

„Konzeption und prototypische Umsetzung
von Authentifizierungsverfahren und
Kommunikationsschnittstellen für das Identity
Management System CIDAS* unter
besonderer Berücksichtigung mobiler
identifizierbarer Datenträger“

DIPLOMARBEIT

Zur Erlangung des akademischen Grades

Diplom-Informatiker (FH)

des Fachbereichs Informatik und Medien an der
Fachhochschule Brandenburg

Vorgelegt von

Jan Tobias Mühlberg

geb. 30.12. 1978 in Potsdam

Betreut von

Prof. Dr. Friedrich-Lothar Holl und *Prof. Dr. Barbara
Wiesner*

Brandenburg an der Havel, 3. März 2004
(überarbeitet am 19. August 2004)

*Configurable Internet Directory and Authentication Service

Copyright © 2004, Jan Tobias Mühlberg.

Jan Tobias Mühlberg, Grabenstraße 5a 14776 Brandenburg an
der Havel, Germany
E-Mail: muehlber@fh-brandenburg.de

Kopieren, Verbreiten und/oder Modifizieren ist unter den Bedingungen der GNU Free Documentation License, Version 1.2 oder einer späteren Version, veröffentlicht von der Free Software Foundation, erlaubt. Es gibt keine unveränderlichen Abschnitte, keinen vorderen Umschlagtext und keinen hinteren Umschlagtext. Eine Kopie des Lizenztextes ist in Anhang D, „GNU Free Documentation License“, enthalten.

Copyright © 2004, Jan Tobias Mühlberg.

Jan Tobias Mühlberg, Grabenstraße 5a 14776 Brandenburg an
der Havel, Germany
email: muehlber@fh-brandenburg.de

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in appendix D, „GNU Free Documentation License“.

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Überblick und Abstract	IV
Grundlegende Terminologie (Glossar)	V
1 Einleitung	1
1.1 Aufgabenstellung	3
1.2 Methodik und Aufbau dieser Arbeit	4
2 Ausgangssituation und Problemstellung	8
2.1 Dienst-Strukturen und vernetzte Umgebungen	9
2.1.1 Dienste und Dienstanbieter	9
2.1.2 Benutzung von Diensten	10
2.2 Vertrauen und Sicherheit	12
2.2.1 Reduzierung von Systemunsicherheit: Open-Source	13
2.2.2 Identität von Benutzern und Benutzerverwaltung	14
2.2.3 Identity-Management	16
2.3 Verfahren zur Autorisierung	19
2.3.1 Identifikation und Authentifizierung	19
2.3.2 Textuelle Authentifizierungsverfahren	20
2.3.3 Biometrische Authentifizierungsverfahren	22
2.3.4 Kryptographische Authentifizierungsverfahren	24
2.4 Bestehende Lösungen und deren Probleme	29
2.4.1 Microsoft Passport	29
2.4.2 Das Liberty Alliance Project	31
3 Das Identity-Management-System CIDAS	34
3.1 Das Komponentenmodell von CIDAS	36
3.1.1 Unterstützte Anwendungen und	38
3.1.2 Single-Sign-On	38
3.1.3 Autorisierungsmethoden in CIDAS	39
3.1.4 Speicherung personenbezogener Daten	40
3.1.5 Special-Feature-Module	42

