedingt vor unbefugtem Zugriff
sicher verwahren - wer sie kennt, kann veranlassen, daß die UNI-CA
```

```
ihr Zertifikat für Deinen Schlüssel widerruft!

[Grußformel]

Anlage:

Der Kommunikationsschlüssel der UNI-CA für 1999

-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----
```

Anhang M

Kontaktinformationen anderer Stellen

SigG-Wurzelinstanz („Zuständige Behörde“)

Die „Zuständige Behörde“ ist die oberste Zertifizierungsstelle (Root-CA), die das Signaturgesetz vorsieht. Ihr sind alle anderen SigG-Zertifizierungsstellen untergeordnet. Wahrgenommen wird diese Aufgabe gemäß § 3 des Gesetzes zur Digitalen Signatur von der

> Regulierungsbehörde für Telekommunikation und Post
> Referat Sicherheit in der Telekommunikation / Digitale Signatur
> Canisiusstraße 21
> D-55122 Mainz
> Ansprechpartner: Herr Schwemmer
> Telefon 0 61 31 / 18 - 22 10
> Telefax 0 61 31 / 18 - 56 18
> E-Mail: `DigitaleSignatur@regtp.de`
> `http://www.regtp.de`
> `http://www.nrca-ds.de`[1] = Verzeichnisdienst der SigG-Root-CA

c't pgpCA

> *c't* pgpCA
> Ansprechpartner: Herr Luckhardt
> Verlag Heinz Heise GmbH & Co KG
> Postfach 61 04 07
> D-30604 Hannover
> Telefon: 05 11 / 53 52 - 5 13 (Hotline, nur 13–14 Uhr)
> Telefax: 05 11 / 53 52 - 4 17
> E-Mail: `pgpCA@ct.heise.de`
> `http://www.heise.de/ct/pgpCA/`

[1] *National Root Certification Authority for Digital Signatures*

Datenschutzbeauftragter Schleswig-Holstein

Der Landesbeauftragte für den Datenschutz
Schleswig-Holstein
Düsternbrooker Weg 82
D-24105 Kiel
Telefon: 04 31 / 9 88 - 12 00
Telefax: 04 31 / 9 88 - 12 23
E-Mail: `LDSH@netzservice.de` (PGP-Key 0x8BFDD679)
`http://www.schleswig-holstein.datenschutz.de`

Tip: Der Schleswig-Holsteinische Datenschutzbeauftragte ist eine gute Quelle für witzige Aufkleber *pro* Verschlüsselung und für entsprechende Faltblätter. Sie werden nach unserer Erfahrung auf Anfrage – bis zu gewissen Grenzen auch in größeren Stückzahlen – zugeschickt.

Bundesausfuhramt

Bundesausfuhramt (BAFA)
Frankfurter Straße 29–35
D-65760 Eschborn

Postadresse:
Postfach 5160
D-65726 Eschborn
Telefon: 0 61 96 / 9 08 - 0
Telefax: 0 61 96 / 9 08 - 8 00
`http://www.bundesausfuhramt.de`
E-Mail: `poststelle@bundesausfuhramt.de`

Bundesamt für Sicherheit in der Informationstechnik (BSI)

Bundesamt für Sicherheit in der Informationstechnik
Godesberger Allee 183
D-53133 Bonn

Postadresse:
Postfach 20 03 63
D-53133 Bonn
Telefon: 02 28 / 95 82 - 0
Telefax: 02 28 / 95 82 - 4 00
`http://www.bsi.de`
E-Mail: `gshb@bsi.de` für Fragen zum Grundschutzhandbuch

Sonstige Zertifizierungsstellen

Die Anschriften weiterer wichtiger Zertifizierungsstellen aus aller Welt können dem *Global Trust Register* (GTR) entnommen werden, das vom Team um Professor Ross Anderson (Cambridge) herausgegeben wird. Das Verzeichnis nennt Namen, Anschriften, Online-Adressen und Schlüsselinformationen der Public-Keys von etlichen hundert der wichtigsten Zertifizierungsstellen weltweit, von Computer Emergency Response Teams (CERTs) aus etlichen Ländern und von wichtigen Einzelpersonen aus dem Bereich der Computersicherheit. Die 1998er-Ausgabe des GTR ist zusätzlich auch online verfügbar [ACL$^+$98, ACP99].

Abkürzungsverzeichnis

3DES	*Triple-DES* (→DES), symmetrisches Verschlüsselungsverfahren
AGB	Allgemeine Geschäftsbedingungen
AMBIX	→DFN-Projekt „Aufnahme von Mail-Benutzern in das X.500-Directory“
ASCII	*American Standard Code for Information Interchange*
ASN.1	*Abstract Syntax Notation One*, geräteunabhängiges Darstellungs- und Austauschformat für Daten
BDSG	Bundesdatenschutzgesetz
BGB	Bürgerliches Gesetzbuch
BGBl.	Bundesgesetzblatt
BIND	*Berkeley Internet Name Domain*, Server-Programm, das Internet-Namen in Internet-Adressen auflöst
BIOS	*Basic Input/Output System*
BlnDSG	Berliner Datenschutzgesetz
BMWi	Bundesministerium für Wirtschaft und Technologie
BSI	Bundesamt für Sicherheit in der Informationstechnik
C	*Country*, →X.500-Attribut für ‘Land’
CA	*Certification Authority* (selten: *Certificate Authority*), Zertifizierungsstelle
CCITT	Comité Consultatif International Pour Télégraphique et Téléphonique, Organ der →ITU
CD-ROM	*Compact-Disc – Read-Only Media*
CEO	*Chief Executive Officer*, Geschäftsführer
CERT	*Computer Emergency Response Team*, Computer-Notfallteam
CESG	*Communications-Electronics Security Group*, britische Regierungsstelle für Kryptographie und Kommunikationssicherheit
CFS	*Cryptographic Filesystem*, verschlüsselndes Dateisystem
CGI	*Common Gateway Interface*, Schnittstelle zum dynamischen Erzeugen von →WWW-Seiten
CN	*Common Name*, →X.500-Attribut für technische Namen
CPS	*Certification Practice Statement*, etwa →„AGB“ einer →CA
CPU	*Central Processing Unit*, Zentraleinheit (Mikroprozessor)

CRL *Certificate Revocation List*, Sperr-/Widerrufsliste, auf der ungültig gewordene Zertifikate geführt werden

CSR *Certificate Signing Request*, Zertifizierungswunsch

DER *Distinguished Encoding Rules*, →ASN.1-Darstellungsformat

DES *Data Encryption Standard*, symmetrischer Verschlüsselungsalgorithmus, US-Standard

DFN Deutsches Forschungsnetz (Verein zur Förderung eines deutschen Forschungsnetzes e. V.)