3.2	Betrachtung der Sicherheit von CIDAS	42
4	Verwendung passiver mobiler Speichermedien und asymmetrischer Kryptographie zur Benutzerauthentifizierung	44
4.1	Anforderungen an die Authentifizierung	45
4.1.1	Verwendbare Speichermedien	45
4.1.2	Sicherheit der Daten auf einem Speichermedium . . .	47
4.1.3	Einsetzbarkeit in heterogenen Umgebungen	47
4.1.4	Verwendung von asymmetrischer Kryptographie . . .	47
4.1.5	Bindung der Authentifizierung an eine Interaktion des Benutzers	49
4.1.6	Einfache Bedienbarkeit, Single-Sign-On	49
4.1.7	Offene Spezifikationen und Quellen	50
4.2	Konzeption des Authentifizierungsverfahrens	51
4.2.1	Identifizierung von Datenträgern	51
4.2.2	Auswahl des Authentifizierungsverfahrens	54
4.2.3	Ablauf einer Authentifizierung	56
4.2.4	Auf einem Authentifizierungsmedium zu speichernde Daten	62
4.2.5	USB-Memory-Sticks	65
4.2.6	CD-ROMs	69
4.2.7	Passive Transponder	71
4.3	Angriffsmöglichkeiten und zusätzliche Sicherungsmechanismen	75
4.3.1	Angriffe und Angriffsszenarien	75
4.3.2	Zusätzliche Sicherheitsmechanismen	77
4.3.3	Identifizierung von Client-Installationen	78
4.3.4	Signierte Client-Software	80
4.3.5	Maßnahmen zur Schadensreduzierung	80
4.3.6	Umgang mit inkorrekten Identifizierungsdaten	83
5	Konzeption eines Kommunikationsprotokolls für CIDAS	84
5.1	Designentscheidungen zum Protokollentwurf	86
5.2	Formale Spezifikation des CIDAS-Protokolls	88
5.2.1	Datenrepräsentation	88
5.2.2	Verbindungsaufbau	89
5.2.3	Identifizierungsmethoden	90
5.2.4	Authentifizierungsmethoden	91
5.2.5	Sicherheitslevel	94
5.2.6	Token	95
5.2.7	Fehler und Systemverhalten bei Fehlern	97
5.2.8	Aufbau einer Nachricht	98

5.2.9	Definition der Nachrichtentypen, Protokollablauf	103
5.2.10	Beschreibung der verwendbaren XML-Tags	116
6	Prototypische Umsetzung des Authentifizierungsverfahrens und des Kommunikationsprotokolls	127
6.1	Designentscheidungen	129
6.2	Prototypische Umsetzung des Kommunikationsprotokolls . . .	133
6.3	Implementierungsansatz für das Authentifizierungsprotokoll .	135
6.3.1	Datenspeicherung auf Authentifizierungsmedien	135
6.3.2	Erzeugung und Größe von Zufallsdaten und Zeitstempeln	140
6.3.3	Umsetzung des Authentifizierungsverfahrens mit Open- PGP	141
7	Zusammenfassung	143
7.1	Ausblick und weiterführende Arbeiten	145
	Abbildungsverzeichnis	147
	Tabellenverzeichnis	148
	Literaturverzeichnis	149
	Index	160
	Anhänge	166
A	Eine Beispielsitzung des CIDAS-Protokolls	166
B	cidas_1-0.dtd	169
C	cidas_1-0.h	181
D	GNU Free Documentation License	184
E	Ehrenwörtliche Erklärung	193

Überblick

In der vorliegenden Arbeit setzt sich der Autor mit der Problematik der Verwaltung und Autorisierung von Benutzern informationstechnologischer Systeme auseinander. Es werden derzeit verwendete Verfahren vorgestellt und versucht, deren Schwachstellen aufzuzeigen. Näher eingegangen wird auf das Identity Management System CIDAS. Im Hauptteil der Arbeit stellt der Autor ein im Rahmen von CIDAS einsetzbares und auf der Verwendung passiver mobiler Speichermedien in Verbindung mit asymmetrischer Kryptographie basierendes Authentifizierungsverfahren und das zwischen CIDAS-Server und -Client verwendete Kommunikationsprotokoll vor.

Abstract

Within the present work the author deals with problems related to the management and authorization of users of information technology systems. A review on presently used systems as well as a discussion on their flaws is given. Furthermore the identity management system CIDAS is explained in detail. The main part of the work deals with the conceptional design of an authentication protocol based on asymmetric cryptography and passive, mobile storage media. Beside this the conception of a communication protocol to be used between CIDAS-server and -client is explained.

Grundlegende Terminologie (Glossar)

Um dem Leser das Verstehen dieser Arbeit zu erleichtern, werden im Folgenden einige grundlegende, häufig verwendete Begriffe erläutert. An der Stelle ihres ersten Auftretens, sind diese Begriffe nochmals durch die Anmerkung ^(Glossar) gekennzeichnet.

A

Angriff

Unter einem Angriff gegen ein Authentifizierungssystem wird jeder absichtliche manuelle oder automatisiert durchgeführte Versuch eines Benutzers (des Angreifers) verstanden, Identifizierungs- bzw. Authentifizierungsdaten eines Dritten zukompromittieren oder sich als dieser auszugeben (vgl. Schneier, [99, S. 11]).

Angriff, aktiver

Unter einem aktiven Angriff werden Versuche verstanden, bei denen durch das Einschleusen falscher oder modifizierter Datenpakete in eine Kommunikationsverbindung versucht wird, die Autorisierung zu erlangen (vgl. RFC 1704, [58]).

Angriff, passiver

Ein passiver Angriff (eng. *Passive Attack*) zeichnet sich dadurch aus, daß keine Daten in die Kommunikationsverbindung eingeschleust werden. Die Kommunikation wird lediglich überwacht, mitgehört. Die empfangenen Nachrichten können vom Angreifer zu einem späteren Zeitpunkt und innerhalb einer validen Sitzung wiederverwendet werden (vgl. RFC 1704, [58, S. 2 f.]).

Angriff, Wiederholungs-

Wiederholungsangriffe engl. *Replay Attack* gegen Authentifizierungssysteme basiert auf dem Aufzeichnen von validen Nachrichten, die später erneut gesendet werden. Jedes konstante und elektronisch übertragene Authentifizierungsdatum kann hierbei zur nachträglichen Erzeugung authentisch wirkender Nachrichten verwendet werden (vgl. RFC 1704, [58, S. 2 f.]).

E

EEPROM

EEPROM steht für „Electrical Erasable Programable Read Only Memory“, also einen nichtflüchtigen, elektrisch beschreib- und löschbaren Speicher, wobei es eine physikalisch bedingte Obergrenze für die Anzahl der schreibenden bzw. löschenden Zugriffe gibt (vgl. Rankl et al. [88, S. 745]).

Einweg-Funktion

Siehe \rightarrow HASH-Funktion

Entschlüsselung

Die Entschlüsselung, auch Dechiffrierung, ist das Umkehrverfahren zur \rightarrow Verschlüsselung. Der \rightarrow Geheimtext wird in den \rightarrow Klartext zurücktransformiert (vgl. Fuhrberg, [56, S. 81]).

G

Geheimtext

Der Geheimtext, auch Chiffre, ist das Ergebnis der \rightarrow Verschlüsselung eines \rightarrow Klartextes mit einem Schlüssel. Er ist erst nach der \rightarrow Entschlüsselung wieder lesbar (vgl. Fuhrberg, [56, S. 81]).

H

HASH-Funktion

HASH-Funktionen sind \rightarrow kryptographische Verfahren, die für einen Eingabewert beliebiger Länge einen Ausgabewert fester Länge liefern, aus dem der Eingabewert nicht mehr rekonstruiert werden kann (vgl. Schneier, [96, S. 30 f.]).

K

Klartext

Im Gegensatz zum → Geheimtext enthält der Klartext Daten, die nicht durch → kryptographische Verfahren geschützt sind.

kryptographisches Verfahren

Kryptographische Verfahren dienen dazu, die Vertraulichkeit, Integrität und Authentizität von Daten sicherzustellen (vgl. Schneier, [96, S. 1 f.]).

O

OpenPGP

Siehe RFC 2440 [43]. OpenPGP kann ebenso wie → PEM Integrität, Authentizität und Vertraulichkeit einer Nachricht sichern. Die Identität der Benutzer wird jedoch nicht durch eine zentrale Stelle garantiert, vielmehr unterschreiben die Benutzer ihre öffentlichen Schlüssel gegenseitig. Das Ergebnis dessen ist ein sog. „Web of Trust“.

P

PEM

PEM – „Privacy Enhancement for Internet Electronic Mail“ wird in RFC 1421, 1422, 1423 und 1424 (vgl.[73, 68, 36, 67]) spezifiziert, ist kompatibel mit dem in X.509 [44] beschriebenen Authentifizierungs-Framework und dient zum Schutz vertraulicher Informationen vor dem Zugriff Dritter. PEM verwendet Zertifikate nach dem X.509v3 Standard. Die Identität eines Benutzers wird durch das Signieren seines Zertifikates durch eine vertrauenswürdige Certificate-Authority garantiert.