DH Diffie-Hellman, Public-Key-Verschlüsselungsverfahren (benannt nach seinen Erfindern Whitfield Diffie und Martin Hellman)

DIHT Deutscher Industrie- und Handelstag

DN Distinguished Name, eindeutiger Name gemäß →X.500

DNA *Desoxyribonucleic Acid*, Desoxyribonukleinsäure (→DNS); Träger der Erbinformationen

DNS *Domain Name Service*, Internet-Namensdienst (→BIND); *auch:* Desoxyribonukleinsäure (→DNA)

DOS *Disc Operating System*, Betriebssystem

DSA *Digital Signature Algorithm*, mathem. Verfahren zur Erzeugung digitaler Signaturen

DSO *Dynamic Shared Object*, zur Laufzeit ladbare Programmmodule

DSS *Digital Signature Standard*, US-Standard, der den Einsatz des →DSA beschreibt

E2S *End-to-End Security*, Ende-zu-Ende-Sicherheit

EMA Elektromagnetische Abstrahlung

EMRK Europäische Menschenrechtskonvention

EU Europäische Union

FAQ *Frequently Asked Questions*, Sammlung häufig gestellter Fragen und Antworten zu einem Thema

FB Fachbereich

FLOPS *Floating-point Operations per Second*, Fließkomma-Operationen pro Sekunde (Maß für die Rechengeschwindigkeit einer →CPU)

FOI *Freedom of Information*, Informationsfreiheit (freier Zugang zu [amtlichen] Informationen)

FTP *File Transfer Protocol*, Internet-Standard zur Datei-Übertragung

GB Gigabyte

GMD GMD – Forschungszentrum Informationstechnik GmbH (ehem. Gesellschaft für Mathematik und Datenverarbeitung)

GPL *GNU General Public License*, Lizenzbestimmungen, unter denen viele Open-Source-Programme vertrieben werden

GSHB Grundschutz-Handbuch des →BSI

GUI *Graphical User Interface*, graphische Benutzeroberfläche

GUUG *German Unix User Group*, Deutsche Unix-Anwendervereinigung

HTML *Hypertext Mark-up Language*, Struktur- und Auszeichnungssprache für Hypertext-Dokumente

HTTP *Hypertext Transfer Protocol*, Anwendungsprotokoll zur Übertragung von Hypertext-Dokumenten

HTTPS *Secure Hypertext Transfer Protocol*, *sicheres* →HTTP

IAB *Internet Architecture Board*, Wahlgremium, Schlichtungsorgan und Beratergruppe der Internet Society, veröffentlicht die →RFC-Dokumente

ICE *Interworking Public Key Certification Infrastructure for Europe*, EU-Forschungsprojekt zum Aufbau einer Public-Key-Zertifizierungs-Infrastruktur in Europa

ICE-CAR *Interworking Public Key Certification Infrastructure for European Commerce, Administration and Research*

ID *Identity/Identification Data*, Identitätsdaten

IDEA *International Data Encryption Algorithm*, symmetrisches Verschlüsselungsverfahren

IE *Internet Explorer* (→MSIE)

IEC *International Engineering Consortium*, internationales Standardisierungsgremium

IESG *Internet Engineering Steering Group*, leitet gemeinsam mit dem →IAB den von der →IETF geführten Internet-Standards-Entwicklungsprozeß

IETF *Internet Engineering Task Force*, loser Zusammenschluß von Menschen, die das Internet voranbringen/weiterentwickeln (wollen)

IP *Internet Protocol*, die gemeinsame „Sprache“, die alle Rechner des Internet miteinander „sprechen“

IFA Internationale Funkausstellung Berlin

IR *Infrared*, Infrarot

ISO *International Standardization Organization*

IT *Information Technology*, Informationstechnik

ITU *International Telecommunication Union*, internationales Gremium der staatlichen Post- und Telekommunikationsministerien

IVBB Informationsverbund Berlin–Bonn

KD Konfigurationsdatei

KRC *Key Revocation Certificate*, Schlüsselwiderruf

L *Locality*, →X.500-Attribut für ‘Ort’

LCD *Liquid Crystal Display*, Flüssigkristall-Anzeige

MacOS *(Apple) Macintosh Operating System*, Betriebssystem

MB Megabyte

MD5 *Message Digest #5*, kryptographisches Prüfsummenverfahren

MHS *Message Handling System*, Standard zum Austausch elektronischer Nachrichten

MIME *Multi-purpose Internet Mail Extensions*, Erweiterung des Internet-Mail-Formates

MIPS *Million Instructions per Second*, Rechenoperationen pro Sekunde

MS Microsoft, US-Soft- und -Hardware-Hersteller

MSIE →MS Internet Explorer, Software

NAI Network Associates Inc., Computer- und Netzwerk-Sicherheitsunternehmen, hält die Rechte an →PGP

NFS *Network File System*, System zum Zugriff auf Dateisysteme zwischen vernetzten Computern

NiCd Nickel-Cadmium, Akkumulatoren-Typ

NiMH Nickel-Metallhydrid, Akkumulatoren-Typ

NRW Nordrhein-Westfalen

NSA *National Security Agency*, scherzhaft auch "No such agency", US-Geheimdienst, für Kommunikationssicherheit und -überwachung zuständig

NSC *Netscape Communicator/Navigator*, Software

O →X.500-Attribut für 'Organisation'

OCSP *Online Certificate Status Protocol*, Internet-Anwendungsprotokoll zur Abfrage eines Zertifikat-Gültigkeitsstatus'

OE →MS Outlook Express, Software

OID *Object Identifier*, numerischer →ASN.1-Objekt-Bezeichner

OU *Organizational Unit*, →X.500-Attribut für 'Organisationseinheit' (i.S.v. „Abteilung“, „Bereich“)

PCMCIA *Personal Computer Memory Card (Ĩnternational Association)*

PC Personal Computer

PDF *Portable Document Format*, Format zum Dokumentenaustausch

PEM *Privacy Enhancement for Internet Electronic Mail*, Internet-Standard für gesicherte E-Mail-Kommunikation

PFX *Personal Information Exchange*, Austauschformat für vertrauliche persönliche Informationen, z.B. eigene geheime Schlüssel

PGP *"Pretty Good Privacy"*, Verschlüsselungssoftware

PIN persönliche Identifikationsnummer

PKCS *Public Key Cryptography Standard*, technisches Dokument der Firma →RSA Security Inc.

PKI Public-Key-Infrastruktur (Infrastruktur zur Verteilung und Verwaltung öffentlicher Schlüssel und entsprechender Zertifikate)

PKIX Internet Public-Key-Infrastruktur auf →X.509-Basis

PoP *Proof of Possession*, Besitznachweis

PR *Public Relations*, öffentlichkeitsarbeit

RA *Registration Authority*, Registrierungsstelle

RC5 *Ron's Cipher #5*, symmetrisches Verschlüsselungsverfahren von Ronald Rivest (→RSA)

RegTP Regulierungsbehörde für Telekommunikation und Post

RFC *Request for Comments*, Dokument der →IETF

RID *Registered →ID*

RIPE	*Réseaux IP Européens*, europäische Vergabestelle für →IP-(Netz-)Nummern
RNG	*Random Number Generator*, Zufallszahlengenerator
RSA	Public-Key-Verfahren, benannt nach seinen Erfindern Ron Rivest, Fiat Shamir und Leonard Adleman (auch: von ihnen gegründete Firma)
RZ	Rechenzentrum
S/MIME	*Secure/Multipurpose Internet Mail Extensions*, Erweiterung des →MIME-Formates um Sicherheitsfunktionalität
SCSI	*Small Computer System Interface*, Computer-Schnittstelle
SGC	*Server-gated Cryptography*, spezielle Form der Aktivierung von starker Verschlüsselung
SHA	*Secure Hash Algorithm*, kryptographisches Prüfsummenverfahren
SigG	Signaturgesetz
SigV	Signaturverordnung
SPD	Sozialdemokratische Partei Deutschlands
SSL	*Secure Socket Layer*, Sicherheitsprotokoll für den Datenaustausch via →IP
ST	*State*, →X.500-Attribut für ‘Bundesstaat’
STOA	*Scientific and Technological Options Assessment*, technisches Beratungsgremium des EU-Parlaments
TCFS	*Transparent Cryptographic Filesystem*, ein für den Benutzer „transparent“, d.h. ohne aufzufallen verschlüsselndes Dateisystem (→CFS)
TIE	*Trust Infrastructure for Europe*, EU-Forschungsprojekt
TKG	Telekommunikationsgesetz
TLS	*Transport Layer Security*, Weiterentwicklung von →SSL
TTP	*Trusted Third Party*, vertrauenswürdige(r) Dritte(r)
TU	Technische Universität
TÜV	Technischer Überwachungsverein
UA	*User Agent* (meist in *Mail-UA*), Anwender-Software
UID	*User-→ID*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*, Computer-Schnittstelle
UTC	*Universal Time Coordinates*, Weltzeit
VPN	*Virtual Private Network*, sichere Verbindung mehrerer eigener Rechner über ein (teilweise) unsicheres Medium
W3C	*World Wide Web Consortium*, Standardisierungsgremium
WWW	*World Wide Web*
XOR	Exklusiv-Oder
X.208	→ITU-T Empfehlung X.208 Spezifikation der “Abstract Syntax Notation One”

X.400	→ITU-T Empfehlung X.400 “Message Handling System”
X.500	→ITU-T Empfehlung X.500 “The Directory”
X.509	→ITU-T Empfehlung X.509 “The Directory – Authentication Framework”
ZZG	Zufallszahlen-Generator

Literaturverzeichnis

[AAG+99] ADAMS, CARLISLE, RICH ANKNEY, SLAVA GALPERIN, AMBARISH MALPANI und MICHAEL MYERS: *RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*, Juni 1999.

[ABR97] ADEN, J.-U., FRITZ BAUSPIESS und R. RUDELOFF: *Ende-zu-Ende-Sicherheit für elektronischen Dokumentenaustausch – Infrastruktur und Leitlinien für die Bundesverwaltung*, 15. Juni 1997. Entwurf, Version 0.12. `http://www.bsi.bund.de/aufgaben/projekte/sphinx/endeende.htm`.

[ACL+98] ANDERSON, ROSS J., BRUNO CRISPO, JONG-HYEON LEE, CHARALAMPOS MANIFAVAS, VÁCLAV MATYÁŠ, JR und FABIEN A. P. PETITCOLAS (Hrsg.): *The Global Trust Register for 1998*. Northgate Consultant, Wrestlingworth, Sandy, Bedfordshire SG19 2HA, UK, 1998. Die Online-Version des GTR von 1998 ist abrufbar unter `http://www.cl.cam.ac.uk/Research/Security/Trust-Register/`.

[ACP99] ANDERSON, ROSS J., BRUNO CRISPO und FABIEN PETITCOLAS: *The Global Trust Register*. MIT Press, April 1999.

[ACPZ00] ADAMS, CARLISLE, PAT CAIN, DENIS PINKAS und ROBERT ZUCCHERATO: *Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP)*, Januar 2000. Internet-Draft PKIX Working Group (work in progress) <draft-ietf-pkix-time-stamp-05.txt>.

[Adl98] ADLEMAN, LEONARD M.: *Rechnen mit DNA*. Spektrum der Wissenschaft, Seiten 70–77, November 1998.

[AE97] ADEN, JÖRG-UDO und OLAF ERBER: *Der Aufbau des Informationsverbunds Berlin-Bonn (IVBB) unter besonderer Berücksichtigung von Sicherheitsfragen*. In *BSI-Kongreß* [BSI97], Seiten 137–150.

[AIG98] *Akteneinsichts- und Informationszugangsgesetz (AIG) Brandenburg*, 10. März 1998. GVBl. I, S. 46, online unter `http://www.brandenburg.de/land/lfdbbg/gesetze/aig.htm`.

[AK96] ANDERSON, ROSS J. und MARKUS KUHN: *Tamper Resistance – a Cautionary Note*. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, Seiten 1–11, Oakland, California, 18.–21. November 1996. The USENIX Association.

[ARH97] ABDUL-RAHMAN, ALFAREZ und STEPHEN HAILES: *A Distributed Trust Model*. In *Proceedings ACM New Security Paradigms Workshop*, Seiten 48–60, Langdale, Cumbria, UK, 23.–26. September 1997.

[AS98] ALMOND, JIM und DAVE SNELLING: *UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web*, 13. Oktober 1998. Online unter `http://www.fz-juelich.de/zam/RD/coop/unicore/whitepaper.ps`.

[ASZ96] ATKINS, DEREK, WILLIAM STALLINGS und PHILIP ZIMMERMANN: *RFC 1991: PGP Message Exchange Format*, August 1996.

[AT99] ARSENAULT, ALFRED und SEAN TURNER: *Internet X.509 Public Key Infrastructure – PKIX Roadmap*, 22. Oktober 1999. Internet-Draft PKIX Working Group (work in progress) <draft-ietf-pkix-roadmap-04.txt>.

[Bau96] BAUSPIESS, FRITZ: *MailTrusT 1.1*. Spezifikation, TeleTrusT e.V., 18. Dezember 1996. online unter `http://www.darmstadt.gmd.de/mailtrust/MTTv1/mttspc11.pdf`.

[BAz98] *Bekanntmachung zur digitalen Signatur nach dem Signaturgesetz und der Signaturverordnung vom 28. September 1998*, 6. Oktober 1998. Bundesanzeiger Nr. 186.

[BBFK98] BARTHEL, JOCHEN, HANS-JOACHIM BRACZYK, GERHARD FUCHS und KORNELIA KONRAD: *Vertrauensbildung aus soziologischer Sicht – das Beispiel Sicherheit in der Kommunikationstechnik*. In MÜLLER, GÜNTER und KURT-HERMANN STAPF [MS98], Seiten 119–147.

[BBPT98] BAUKNECHT, KURT, ALFRED BÜLLESBACH, HARTMUT POHL und STEPHANIE TEUFEL (Hrsg.): *Sicherheit in Informationssystemen. Proceedings der Fachtagung SIS '98*, Zürich, 1998. vdf Hochschulverlag.

[BC98] BAKER, FRED (IESG AND IETF CHAIR), und BRIAN CARPENTER (IAB CHAIR): *Harmful changes to the Wassenaar Arrangement*, 18. Dezember 1998. Mail an die IETF-Announce Mailingliste, online unter `http://www.ietf.org/mail-archive/ietf-announce/Current/msg02603.html`.

[BDS90] *Bundesdatenschutzgesetz –BDSG– (Artikel 1 des Gesetzes zur Fortentwicklung der Datenverarbeitung und des Datenschutzes)*, 20. Dezember 1990. BGBl. I, S. 2954.

[BF93] BORENSTEIN, NATHANIEL S. und NED FREED: *RFC 1521: MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, September 1993.

[BHK+94] BIZER, JOHANN, VOLKER HAMMER, CHRISTEL KUMBRUCK, ULRICH PORDESCH, ALEXANDER ROSSNAGEL, HEINZ SARBINOWSKI, MICHAEL J. SCHNEIDER, PROJEKTGRUPPE VERFASSUNGSVERTRÄGLICHE TECHNIKGESTALTUNG E.V. (PROVET) und GESELLSCHAFT FÜR MATHEMATIK UND DATENVERARBEITUNG MBH (GMD): *Die Simulationsstudie Rechtspflege. Eine neue Methode zur Technikgestaltung für Telekooperation*. Edition Sigma, Berlin, 1994.

[Bir98] BIRNBAUM, ROBERT: *„Mir ist auch mal die Hand ausgerutscht"*. Der Tagesspiegel, Seite 5, 12. November 1998.

[Bla93] BLAZE, MATT: *A Cryptographic File System for Unix*. In *1st ACM Conference on Communications and Computing Security*, Seiten 158–165, Fairfax, VA, 1993.

[BMW99a] BUNDESMINISTERIUM FÜR WIRTSCHAFT UND TECHNOLOGIE: *Mosdorf gibt Startschuß für Projekt Wahlen im Internet*, 3. März 1999. Pressemitteilung, online unter `http://www.bmwi.de/presse/1999/0303prm2.html`.

[BMW99b] BUNDESMINISTERIUM FÜR WIRTSCHAFT UND TECHNOLOGIE: *Initiative des Bundesministeriums für Wirtschaft und Technologie für mehr Sicherheit in der Informationsgesellschaft findet breite Unterstützung*, 1. März 1999. Pressemitteilung, online unter `http://www.bmwi.de/presse/1999/0301prm1.html`.

[BP99] BÖTTGER, CHRISTIAN und NIELS POLLEM: *Indianische Rauchzeichen. Aufbau einer Website mit Corporate Identity*. iX Magazin für professionelle Informationstechnik, (3):112, 1999.

[BRD97] *Begründung zur Verordnung zur digitalen Signatur*, 8. Oktober 1997. Fassung des Beschlusses der Bundesregierung, online unter `http://www.iid.de/rahmen/sigv_begr.html`.

[BRR97] BIZER, JOHANN, JOACHIM RIESS und ALEXANDER ROSSNAGEL: *Beschränkungen kryptographischer Verfahren sind verfassungswidrig*, 14. Januar 1997. Online unter `http://www.provet.org/kk/stenahme.htm`.

[Bru98] BRUNO, LEE: *Banking On Trust*. Data Communications, Seiten 43–47, 21. Mai 1998.

[BS97] BAUSPIESS, FRITZ und ALFRED SCHEERHORN: *Zertifikatswechsel und Schlüsselgültigkeiten*. Datenschutz und Datensicherheit, 21(6):334–340, 1997.

[BSI97] *Mit Sicherheit in die Informationsgesellschaft. Tagungsband 5. Deutscher IT-Sicherheitskongreß des BSI 1997*, Ingelheim, 1997. SecuMedia Verlag.

[BSI98] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *IT-Grundschutzhandbuch 1998. Maßnahmenempfehlung für den mittleren Schutzbedarf (CD-ROM)*, 1998.

[Bun97] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Kulturelle Beherrschbarkeit digitaler Signaturen*. SecuMedia-Verlag, Ingelheim, 1997.

[Cam98a] CAMPHAUSEN, INGMAR: *Entwurf und Implementierung eines generischen Verfahrens zum Abruf von Public-Key-Zertifikaten*, 16. Juni 1998. Studienarbeit, TU Berlin.

[Cam98b] CAMPHAUSEN, INGMAR: *Erfahrungen beim Aufbau einer bundesweiten Zertifizierungsinfrastruktur für den Individual Network e. V.* In BAUKNECHT, KURT et al. [BBPT98], Seiten 387–410. Online unter `http://www.in-ca.in-berlin.de/doc/papers/SIS98/sis98.zip`.

[Cam98c] CAMPHAUSEN, INGMAR: *Schlüsselzertifizierung mit PGP*. Datenschutz und Datensicherheit, 22(7):382–385, 1998. (Basiert in Teilen auf dem Vortrag zu [CD98]).

[Cam99] CAMPHAUSEN, INGMAR: *Entwurf eines Konzeptes für eine Zertifizierungsstelle für die Freie Universität Berlin*. Diplomarbeit, Technische Universität Berlin, März 1999. `http://userpage.fu-berlin.de/~ingmar/diplomarb/`.

[CD98] CAMPHAUSEN, INGMAR und LUTZ DONNERHACKE: *Probleme beim PGP-Einsatz in Zertifizierungsstellen und deren Lösung mit PGP2.6.3in und OpenPGP*. Seiten B1–B16, Hamburg, März 1998. Verein zur Förderung eines Deutschen Forschungsnetzes e.V. Online unter `ftp://ftp.fokus.gmd.de/pub/docs/platin/1998/camphausen-dfncert.ps`.

[CDFT98] CALLAS, JOHN, LUTZ DONNERHACKE, HAL FINNEY und RODNEY THAYER: *RFC 2440: OpenPGP Message Format*, November 1998.

[CDFT99] CALLAS, JON, LUTZ DONNERHACKE, HAL FINNEY und RODNEY THAYER: *OpenPGP Message Format*, Dezember 1999. Internet-Draft Network Working Group (work in progress) <draft-ietf-openpgp-rfc2440bis-00.txt>.

[CER94] DFN-CERT: *Informationsbulletin Public-Key-Infrastruktur*, 20. Dezember 1994. Informationsbulletin DIB-94:07, letzte Änderung: 15. März 1999, `http://www.cert.dfn.de/infoserv/dib/dib-9407.html`.

[Cer98] CERTCO, INC.: *Major financial institutions announce new company to provide businesses globally with a single electronic identity*, 21. Oktober 1998. Pressemitteilung.

[CES94] CROCKER, STEVE, DONALD EASTLAKE und JEFFREY SCHILLER: *RFC 1750: Randomness Recommendations for Security*, Dezember 1994.

[Con99] CONTINI, SCOTT: *The Factorization of RSA-140.* RSA Laboratories' Bulletin, (10), 8. März 1999. Online unter `ftp://ftp.rsa.com/pub/pdfs/bulletn10.pdf` bzw. Meldung unter `http://www.rsa.com/rsalabs/html/rsa140.html`).

[CP] CATTANEO, G. und G. PERSIANO: *Design and Implementation of a Transparent Cryptographic file system for Unix.* `http://mikonos.dia.unisa.it/tcfs`.

[CSA99] CHESKIN RESEARCH AND STUDIO ARCHETYPE/SAPIENT: *eCommerce Trust Study*, Januar 1999. Online abrufbar unter `http://www.studioarchetype.com/cheskin`.