Protokoll, Kommunikations-

Ein Kommunikationsprotokoll ist eine Vereinbarung über den geordneten Ablauf einer Kommunikation, die von beiden miteinander kommunizierenden Parteien eingehalten werden. Protokolle sind bei der

Kopplung von Computern zu Netzwerken unverzichtbar (vgl. Claus et al., [45, S. 517]).

Protokoll, kryptographisches

Unter einem kryptographischen Protokoll wird eine Reihe von Arbeitsschritten verstanden, die unter Einbeziehung zweier oder mehrerer Kommunikationspartner dazu dienen, ein *rightarrow* kryptographisches Verfahren umzusetzen (vgl. Schneier, [96, S. 22 f.]).

R

ROM

ROM: Read Only Memory, ein Speicher, der lediglich einmal, während des Herstellungsprozesses, beschrieben und danach nur noch gelesen werden kann (vgl. Claus et al. [45, S. 567]).

S

Schlüsselmaterial

Bei der \rightarrow Verschlüsselung und \rightarrow Entschlüsselung von Nachrichten werden Schlüssel benötigt. Unter einem Schlüssel werden in diesem Zusammenhang eine arbiträre Menge von Zeichen verstanden. In Abhängigkeit von der Länge dieser Zeichenkette gibt es unterschiedlich viele mögliche Schlüssel; deren Gesamtzahl wird als Schlüsselraum bezeichnet (vgl. Schneier, [96, S. 3]).

Signatur, digitale

Digitale Signaturen garantieren die Authentizität einer Nachricht. Hierfür werden Verfahren der asymmetrischen Kryptographie verwendet, detaillierte Erläuterungen finden sich bei Schneier [96, S. 34-41] und Saolmaa [91]. Signaturen enthalten häufig einen Zeitstempel, der den Zeitpunkt, zu dem die Nachricht erstellt wurde, angibt.

T

Transaktion, digitale

„Unter einer digitalen Transaktion werden“, entsprechend Petrovic et al. [82], „alle über elektronische Netze durchgeführten Aktivitäten im Zuge eines Leistungsaustausches von der Informations- über die Vereinbarungs- bis hin zur Abwicklungsphase verstanden.“

Transponder, passiver

Unter dem Begriff „passive Transponder“ werden in dieser Arbeit „Radio Frequency Identification Devices“, kurz RFIDs verstanden. Dabei handelt es sich um verhältnismäßig kleine Geräte mit etwas Speicher und einer kontaktlosen Schnittstelle entsprechend ISO 14443 [15] und ISO 15693 [16].

Trust-Center

Unter einer Zertifizierungsstelle, auch Trust-Center genannt, wird allgemein eine glaubwürdige Instanz verstanden, die im digitalen Geschäftsverkehr die Identität Dritter garantiert (vgl. Fuhrberg, [56, S. 100])

V

Verschlüsselung

Bei der Verschlüsselung, auch Chiffrierung, wird der → Klartext unter Verwendung eines speziellen Algorithmus und eines Schlüssels in den → Geheimtext umgewandelt (vgl. Fuhrberg, [56, S. 81]).

Verzeichnisdienst

Ein Verzeichnisdienst (engl. *directory service*) ist ein Dienst, der sehr schnellen Zugriff auf in Datenbanken gespeicherte Informationen erlaubt. Die Datenbank stellt hierbei das Verzeichnis dar, der Dienst beschreibt die Zugriffsschnittstelle zu diesem Verzeichnis (vgl. Wilcox, [110, S. 10])

X

X.509

Siehe → PEM

Z

Zertifikat

Unter einem Zertifikat wird ein öffentlicher → Schlüssel verstanden, für dessen Vertrauenswürdigkeit beispielsweise ein → Trust-Center garantiert (vgl. Fuhrberg, [56, S. 98]).

Kapitel 1

Einleitung

*There's security that really makes us safer and security
that only lets us feel safer, with no reality behind it.*
— Bruce Schneier, *Beyond Fear* ([99, S. 10])

Einleitung

Während noch vor 25 Jahren Computernetzwerke als eine „akademische Kuriosität“ galten, verschaffen sich heutzutage Millionen von Menschen, nach Pötzsch [86, S. 15] waren es im Jahr 2000 bereits mehr als 34 Millionen allein in Deutschland, täglich und auf einfachste Weise Zugang zum Internet (vgl. Tannenbaum, [103, S. 13]). Das Internet, derzeit das einzige weltweit verfügbare Computernetz, wird sowohl in professionellen Einsatzumgebungen als auch im privaten Bereich zur Geschäftsabwicklung, zur Kommunikation zwischen Teilnehmern via E-Mail, Chat oder Videokonferenz, zur Verbindung räumlich entfernter Filialen von Unternehmen – kurzum zur Beschaffung und zum Austausch digitaler Informationen aller Art – genutzt (vgl. Fuhrberg, [56, S. 1 f.] und Eckert, [52, S. V f., 57-59]).

Eckert [52, S. 58] schreibt weiterhin, daß durch die „Öffnung des Internets für eine breite Anwenderschaft von Nicht-Spezialisten, die keine oder nur rudimentäre Kenntnisse über die mit Internet [...] verbundenen Gefährdungen besitzen“, die Zahl der möglichen Opfer von Computerkriminalität ansteigt. Entsprechend Petrovic et al. [82] erlaubt es der Mangel an Sicherheit einem Teilnehmer, sich zu seinem eigenen Vorteil und zum Schaden des Kommunikationspartners zu verhalten. Auch kann, so Petrovic et al. weiter, sich das zugrundeliegende informationstechnologische System als nicht funktionsfähig erweisen und damit das Zustandekommen digitaler Transaktionen^(Glossar) verhindern. Insbesondere an dem daraus resultierenden gegenseitigen Mißtrauen potentieller Transaktionspartner und am nicht vorhandenen Vertrauen in die angewandten Technologien scheitert heute häufig das Zustandekommen digitaler Transaktionen. Bemühungen, diese Transaktionen sicherer zu machen und somit Vertrauen aufzubauen, sind zwar immer notwendig, leider jedoch niemals ausreichend (vgl. Petrovic et al., [82]).

Um dem Problem des mangelnden Vertrauens und der Unüberschaubarkeit der verwendeten Sicherheitsmechanismen zu begegnen, wurden verschiedene neue Ansätze entwickelt. Hierzu gehören beispielsweise Trust-

Center^(Glossar). Neben diesen entwickeln verschiedene Anbieter Identity-Management-Systeme, die eine einfache Verwaltung großer Mengen personenbezogener Daten erlauben und dabei dem verwalteten Personenbestand unkomplizierte Dateneinsicht, Synchronisation von Authentifizierungsdaten und Single-Sign-On ermöglichen (vgl. Yasin, [111] sowie Abschnitt 2.2.3 dieser Arbeit.).

CIDAS¹ ist ein solches Identity-Management-System. Es unterscheidet sich von Konkurrenzprodukten unter anderem durch seine offenen Spezifikationen, den offenen Quellcode, die Vielfalt seiner Einsatzmöglichkeiten und die hohe Sicherheit der verwendbaren Verfahren zur Identifizierung und Authentifizierung seiner Benutzer.

1.1 Aufgabenstellung

Im Rahmen dieser Diplomarbeit wird vorrangig ein auf der Verwendung mobiler identifizierbarer Speichermedien und asymmetrischer Kryptographie basierendes Authentifizierungsverfahren zum Einsatz im an der Fachhochschule Brandenburg entwickelten Identity Management System CIDAS konzipiert werden.

Darüber hinaus soll ein für die sichere Kommunikation zwischen Authentifizierungsserver und -client geeignetes Kommunikationsprotokoll unter Berücksichtigung der von Schäfer in [93] spezifizierten Besonderheiten des Systems CIDAS entworfen werden.

¹CIDAS steht für „Configurable Internet Directory and Authentication Service“. Aufbau und Funktionsweise des Systems sind primär von Schäfer in [93] erläutert, in Kapitel 3 wird das System kurz vorgestellt.