[Cur98] CURRY, IAN: *Key Update and the Complete Story on the Need for Two Key Pairs*, Dezember 1998. Whitepaper, Entrust Technologies. Online abrufbar unter `http://www.entrust.com/resources/pdf/2keypairs10.pdf`.

[CZ97] *Frankreich strebt Key Recovery an – Nachschlüssel für Internet-Mails gefordert.* Computer Zeitung, (43):6, 23. Oktober 1997.

[CZ98a] *Banken vergeben digitale Signaturen. Konsortium ignoriert deutsches Gesetz.* Computer Zeitung, (44):1, 29. Oktober 1998.

[CZ98b] *Britanniens Tony greift nach dem Key.* Computer Zeitung, (19):1, 7. Mai 1998.

[CZ98c] *Cocom-Nachfolger: Einigung bei Krypto.* Computer Zeitung, (50), 1998.

[CZ98d] *Datendiebe sitzen in eigenen Reihen.* Computer Zeitung, (15):1, 9. April 1998.

[CZ98e] *Firmen klagen über E-Mail-Piraterie – Jede zehnte Nachricht wird mitgelesen.* Computer Zeitung, (25):20, 18. Juni 1998.

[CZ98f] *Hacker verursachen Milliardenschäden. Mitarbeiter begehen 70 Prozent der Delikte.* Computer Zeitung, (26):6, 25. Juni 1998.

[CZ98g] *Kanther bläst zum Krypto-Angriff.* Computer Zeitung, (7):1, 1998.

[CZ98h] *Linux soll Leid lindern – NT ist zu unsicher.* Computer Zeitung, (25):5, 18. Juni 1998.

[CZ98i] *Opera-Browser fügt 128-Bit-Slüssel zu.* Computer Zeitung, (42):15, 15. Oktober 1998.

[CZ98j] *Opera kündigt sich für Linux an.* Computer Zeitung, (28):11, 9. Juli 1998.

[CZ98k] *Sichere Daten – Neues Trust Center.* Computer Zeitung, (8):16, 19. Februar 1998.

[CZ98l] *Sicherheit für Domains.* Computer Zeitung, (37):5, 10. September 1998.

[CZ98m] *Sicherheit kein Thema.* Computer Zeitung, (36):21, 3. September 1998.

[CZ99a] *Caligula reißt ins PGP-Netz Löcher.* Computer Zeitung, Seite 1, 11. Februar 1999.

[CZ99b] *Kopenhagen führt erste Kontrollen durch: Krypto-Regelung bald auf EU-Basis.* Computer Zeitung, (1+2):3, 14. Januar 1999.

[DES77] *Data Encryption Standard*, Januar 1977. Federal Information Processing Standard (FIPS) PUB 46, National Bureau of Standards, US Dept. of Commerce.

[DFN97] *Digitale Signaturen anerkannt*, 20. November 1997. Presse-Information des DFN-Vereins, online unter `http://www.dfn.de/presse/dfn-presse/pm97-11-20.html`.

[DFS96] DAMKER, HERBERT, HANNES FEDERRATH und MICHAEL J. SCHNEIDER: *Maskerade-Angriffe im Internet. Eine Demonstration von Unsicherheit.* Datenschutz und Datensicherheit, (5):286–294, 1996.

[DH76] DIFFIE, WHITFIELD und MARTIN E. HELLMAN: *New Directions in Cryptography.* IEEE Trans. on IT, IT-22(6):644–654, November 1976.

[Die98] DIEDRICH, OLIVER: *Supercomputer aus Alpha-Rechnern: Linux-Cluster unter den 500 schnellsten Supercomputern*. c't Magazin für Computertechnik, (14):50, 1998.

[Dir98] DIREGGER, EKKEHARD: *Kryptographie und Menschenrechte. Beschränkungen der Kryptographie im Lichte der EMRK* [Europ. Menschenrechtskonvention]. Datenschutz und Datensicherheit, 22(1):28–31, 1998.

[DK98] DIEDRICH, OLIVER und JÜRGEN KURI: *Zwerge mit exotischen Freunden. Linux und OS/2 auf aktuellen Mittelklasse-Notebooks*. c't Magazin für Computertechnik, (12):182–189, 1998.

[DLLC97] DESAUTELS, PHILIP, PETER LIPP, BRIAN LAMACCHIA und YANG-HUA CHU: *DSig 1.0 Signature Labels. Using PICS 1.1 Labels for Digital Signatures*. In *WWWJournal* [WWW97], Seiten 29–48. Inzwischen W3C-Empfehlung *PICS Signed Labels (DSig) 1.0 Specification* vom 27. Mai 1998, online `http://www.w3.org/TR/REC-DSig-label/`.

[Dob96] DOBBERTIN, HANS: *The Status of MD5 After a Recent Attack*. CryptoBytes, 2(2), 1996.

[Don96] DONNERHACKE, LUTZ: *Medienbruch – Sicherung langfristiger Beweise*, 1996. Online unter `http://www.iks-jena.de/mitarb/lutz/logfile/`.

[Drä96] DRÄGER, DIETMAR: *Vertrauliche Kommunikation zwischen Bürger und Verwaltung*. Diplomarbeit, Technische Universität Berlin, November 1996. Online abrufbar unter `http://ig.cs.tu-berlin.de/da/041/`.

[DSB92] BERLINER DATENSCHUTZBEAUFTRAGTER: *Datenschutzrechtliche Anforderungen beim Einsatz von Laptops*, 2. Auflage, 1992. Broschüre der Reihe „Materialien zum Datenschutz“.

[DSB98] *Garstka: Europäische Datenschutzrichtlinie ist zu beachten*, 30. Oktober 1998. Pressemitteilung des Berliner Datenschutzbeauftragten, online unter `http://www.datenschutz-berlin.de/aktuelle/presse98/presse15.htm`.

[Eas99] EASTLAKE, DONALD: *RFC 2535: Domain Name System Security Extensions*, März 1999.

[Ebe98] EBELING, ADOLF: *US-Geheimdienst fängt europaweit E-Mails ab*. Heise Newsticker, 9. Januar 1998.

[Edw98] EDWARDS, MARK JOSEPH: *nFast/KM improves Web performance*. INFOWORLD, 24. August 1998.

[EFF98] *Cracking DES. Secrets of Encryption Research, Wiretap Politics & Chip Design*. O'Reilly & Associates, Juli 1998.
Details zum DES-Crack der Electronic Frontier Foundation online unter `http://www.eff.org/descracker/`.

[EFL+99] ELLISON, CARL M., BILL FRANTZ, BUTLER LAMPSON, RON RIVEST, BRIAN THOMAS und TATU YLONEN: *RFC 2693: SPKI Certificate Theory*, September 1999.

[Elk96] ELKINS, MICHAEL: *RFC 2015: MIME Security with Pretty Good Privacy (PGP)*, Oktober 1996.

[Ell97] ELLIS, J. H.: *The Story of Non-Secret Encryption*, 17. Dezember 1997. Angeblich von 1987, online unter `http://www.cesg.gov.uk/about/nsecret/ellis.htm`.

[Ell99] ELLISON, CARL M.: *RFC 2692: SPKI Requirements*, September 1999.

[ENT99] *Integration with Entrust/PKI Extends Adobe Acrobat 4.0 Digital Signature Technology to Protect Documents from Unauthorized Access or Alterations*, 16. Februar 1999. Pressemitteilung Entrust Technologies, online unter `http://www.entrust.com/news/1999/02_16_99_2.htm`.

[EU99] *RICHTLINIE 1999/93/EG DES EUROPÄISCHEN PARLAMENTES UND DES RATES über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen*, 13. Dezember 1999. Online unter `ftp://ftp.pca.dfn.de/pub/pca/docs/SigG/Europe/l_01320000119de00120020.pdf`.

[EUK97] EUROPÄISCHE KOMMISSION: *Ensuring security and trust in electronic communication – Towards a European framework for digital signatures and encryption*, Oktober 1997. COM(97)503.

[EUK98a] EU-KOMMISSION, GD XIII/03: *Ausschreibung über Zertifizierungsdienste zur Unterstützung der elektronischen Kommunikation zwischen den Dienststellen der Kommission und Rechtspersonen im Zusammenhang mit FuE-Programmen der Gemeinschaft*, 25. August 1998. Online verfügbar via `http://www.cordis.lu/fifth/src/docs.htm`.

[EUK98b] EUROPÄISCHE KOMMISSION: *Proposal for a European Parliament and Council Directive on a common framework for electronic signatures*, 13. Mai 1998. COM(1998)297final.

[EUK99] RAT DER EUROPÄISCHEN KOMMISSION: *Draft for a European Parliament and Council Directive on a common framework for electronic signatures*, 25. Januar 1999. Online unter `http://www.dud.de/dud/files/egsigrwd3.zip`.

[EUP95] *Richtlinie 95/46/EG des Europäischen Parlaments und des Rates zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr*, 23. November 1995. Amtsblatt der Europäischen Gemeinschaften L 281, online unter `http://www2.echo.lu/legal/de/datenschutz/datensch.html`.

[Fed97] FEDERRATH, HANNES: *Schlüsselgenerierung in Trust Centern? Einseitig sicher ist nicht sicher genug*. Datenschutz und Datensicherheit, 21(2):98–99, 1997.

[FG98] FOX, DIRK und RÜDIGER GRIMM: *Entwurf einer EU-Richtlinie zu Rahmenbedingungen „elektronischer Signaturen"*. Datenschutz und Datensicherheit, 22(7):407–408, 1998.