1.2 Methodik und Aufbau dieser Arbeit

Das Vorgehen des Autors zur Bearbeitung der im letzten Abschnitt benannten Aufgabenstellung spiegelt sich im Aufbau der vorliegenden Diplomarbeit wieder und entspricht grundsätzlich den ersten drei Stufen des Vorgehens bei der Softwareentwicklung nach dem Wasserfall-Modell (vgl. Sommerville, [100, S. 57 f.]). Nach diesem Modell durchläuft eine Software während ihres Entwicklungsprozesses nacheinander die folgenden Phasen:

- Analyse und Definition der Anforderungen
- System- und Softwareentwurf
- Implementierung und Komponententest
- Integration und Systemtest
- Betrieb und Wartung

Die Abschließenden Schritte *Integration und Systemtest* sowie *Betrieb und Wartung* werden in dieser Arbeit nicht betrachtet. In der Praxis überlappen sich die oben genannten Entwicklungsphasen häufig. Es werden Informationen ausgetauscht, während der Bearbeitung einer Stufe Probleme in den Ergebnissen einer vorhergehenden gefunden und diese erneut bearbeitet. Dieses Vorgehen führt bei der Umsetzung von Softwareprojekten oft zu erheblichen Kosten durch partielle Vorhabensabbrüche und häufige Neuanfänge (vgl. Sommerville, [100, S. 58]).

Als eine Alternative zum oben beschriebenen Vorgehen betrachtet der Autor die Entwicklungsmethode *Extreme Programming* (XP), beschrieben von Beck in [37]. XP stellt eine Quellcode-bezogene Entwicklungsmethode dar, bei der die gesamte Softwareentwicklung einschließlich des System- und Softwareentwurfs testorientiert während der Implementierung geschieht (vgl. Beck, [37, S. 9]). Diese Methode kann, so Beck, durch permanente Überarbeitungen des Quellcodes und damit des gesamten Software-Designs, mit dem oben dargestellten Problem umgehen, wirft jedoch neue Schwierigkeiten bei

der Konzeption umfangreicher Schnittstellen und Verfahren auf (vgl. Beck, [37, S. 151-158]).

Zur Bearbeitung der Aufgabenstellung bemühte sich der Autor, beide vorgestellten Entwicklungsmodelle in der vorliegenden Arbeit miteinander zu kombinieren. Abzüglich dieser Einleitung und mehrerer Anhänge gliedert sich die Arbeit in sechs Kapitel. In den Kapiteln 2 und 3, einschließlich Abschnitts 4.1, wird die bestehende Situation analysiert und Anforderungen an die zu entwickelnde Software definiert. Den Hauptteil der Arbeit machen die Kapitel 4 und 5 aus, in denen der Autor große Teile des zu implementierenden Systems konzipiert – die Phase des *System- und Softwareentwurfs*. Die Phase der *Implementierung und Komponententests* wird in Kapitel 6 beschrieben. In diesem Kapitel wendet der Autor die von Beck beschriebene Entwicklungsmethode für die Konzeption Software-interner Schnittstellen und die eigentliche Implementierung an.

Im Detail werden in den einzelnen Kapiteln die folgenden Themen behandelt:

Im ersten Teil, Kapitel 2, wird ein Überblick über bestehende Infrastrukturen in Computernetzwerken gegeben, die derzeitigen Vorgehensweisen bei der Autorisierung von Benutzern vorgestellt und deren Probleme erläutert. Es wird ferner auf moderne Identity-Management-Systeme eingegangen und anhand zweier Beispiele auf deren Schwächen hingewiesen. Insbesondere in der zweiten Hälfte des Abschnittes wird festgestellt, daß aufgrund der Vielfalt der Anwendungsfälle für Identity-Management-Systeme und der Probleme der vorgestellten Lösungen weitere Entwicklungsarbeiten nötig sind. Das Kapitel stellt also den derzeitigen Entwicklungsstand zu der bearbeiteten Thematik dar. Grundlage hierfür bilden intensive Literatur- und Internet-Recherchen.