[FHK95] FOX, DIRK, PATRICK HORSTER und PETER KRAIBEEK: *Grundüberlegungen zu Trust Centern*. In HORSTER, PATRICK [Hor95], Seiten 1–10. Online unter `http://www.secorvo.de/publikat/trustcen.htm`.

[FHPS99] FORD, WARWICK, RUSSELL HOUSLEY, TIM POLK und DAVID SOLO: *RFC 2459: Internet X.509 Public Key Infrastructure – Certificate and CRL Profile*, Januar 1999.

[FKK96] FREIER, ALAN O., PHILIP KARLTON und PAUL C. KOCHER: *The SSL Protocol Version 3.0*, 18. November 1996. <draft-freier-ssl-version3-02.txt>, Online-Dokument `http://home.netscape.com/eng/ssl3/`.

[For94] FORD, WARWICK: *The Need for Separate Key Pairs for Symmetric Key Transfer and Digital Signature*. Entrust Technologies, Februar 1994. Entrust Technologies White Paper, issue 1.0.

[Fox95] FOX, DIRK: *Automatische Autogramme*. c't Magazin für Computertechnik, Seiten 278–284, November 1995. `http://www.heise.de/ct/95/10/278/`.

[Fox98] FOX, DIRK: *Eine „PGP"-Policy für Unternehmen*. Datenschutz und Datensicherheit, 22(9):519–520, 1998.

[Fri99] FRISCHEISEN, SIMON: *Konzeption und prototypische Implementierung eines Systems für den organisatorischen Betrieb einer Zertifizierungs-Autorität (CA) am LRZ*. Diplomarbeit, TU München, August 1999. `http://wwwca.lrz-muenchen.de/dev/doc/Ausarbeitung/final/Ausarbeitung.ps.gz`.

[FSV98] FALTIN, UTE, WOLFGANG SCHNEIDER und URSULA VIEBEG: *SPHINX Pilotversuch Ende-zu-Ende-Sicherheit*. Abschlußbericht Phase 1, GMD – Forschungszentrum Informationstechnik GmbH, Institut für Telekooperationstechnik, Darmstadt, 31. Dezember 1998. Online abrufbar unter `http://www.darmstadt.gmd.de/SPHINX/AB1PDF.zip`.

[FUB99] FREIE UNIVERSITÄT BERLIN, ZENTRALEINRICHTUNG FÜR DATENVERARBEITUNG (ZEDAT): *ZEDAT in Zahlen*, 1999. Stand: 1. Januar 1999.

[Gat95] GATES, WILLIAM: *The Road Ahead*. Viking, 1995.

[Geh98] GEHRING, ROBERT: *Digitale Signaturen*. Diplomarbeit, Technische Universität Berlin, März 1998. Online (größtenteils) abrufbar unter `http://ig.cs.tu-berlin.de/ap/rg/002/`.

[Ger00] GERLING, RAINER W.: *Einführung von E-Mail und PGP in großen Unternehmen*. In *Proceedings 7. Workshop „Sicherheit in vernetzten Systemen"*, Seiten H1–H14, Hamburg, März 2000. DFN-CERT.

[Gil97] GILMORE, JOHN: *Security for the Domain Name System*. In *WWWJournal* [WWW97], Seiten 175–180.

[GNRS98] GRUFFERTY, SHARON, JACK NAGLE, GUIDO REINKE und PETER SYLVESTER: *Report to the Commission of the European Communities DG XIII/Infosec for the EUROTRUST PKI pilot service*. Techn. Report, Baltimore Technologies/PriceWaterhouse/edelweb, Juli 1998.

[Gri95] GRIMM, RÜDIGER: *Verfahren zur Sicherung der Vertraulichkeit, Authentizität und Integrität. Notwendigkeit von Zertifizierungsinstanzen*. In NUTZERGRUPPE HOCHSCHULVERWALTUNG IM DFN und HIS GMBH (Hrsg.): *DFN-Tagung 1995: „Sichere Datenübertragung in offenen Netzen"*, Seiten 112–131, August 1995.

[Gri97] GRIMM, RÜDIGER: *Rechts- und Zahlungssicherheit im Internet*. In KUBICEK, KLUMPP, MÜLLER, NEU und ROSSNAGEL (Hrsg.): *Jahrbuch Telekommunikation und Gesellschaft 1997*, Seiten 211–220. R.v. Decker's Verlag, Heidelberg, 1997.

[GS00] GIALOURIS, EVANGELOS und KLAUS SCHMEH: *Schlüsselgewalt*. iX Magazin für professionelle Informationstechnik, (2):93–96, 2000.

[Gut98] GUTMANN, PETER: *How to recover private keys for Microsoft Internet Explorer Internet Information Server, Outlook Express, and many others – or – Where do your encryption keys want to go today?*, 20. Januar 1998. Mail an die Cypherpunks-Mailingliste; online unter `http://www.cs.auckland.ac.nz/~pgut001/pubs/breakms.txt`.

[Hag96] HAGER, NICKY: *Secret Power. New Zealand's Role in the International Spy Network*. Craig Potton Publishing, PO Box 555, Nelson, New Zealand, 1996. Reprint 1996; kann unter `http://www.fas.org/irp/eprint/sp/index.html` bestellt werden.

[Hag97] HAGER, NICKY: *Exposing the Global Surveillance System*. CovertAction Quarterly, (59), Winter 1996/1997. Online unter `http://caq.com/CAQ59GlobalSnoop.html`; Auszug aus [Hag96].

[Ham98] HAMMER, VOLKER: *Wie nennen wir Infrastrukturen für die Schlüsselverwaltung?* Datenschutz und Datensicherheit, 22(2):91–92, 1998. Online unter `http://www.provet.org/iukdg/sis-sis.htm`.

[Hil98] HILTWEIN, JÖRG: *Stellungnahme für den Individual Network e.V. zur Frage der Haftung einer nicht-kommerziellen Zertifizierungsstelle*. E-Mail an I. Camphausen, 21. März 1998.

[HJJS98] HILLYER, BRUCE K., MARKUS JAKOBSSON, ARI JUELS und ELIZABETH SHRIVER: *A Practical Secure Physical Random Bit Generator*. In *5th ACM Conference on Computer and Communications Security*, Seiten 103–111, San Francisco, California, 3.–5. November 1998. ACM SIGSAC, ACM Press.

[HK98] HALEVI, SHAI und HUGO KRAWCZYK: *Public-key cryptography and password protocols*. In *5th ACM Conference on Computer and Communications Security*, Seiten 122–131, San Francisco, California, 3.–5. November 1998. ACM SIGSAC, ACM Press.

[Hoe97] HOEREN, THOMAS: *Haftung des Vereins zur Förderung eines Deutschen Forschungsnetzes e.V. als Online-Diensteanbieter*. Rechtsgutachten, DFN-Verein, Juli 1997. `ftp://ftp.dfn.de/pub/net/www/data/berichte/ber83.ps`.

[Hor95] HORSTER, PATRICK (Hrsg.): *Trust Center. Proceedings der Arbeitskonferenz Trust Center 95*. Vieweg, Braunschweig, 1995.

[Hor96] HOROWITZ, MARC: *A PGP Public Key Server*. Diplomarbeit, MIT, 24. Februar 1996. Online unter `http://www.mit.edu/afs/net.mit.edu/project/pks/thesis/paper/thesis.html`.

[How97] HOWLAND, GARY: *Fun and Games with PGP*, 8. August 1997. Online unter `http://www.hotlava.com/doc/fag-pgp/`.

[HP96] HUHN, MICHAELA und ANDREAS PFITZMANN: *Technische Randbedingungen jeder Kryptoregulierung*. Datenschutz und Datensicherheit, (1):23–26, 1996. Online unter `http://www.zerberus.de/texte/ccc/ccc95/div/pfitz/krypto.htm`.

[Hub98] HUBERTZ, JOHANNES: *Ins Allerheiligste. Projektbericht: AS400 sicher im Internet*. iX Magazin für professionelle Informationstechnik, (1):104–107, 1998.

[HW98] HORSTER, PATRICK und PETRA WOHLMACHER: *Grundüberlegungen zur Gestaltung von Sicherheitsinfrastrukturen*. In BAUKNECHT, KURT et al. [BBPT98], Seiten 139–168.

[IN97] INDIVIDUAL NETWORK E.V.: *Oberste Zertifizierungsstellen des Deutschen Forschungsnetzes und des Individual Network e.V. erkennen sich gegenseitig an*, 5. November 1997. Presseerklärung, online unter `http://www.in-ca.individual.net/presse/1997-11-05.in-ca`.

[inf96] INFINITY DAEMON9@NETCOM.COM / ROUTE@INFONEXUS.COM: *PGP Attacks*, Februar 1996. v0.50 [beta], dokumentiert von Bill Unruh unter `http://axion.physics.ubc.ca/pgp-attack.html`.

[ISO] *ISO/IEC DIS 11770-3 Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*.

[ISO96] *ISO/IEC 11770-1:1996 Information technology – Security techniques – Key management – Part 1: Framework*, 1996.

[ITU97] *ITU-T Recommendation X.509 (1997:E) The Directory: Authentication Framework*, Juni 1997. X.509v3 = ISO/IEC 9594-8. (Die wichtigen Struktur-Definitionen aus diesem Standard sind auch im Anhang von [FHPS99] wiedergegeben.).

[IW99] *Verschlüsselung inbegriffen*. Information Week, (3):12, 4. Februar 1999.

[JK99] JUN, BENJAMIN und PAUL KOCHER: *The Intel Random Number Generator*. Techn. Report, Intel Corporation, 22. April 1999. Cryptography Research, Inc. White Paper. Online unter `http://www.cryptography.com/intelRNG.pdf`.

[Jos99] *Discours et interventions*, 19. Januar 1999. Conférence de presse de Monsieur Lionel Jospin, Premier ministre, à l'issue du Comitée interministériel pour la société de l'information, Hôtel de Matignon. Online unter `http://www.premier-ministre.gouv.fr/PM/D190199.HTM`.

[Kal93] KALISKI, JR., BURT: *A Layman's Guide to a Subset of ASN.1, BER, and DER*. Technical Note, RSA Laboratories, 1. November 1993. Link auf `http://www.rsasecuritu.com/rsalabs/pkcs/`.

[Kar99] KAREPIN, ROLF: *Etliche Bugs werfen Haftungsfragen auf*. Computer Zeitung, (3):4, 21. Januar 1999.

[Kel99] KELM, STEFAN: *Signed in Germany? SigG-Zertifikate für's Volk*. Datenschutz und Datensicherheit, 23(9), 1999.

[Ken93a] KENT, STEPHEN T.: *Internet Privacy Enhanced Mail*. Comm. of the ACM, 36(8):48–60, August 1993.

[Ken93b] KENT, STEVEN: *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*, Februar 1993.

[Kha97] KHARE, ROHIT: *Web Security. A Matter of Trust*. In *WWWJournal* [WWW97], Seite 2. Editorial.

[KL96] KELM, STEFAN und BRITTA LIEDTKE: *DFN-PCA – Sicherheit im Netz. Das Pilotprojekt „Policy Certification Authority" – PCA*. DFN-Mitteilungen, (40):12–16, März 1996. Online verfügbar unter `http://www.rtb-nord.uni-hannover.de/dfn/mitteilungen/html/heft40/s11/s11.html`.

[KL99] KELM, STEFAN und BRITTA LIEDTKE: *DFN-PCA Die World Wide Web Policy*. Universität Hamburg, 4. Januar 1999. Zertifizierungsrichtlinien für das PCA-Projekt, Version 0.90 (vorläufige Fassung), online unter `http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html`.

[KPS95] KAUFMAN, CHARLIE, RADIA PERLMAN und MIKE SPECINER: *Network Security*. Prentice Hall, März 1995. Prentice Hall Series in Computer Networking and Distributed Systems, ISBN 0-1306-1466-1.

[KR97] KHARE, ROHIT und ADAM RIFKIN: *Weaving A Web of Trust*. In *WWWJournal* [WWW97], Seiten 77–112.

[Kre99a] KREMPL, STEFAN: *Konzerne im Visier*. c't Magazin für Computertechnik, (4):182–184, 1999. Interview mit Abhörspezialist Hans-Georg Wolf über Lauschangriffe von Geheimdiensten auf Unternehmen, `http://www.heise.de/ct/99/04/182/`.

[Kre99b] KREMPL, STEFAN: *Kryptographie und die Kraft des Faktischen*. Der Tagesspiegel, Seite 30, 24. Januar 1999.

[KS94] KIM, GENE H. und EUGENE H. SPAFFORD: *The Design and Implementation of Tripwire: A File System Integrity Checker*. In *Proceedings of the 2nd ACM Conference on Computer and Communication Security*, 1994.

[KS98] KELSEY, JOHN und BRUCE SCHNEIER: *Cryptographic Support for Secure Logs on Untrusted Machines*. In *The Seventh USENIX Security Symposium Proceedings*, Seiten 53–62. USENIX Press, Januar 1998.

[Kuh98] KUHN, MARKUS: *In die Röhre geguckt. Unerwünschte Abstrahlung erlaubt Lauschangriff*. c't Magazin für Computertechnik, (24):90–97, 1998.

[Lan98] LANGE, BARBARA: *Zehn kleine Fingerlein. Login per Fingerabdruck: BioMouse GINA 2.6.* iX Magazin für professionelle Informationstechnik, (10):58, 1998.

[Lei99] LEICH, STEFFEN: *Schlüsselkind. Freie PGP-Implementierung GnuPG.* iX Magazin für professionelle Informationstechnik, (3):94–98, 1999.

[Lic97] LICHT, RAINER: *Sicherheitsmaßnahmen beim Ausscheiden von Mitarbeitern.* KES Zeitschrift für Kommunikations- und EDV-Sicherheit, (2):15–19, 1997.

[LIN] *LINUX MAGAZIN.* Herausgegeben von Rudolf Strobl. Linux-Magazin Verlag, München. Homepage: `http://www.linux-magazin.de`.

[Lin98] LINDSEY, CHARLES H.: *Critique of PGP Key Generation*, 4. August 1998. Online `http://www.cs.man.ac.uk/~chl/critique.html`.

[LM91] LAI, X. und J. L. MASSEY: *A Proposal for a New Block Encryption Standard.* In *Proceedings of Eurocrypt '90*, Nr. 473 Reihe *LNCS*, Seiten 389–404. Springer, 1991.

[Lom98] LOMONACO, JR., SAMUEL J.: *A Quick Glance at Quantum Cryptography*, 8. November 1998. Online unter `http://xxx.lanl.gov/abs/quant-ph/9811056`.

[LR96] LAMPSON, BUTLER und RONALD L. RIVEST: *SDSI – A Simple Distributed Security Infrastructure*, 15. September 1996. Version 1.0 `http://theory.lcs.mit.edu/~rivest/sdsi10.ps`, neuere Versionen: siehe SDIS-Homepage `http://theory.lcs.mit.edu/~cis/sdsi.html`.

[Luc96] LUCKHARDT, NORBERT: *Horchideen. Datenspionage durch kompromittierende Abstrahlung.* c't Magazin für Computertechnik, (6):70–73, 1996.

[Luc97] LUCKHARDT, NORBERT: *Schlüsselringen. Auf der IFA geht die c't-Krypto-Kampagne in die zweite Runde.* c't Magazin für Computertechnik, (9):276, 1997.

[Luc98] LUCKHARDT, NORBERT: *Nächste Runde. Die c't-Krypto-Kampagne geht weiter.* c't Magazin für Computertechnik, (6):32–33, 1998. `http://www.heise.de/ct/98/06/032/`.

[Luc99a] LUCKHARDT, NORBERT: *c't-Krypto-Kampagne. Die Zertifizierungsaktion geht ins dritte Jahr.* c't Magazin für Computertechnik, (6):99, 15. März 1999. `http://www.heise.de/ct/99/06/099/`.

[Luc99b] LUCKHARDT, NORBERT: *Digitale Signatur: Jetzt gilt's.* Heise Newsticker, 26. Januar 1999. `http://www.heise.de/newsticker/data/nl-26.01.99-000/`.

[Mac97] MACHEFSKY, IRA: *A First Look at Cryptographic Accelerators.* Techn. Report, Giga Information Group, 29. Dezember 1997.

[Mac98] MACHEFSKY, IRA: *A Total Economic Impact Analysis of Two PKI Vendors: Entrust and VeriSign.* Techn. Report, Giga Information Group, Norwell, MA, USA, September 1998. Online abrufbar unter `http://www.entrust.com/resources/pdf/pki_tei_report.pdf`. Das „Gegenstück" der Firma Verisign findet man unter `http://www.verisign.com/whitepaper/enterprise/difference/introduction.html`.

[Mar99] MARTIUS, KAI: *Nachschlag. Domain Name System gegen Mißbrauch schützen.* iX Magazin für professionelle Informationstechnik, (2):108–113, 1999.

[Mau97] MAURIELLO, ERMELINDO: *TCFS: Transparent Cryptographic File System.* Linux Journal, Seiten 64–68, August 1997.

[Med99] MEDOSCH, ARMIN: *Überraschende Wendung in UK-Kryptopolitik.* Telepolis, 7. März 1999. `http://www.heise.de/tp/deutsch/inhalt/te/1821/1.html`.

[Mee99] MEEKS, BROCK N.: *The Privacy Hoax*. Comm. of the ACM, 42(2):17–19, Februar 1999.

[Moe98] MOECHEL, ERICH: *Kryptoexportkontrolle: Protestgruppen organisieren sich*. Telepolis, 8. Dezember 1998. `http://www.heise.de/tp/deutsch/inhalt/te/1708/1.html`.

[Moe99] MOECHEL, ERICH: *Virenattacken im Netz wirken wie biologische Waffen. Interview mit Bill Larson, Network Associates (NAI)*. Computer Zeitung, (9):25, 4. März 1999.

[MoP98] *Brief und Siegel per Chip. Bundesdruckerei entwickelt Internet-Codesystem*. Berliner Morgenpost, Seite 11, 16. August 1998.

[Mos97] MOSES, TIM: *Building a public key infrastructure – in-source or out-source?*, 1997. Whitepaper, Entrust Technologies.

[MR96] MILLER, JAMES und PAUL RESNICK: *PICS: Internet Access Control Without Censorship*. Comm. of the ACM, 39(10):87–93, Oktober 1996.

[MR98] MEYER, CARSTEN und JÜRGEN RINK: *Das große Messen*. c't Magazin für Computertechnik, (15):114–118, 1998.

[MS98] MÜLLER, GÜNTER und KURT-HERMANN STAPF (Hrsg.): *Mehrseitige Sicherheit in der Kommunikationstechnik: Band 2. Erwartung, Akzeptanz, Nutzung*. Addison-Wesley, Bonn, Reading Massachusetts, 1998.

[MW97] MRAZ, VIKTOR und KLAUS WEIDNER: *Falsch verbunden. Gefahr durch DNS-Spoofing*. c't Magazin für Computertechnik, (10):286–290, 1997. `http://www.heise.de/ct/97/10/286/`.