In Kapitel 3 befaßt sich der Autor mit dem an der Fachhochschule Brandenburg entwickelten Identity-Management-System CIDAS. Es wird auf die

Vorzüge von CIDAS, verglichen mit den in Kapitel 2 dargestellten Systemen eingegangen und die von CIDAS gebotene Sicherheit analysiert. Die Auseinandersetzung mit CIDAS geschieht vorrangig auf Basis der Diplomarbeit von I. Schäfer ([93]), die die Grundlage für diese Arbeit darstellt. Verschiedene Gespräche mit der Projektleitung, F. Holl und I. Schäfer förderten das Verständnis des Autors.

Der erste Teil der Aufgabenstellung, die Konzeption eines auf asymmetrischer Kryptographie basierenden Authentifizierungsverfahrens, wird in Kapitel 4 bearbeitet. Der Autor stellt hierfür zuerst die Anforderungen an das Authentifizierungsverfahren dar, geht auf Möglichkeiten zur Identifizierung von Datenträgern und existierende, auf asymmetrischer Kryptographie basierende Verfahren zur Benutzerauthentifizierung ein und zeigt deren Probleme auf. Grundlage hierfür bilden wiederum verschiedene Gespräche mit den Leitern des Projektes CIDAS und die systematische Erarbeitung verschiedener Quellen. Das im Rahmen von CIDAS verwendbare Authentifizierungsverfahren wird am Abschnitt 4.2.3 konzipiert, erläutert und analysiert. Eine abschließende Betrachtung von Sicherheitsproblemen und Lösungsmöglichkeiten zu diesen bildet den Schluß des Kapitels.

Parallel zur Erarbeitung des Authentifizierungsverfahrens wird in der vorliegenden Arbeit ein Kommunikationsprotokoll für den Datenaustausch zwischen CIDAS-Server und Client spezifiziert, was unter Erläuterung der getroffenen Designentscheidungen in Kapitel 5 wiedergegeben wird. Die Grundlagen für dieses Protokoll sind von Schäfer in [93, S. 90-96] angegeben. Informationen über Lösungsansätze und Vorgehensweisen für den Entwurf von Kommunikationsprotokollen entnahm der Autor vorrangig den am Anfang des Kapitels aufgeführten Quellen. Der Protokollentwurf selbst basiert vorrangig auf persönlichen Entscheidungen des Autors.

Alle, eine Implementierung des Authentifizierungsverfahrens und des Kommunikationsprotokolls, sowie die tatsächliche prototypische Umsetzung des letzteren betreffenden Details, sind in Kapitel 6 dargestellt.

In Kapitel 7 wird die Arbeit zusammengefaßt und ein Ausblick auf bereits laufende bzw. notwendigerweise an CIDAS durchzuführende Arbeiten gegeben. Es wird auf zukünftige Diplomarbeiten mit thematischem Bezug zu CIDAS hingewiesen.

Die Anhänge der Arbeit enthalten einige, die Umsetzung des konzipierten Kommunikationsprotokolls betreffende Quellcodeauszüge und einen beispielhaften Ablauf einer CIDAS-Sitzung.

Kapitel 2

Ausgangssituation und Problemstellung

Jedes der letzten drei Jahrhunderte wurde von einer bestimmten Technologie beherrscht. Das achtzehnte Jahrhundert war die Blütezeit der großen mechanischen Systeme, die die industrielle Revolution mit sich brachte. Das neunzehnte Jahrhundert war das Zeitalter der Dampfmaschine. Im zwanzigsten Jahrhundert spielen das Sammeln, Verarbeiten und Verbreiten von Informationen die wichtigste technologische Rolle.

— Andrew S. Tannenbaum, *Computernetzwerke* ([103, S. 17])

Ausgangssituation und Problemstellung

In diesem Kapitel wird ein Überblick über die Struktur des Internets sowie derzeit verwendete Verfahren zur Verwaltung von Benutzern und zur Abwicklung digitaler Transaktionen gegeben. Es wird darüber hinaus auf gebräuchliche Systeme zur Identifizierung, Authentifizierung und Autorisierung von Benutzern eingegangen. Kernziel ist es, dem Leser die Probleme der derzeit eingesetzten Verfahren aufzuzeigen.