[NC98] *Zertifizierte Sicherheit zahlt sich aus*. Network Computing, Seiten 50–51, 14. Dezember 1998.

[Neh97] NEHL, ROLAND: *Schlüsselgenerierung in Trust Centern? Vertrauen durch Trust Center*. Datenschutz und Datensicherheit, 21(2):100–101, 1997.

[Neu99] NEUMANN, PETER G.: *Robust Open-Source Software*. Comm. of the ACM, 42(2):128, Februar 1999.

[ötv99] *E-Mail lockt Spione*. das ötv-magazin, (1–2):24, 1999.

[PCAa] DFN-PCA: *Leitfaden zur CA-Zertifizierung*. Online unter `http://www.pca.dfn.de/dfnpca/certify/pgp/instca.html`.

[PCAb] DFN-PCA: *Teilnehmer-Erklärung zwischen DFN-PCA und CA*. Online unter `http://www.pca.dfn.de/dfnpca/certify/ca.html`.

[Per98] PERLUSZ, STEFANO: *Survey on Trust Enhancing Products in the Security Domain*, 27. Oktober 1998. `http://ntsta.jrc.it/dsa/surviv.htm`.

[Pet97] PETZEL, ERHARD: *Sichere Authentisierung*. KES, Zeitschrift für Kommunikations- und EDV-Sicherheit, (1):50–56, 1997.

[Pos00] *Offizieller Startschuss für das Trustcenter – Deutsche Post Signtrust erhält Genehmigungsurkunde*, 24. Februar 2000. Pressemitteilung der Deutschen Post AG, `http://www.deutschepost.de/postag/news/new0002/ne000206.html`.

[Rab70] RABBOW, ARNOLD: *dtv-Lexikon politischer Symbole A–Z*. Deutscher Taschenbuch Verlag, München, 1970.

[Rai98] RAISON, ANDRÉ: *Schlüsselfertig. Kryptographie-Lösungen für Intra- und Internet*. iX Magazin für professionelle Informationstechnik, (12):120–123, 1998.

[Rec98] RECKINGER, GABRIELE: *Haftpflichtversicherungen müssen auf Internet-Risiken hin angepaßt werden*. Computer Zeitung, (42):4, 15. Oktober 1998.

[Rei97a] REIF, HOLGER: *Herr der Zertifikate*. iX Magazin für professionelle Informationstechnik, (4):176–183, 1997. `http://www.heise.de/ix/artikel/1997/04/176/`.

[Rei97b] REISEN, ANDRE: *Anforderungen an sichere Trustcenter vor dem Hintergrund des Signaturgesetzes*. In *BSI-Kongreß* [BSI97], Seiten 27–44.

[Rei97c] REISEN, ANDRE: *Signaturgesetz und -verordnung: Die ersten Schritte*, 1997. `http://www.bsi.bund.de/pbdigsig/download/siswv.pdf`.

[Rei98] REIF, HOLGER: *Winnetou und Old SSLeay*. iX Magazin für professionelle Informationstechnik, (7):128–132, 1998. `http://www.heise.de/ix/artikel/1998/07/128/`.

[Ric97] RICHARDSON, MATTHEW: *PGP Digital Timestamping Service*, 29. April 1997. Online unter `http://www.itconsult.co.uk/stamper/stampinf.htm`.

[Rin97] RINK, JÜRGEN: *Alice im Wunderland. Quantenrechner: Auf dem Sprung zur Realität?* c't Magazin für Computertechnik, (3):110–116, 1997.

[Roß97] ROSSNAGEL, ALEXANDER: *Das Signaturgesetz. Eine kritische Bewertung des Gesetzentwurfs der Bundesregierung*. Datenschutz und Datensicherheit, 21(2):75–81, 1997.

[Roe98] ROESSLER, THOMAS: *PGP – „Kryptographie fürs Volk"*. Datenschutz und Datensicherheit, 22(7):377–381, 1998.

[Rot98a] ROTH, HARALD: *Exportkontrollen für Verschlüsselungsprodukte*. Datenschutz und Datensicherheit, 22(1):8–13, 1998.

[Rot98b] ROTH, HARALD: *Exportkontrollen für Verschlüsselungsprodukte (Teil II)*. Datenschutz und Datensicherheit, 22(2):81–85, 1998.

[RS97a] REITER, MICHAEL K. und STUART G. STUBBLEBINE: *Path Independence for Authentication in Large-Scale Systems*. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, April 1997.

[RS97b] REITER, MICHAEL K. und STUART G. STUBBLEBINE: *Toward Acceptable Metrics of Authentication*. In *Proceedings New Security Paradigms Workshop*, Seiten 48–60, Langdale, Cumbria, UK, 23.–26. September 1997. ACM.

[RSA78] RIVEST, RONALD L., ADI SHAMIR und LEONARD M. ADLEMAN: *A Method for obtaining Digital Signatures and Public Key Cryptosystems*. Comm. of the ACM, 21(2):120–126, Februar 1978.

[RTP97] REGULIERUNGSBEHÖRDE FÜR TELEKOMMUNIKATION UND POST: *ENTWURF Maßnahmenkatalog für digitale Signaturen – auf Grundlage von SigG und SigV –*, 1997. Nach Angaben des Bundesamtes für Sicherheit in der Informationstechnik, Stand 18.11.1997, Version 1.0, online unter `http://www.bsi.bund.de/aufgaben/projekte/pbdigsig/download/kat.pdf`.

[RTP98] REGULIERUNGSBEHÖRDE FÜR TELEKOMMUNIKATION UND POST: *Maßnahmenkatalog für Zertifizierungsstellen nach dem Signaturgesetz*, 1998. Nach Angaben des Bundesamtes für Sicherheit in der Informationstechnik, Stand: 15. Juli 1998, online unter `http://www.regtp.de/imperia/md/content/tech_reg_t/digisign/6.pdf`.

[Rud97] RUDLOFF, DIRK: *Sichere elektronische Post für Behörden in NRW*. In *BSI-Kongreß* [BSI97], Seiten 461–472.

[Rue95] RUEPPEL, RAINER A.: *Revocation and Revocation Certificates*. In *Proceedings of the EDI Trusted Third Party Workshop*, Barcelona, Februar 1995.

[Sch96] SCHNEIER, BRUCE: *Applied Cryptography*. Wiley, 2. Auflage, 1996.

[Sch98] SCHMIDT, JÜRGEN: *Die Geheimniskrämer. Verschlüsselnde Dateisysteme für Linux*. c't Magazin für Computertechnik, (19):228–230, 1998.

[Sch99] SCHRÖDER, BURKHARD: *Frankreich verschlüsselt*. Der Tagesspiegel, Seite 30, 24. Januar 1999.

[Sen98] SENDEREK, RALF: *Die Sicherheit des geheimen Schlüssels*, August 1998. Online-Dokument `http://www-users.informatik.rwth-aachen.de/~senderek/certify/schutz.html`.

[SH98a] SCHULZKI-HADDOUTI, CHRISTIANE: *Mehr Sicherheit mit „Digitalen Ausweisen"*. Der Tagesspiegel, Seite 30, 26. Juli 1998.

[SH98b] SCHULZKI-HADDOUTI, CHRISTIANE: *Überwachungskontinent Europa. EU-Pläne für Überwachungsmaßnahmen enthüllt*. c't Magazin für Computertechnik, (25):48–49, 1998. `http://www.heise.de/ct/98/25/048/`.

[SH98c] SCHULZKI-HADDOUTI, CHRISTIANE: *Umrüstung. Kryptographie gilt weiterhin als Waffe*. c't Magazin für Computertechnik, (26):52–53, 1998. `http://www.heise.de/ct/98/26/052/`.

[SH99] SCHULZKI-HADDOUTI, CHRISTIANE: *Scheideweg. BSI soll 'bürgernahe Beratungsstelle' werden*. c't Magazin für Computertechnik, (5):216–218, 1999.

[Sho97] SHOR, PETER W.: *Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM Journal on Computing, 26:1484ff, 1997. Online unter `http://xxx.lanl.gov/abs/quant-ph/9508027`.

[Sig97a] *Gesetz zur digitalen Signatur (Signaturgesetz – SigG)*, 22. Juli 1997. Artikel 3 des *Gesetz zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste (Informations- und Kommunikationsdienste-Gesetz – IuKDG)*, BGBl. I S. 1870, 1872, online unter `http://www.iid.de/rahmen/iukdgbt.html`.

[Sig97b] *Verordnung zur digitalen Signatur (Signaturverordnung – SigV)*, 22. Oktober 1997. BGBl. I S. 2498, online unter `http://www.iid.de/rahmen/sigv.html`, Begündung dazu unter `http://www.iid.de/rahmen/sigv_begr.html` (siehe [BRD97]).

[Sim99] SIMPSON, SAM: *PGP DH vs. RSA FAQ*, 20. November 1999. Version 1.5, `http://www.scramdisk.clara.net/pgpfaq.html`.

[Spa99] SPANIER, KURT: *LDAP und die Ablage von PGP-Schlüsseln im X.500-Directory*. In *Proceedings 6. Workshop „Sicherheit in vernetzten Systemen"*, Seiten H1–H14, Hamburg, März 1999. DFN-CERT. Online unter `ftp://ftp-ambix.directory.dfn.de/ambix/Cert-6/LDAP-PGP.ps.gz`.

[SPI96] *Ein Schlüssel für den Staat*. Der Spiegel, (52):16, 1996.

[SPI98] *Geheime Botschaften im Netz*. Der Spiegel, (8):22–24, 1998.

[Sta94] STALLINGS, WILLIAM: *Network and Internetwork Security*. Prentice Hall, 1994.

[StW97] *European Union and FBI Launch Global Surveillance System*. Report, Statewatch, PO Box 1516, London N16 0EW, UK, Februar 1997. Online unter `http://www.privacy.org/pi/activities/tapping/statewatch_tap_297.html`.

[SU97] SCHMEH, KLAUS und HUBERT UEBELACKER: *Zufallstreffer. Kryptographisch sichere Zufallsgeneratoren.* c't Magazin für Computertechnik, (14):220–223, 1997.

[SW] STUMM, FRANK-STEFAN und FRANK WEBER: *Maßnahmenempfehlungen: Infrastruktur für Zertifizierungsstellen (SigG/SigV).* Bundesamt für Sicherheit in der Informationstechnik, online unter `http://www.bsi.de/aufgaben/projekte/pbdigsig/download/tcinfra.pdf`.

[SW97] SCHOLVIN-WULFF, BARBARA: *Sicherheitsexperten streiten um einen Mail-Standard.* Computer Zeitung, (51+52):9, 18. Dezember 1997.

[Tim97] TIMM, BIRTE: *Signaturgesetz und Haftungsrecht.* Datenschutz und Datensicherheit, 21(9):525–528, 1997.

[Tsp99a] *Regulierer gibt digitale Unterschrift frei.* Der Tagesspiegel, Seite 18, 26. Januar 1999.

[Tsp99b] *Verbraucherschützer raten zur Verschlüsselung.* Der Tagesspiegel, Seite 30, 12. Februar 1999.

[Ull60] ULLRICH, KARL-HEINZ: *Das Goldene Buch der Zitate.* SÜD-WEST Verlags- und Vertriebs-GmbH, München, 1960.

[USC99] ARBEITSGRUPPE USCA DES RECHENZENTRUMS DER UNIVERSITÄT STUTTGART: *Universität Stuttgart Certification Authority: Certification Policy*, 16. März 1999. Version 1.1.1, online unter `http://ca.uni-stuttgart.de/POLICY/`.

[Ver97] VERISIGN, INC.: *VERISIGN CERTIFICATION PRACTICE STATEMENT IN SUPPORT OF VERISIGN'S PUBLIC CERTIFICATION SERVICES CLASS 1–3 DIGITAL IDs / CERTIFICATES.* Mountain View, CA, 30. Mai 1997. Version 1.2, ISBN 0-9653555-2-7, online unter `http://www.verisign.com/repository/CPS1.2/`.

[Wal97] WALZ, STEFAN: *Ist „Privatheit" im Multimedia-Zeitalter noch zu retten?* In *BSI* [Bun97], Seiten 25–30.

[Wie98] WIENER, MICHAEL J.: *Performance Comparison of Public-Key Cryptosystems.* CryptoBytes, 4(1):1,3–5, 1998.

[Wil97] WILLIAMS, RANDALL T.: *The passphrase FAQ, version 1.04*, 23. März 1997. Online abrufbar unter `http://www.stack.nl/~galactus/remailers/passphrase-faq.html`.

[Wil98] WILZOPOLSKI, AXEL: *Power im Pinguin-Land. MkLinux und Linux-PMac auf Power Macintosh.* iX Magazin für professionelle Informationstechnik, (5):96–98, 1998. Online unter `http://www.heise.de/ix/artikel/1997/05/096/`.

[Wol98] WOLF, HANS-GEORG: *Kompromittierende Emissionen*, 29. Dezember 1998. Vortrag auf dem Chaos Computer Congress 1998, Berlin.

[Wri98a] WRIGHT, STEVE: *An appraisal of technologies for political control.* Report für das Europ. Parlament, Directorate General for Research, Omega Foundation, Manchester, 6. Januar 1998. PE 166 499, online unter `http://www.heise.de/tp/deutsch/inhalte/te/1393/s1.html`.

[Wri98b] WRIGHT, STEVE: *An appraisal of technologies for political control.* STOA Interim Study, Executive Summary, Omega Foundation, Manchester, September 1998. Online unter `http://www.europarl.eu.int/dg4/stoa/en/publi/166499/execsum.htm`.

[WWW97] *Web Security. A Matter of Trust.* The World Wide Web Journal. O'Reilly, Sommer 1997.

[Zie97] ZIESCHANG, THILO: *Sicherheitsrisiken bei der Schlüsselzertifizierung.* Datenschutz und Datensicherheit, 21(6):341–343, 1997.

[Zie98] ZIESCHANG, THILO: *Fisch und Chips*. iX Magazin für professionelle Informationstechnik, (9):48–52, 1998.

[Zim95] ZIMMERMANN, PHILIP: *The Official PGP User's Guide*. MIT Press, 1995. Deutsche Übersetzung von Abel Deuring und Christopher Creutzig, erschienen im Verlag Art d'Ameublement, 4. Aufl., Bielefeld, 1999, ISBN 3-9802182-9-5, Online-Fassung unter `http://www.foebud.org/pgp/`.

[Zoo54] ZOOZMANN, RICHARD: *Zitatenschatz der Weltliteratur*. Verlag Praktisches Wissen, Berlin, 8. Auflage, 1954.

[Zsa99] ZSAKO, JANOS: *RFC 2726: PGP authentication for RIPE database updates*, Dezember 1999.

Informationen zur DFN-PCA

Kontakt

DFN Policy Certification Authority (DFN-PCA)
Universität Hamburg
FB Informatik – RZ
Vogt-Kölln-Straße 30
D-22527 Hamburg

Telefon: 0 40 / 4 28 83 - 22 62
Telefax: 0 40 / 4 28 83 - 22 41
E-Mail: `dfnpca@pca.dfn.de` (PGP-Key 0xE77ADB85, s. nächste Seite)
`http://www.pca.dfn.de/dfnpca/`

Detaillierte Informationen zu allen aktuellen Zertifikaten und Widerrufslisten der DFN-PCA erhalten Sie unter:

`http://www.pca.dfn.de/dfnpca/keys.html`
`ftp://ftp.pca.dfn.de/pub/pca/keys/`

Schlüsselinformationen

PGP-Schlüssel

Low-Level Policy

PCA (Wurzelzertifikat):

Benutzer-ID:

`DFN-PCA, CERTIFICATION ONLY KEY (Low-Level: 1999-2000) <not-for-mail>`

Schlüssel-ID: `F7E87B9D`
Schlüssellänge: 2048 Bits – Erstellungsdatum: `1998/12/29`
Fingerprint: `65 70 72 74 B5 E0 3F F0  EA 7C AB E4 46 5F B8 B2`

User-CA:

Benutzer-ID:

DFN-User-CA, CERTIFICATION ONLY KEY (Low Level: 1999-2000)
<http://www.pca.dfn.de/dfnpca/>
Schlüssel-ID: 890C0981
Schlüssellänge: 2048 Bits – Erstellungsdatum: 1999/01/05
Fingerprint: 6C 02 E0 5C 67 3A 41 59 BC A6 C8 00 19 D4 58 49

DFN-PCA (Kommunikationsschlüssel):

Benutzer-ID:

DFN-PCA, ENCRYPTION KEY <dfnpca@pca.dfn.de>

Schlüssel-ID: E77ADB85
Schlüssellänge: 2048 Bits – Erstellungsdatum: 1998/04/21
Fingerprint: 48 BE 74 79 7F 5D BD 4C 65 2B 98 53 DD 5A 03 05

X.509-Zertifikate

World Wide Web Policy

DFN Top Level CA (Wurzelzertifikat):

```
Certificate:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=DE, O=Deutsches Forschungsnetz, OU=DFN-PCA, \
            CN=DFN Top Level Certification Authority/Email=certify@pca.dfn.de
    Validity
        Not Before: Oct 29 18:03:10 1998 GMT
        Not After : Dec 31 18:03:10 2001 GMT
    Subject: C=DE, O=Deutsches Forschungsnetz, OU=DFN-PCA, \
             CN=DFN Top Level Certification Authority/Email=certify@pca.dfn.de
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (2048 bit)
    X509v3 extensions:
        Key Usage:                   keyCertSign cRLSign
        Basic Constraints:           allowed to act as a CA !
        Identifier: Certificate Type
            Critical: no
            Certified Usage:
                SSL CA
                E-mail CA
                Object Signing CA
    MD5 Fingerprint=45:BB:9B:C8:8A:A4:84:8B:2D:A0:08:8F:9E:B6:B8:10
```

DFN Server CA:

```
Certificate:
    Version: 3 (0x2)
    Serial Number: 2 (0x2)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=DE, O=Deutsches Forschungsnetz, OU=DFN-PCA, \
            CN=DFN Top Level Certification Authority/Email=certify@pca.dfn.de
    Validity
        Not Before: Nov  3 16:47:23 1998 GMT
        Not After : Nov  2 16:47:23 2000 GMT
    Subject: C=DE, O=Deutsches Forschungsnetz, OU=DFN-PCA, \
             CN=DFN Server Certification Authority/Email=certify@pca.dfn.de
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (2048 bit)
    X509v3 extensions:
        Key Usage:                   keyCertSign cRLSign
        Basic Constraints:           allowed to act as a CA !
        Identifier: Certificate Type
            Critical: no
            Certified Usage:
                SSL CA
    MD5 Fingerprint=89:95:E8:3F:CD:EF:9B:62:5B:5C:FE:05:27:D1:34:D8
```