2.1 Dienst-Strukturen und vernetzte Umgebungen

Nach Fuhrberg [56, S. 5] sind Anwendungen wie das „World Wide Web“ derart aufgebaut, daß Informationen von einem Rechner, dem sogenannten *Server*, angeboten und von anderen Computern, den *Clients*, abgerufen werden können. In diesem Zusammenhang bezeichnet man die serverseitigen Teil einer Anwendungen, die primär für die Kommunikation mit den Clients verantwortlich ist, auch als *Dienste*, engl. *services* (vgl. Eckert, [52, S. 58]). Damit Rechnersysteme unterschiedlicher Hersteller, die möglicherweise auch mit unterschiedlicher Software betrieben werden, miteinander kommunizieren, d.h. jeweils bereitgestellte Dienste gegenseitig nutzen können, existieren feste Kommunikationsprotokolle^(Glossar) sowie verschiedene Meta-Dienste, die den Informationsaustausch koordinieren (vgl. Fuhrberg, [56, S. 6 f.]).

2.1.1 Dienste und Dienstanbieter

Bereits kurz nach der Spezifikation der grundlegenden Kommunikationsprotokolle wurden die ersten Dienste entwickelt. Diese erlauben, so Eckert in [52, S. 58], die relativ komfortable Nutzung der in einem Netzwerk verfügbaren Ressourcen.

Ein Dienst besteht nach Claus et. al. [45, S. 594 f.] immer aus einer Software, also einem Programmpaket, das für den Dienst zuständig ist und die

logische Komponente des Systems darstellt. Die *Server-Software* benötigt eine physikalische Umgebung, in der sie ausgeführt wird. Dies ist die *Server-Hardware*, üblicherweise ein Computer in einem Rechnernetz [ebd.]. Entsprechend dieses Modells können auf einer Server-Hardware mehrere Dienste, also die entsprechende Software, laufen [ebd.]. Entsprechend Claus et. al. [ebd.] wird mit dem Begriff „Server“ im Allgemeinen sowohl die Server-Software als auch der Server-Hardware bezeichnet. Um Mißverständnissen vorzubeugen wird der Begriff im Folgenden ausschließlich zur Bezeichnung der Server-Software verwendet. Die entsprechenden Computer werden im Rahmen dieser Arbeit „Server-Hardware“ genannt.

Dienste werden prinzipiell dadurch angeboten, daß ein Hardware-Server mit den entsprechend installierten Diensten an das Internet angeschlossen wird. Die installierten Dienste sind ab diesem Moment von allen anderen Computern des Netzes aus erreichbar. Soll der freie Zugriff eingeschränkt werden, muß die jeweilige Server-Software entsprechende Mechanismen, beispielsweise eine Passwort-Abfrage, implementieren (vgl. Claus et al. [45, S. 64 f.]).

2.1.2 Benutzung von Diensten

Auf Dienste im Internet wird immer über ein Kommunikationsprotokolle zugegriffen. Die Aufgabe dieses Protokolls ist es, Quell- und Zielcomputer korrekt zu adressieren und Übertragungsfehler zu erkennen und zu beheben (vgl. Fuhrberg, [56, S. 6]). Darüber hinaus muß sichergestellt werden, daß die zu übertragenden Nutzdaten, die meist auf mehrere Einzelnachrichten bzw. Datenpakete aufgeteilt übertragen werden, vollständig und in der richtigen Reihenfolge beim Empfänger ankommen ([ebd.]).

Um Nachrichten unabhängig vom Protokoll des jeweiligen Dienstes über ein Kommunikationsmedium vom Sender zum Empfänger zu übertragen, müssen diese laut Eckert [52, S. 59-62] nach einem weiteren, dem Übertragungsmedium angepassten Protokoll transformiert, übertragen und wie-

der zurück transformiert werden. Das ISO/OSI-Referenzmodell¹ legt hierfür standardisierte Vorgehensweisen fest. In dieser sind sieben Protokollschichten, beginnend mit der physikalischen Datenübertragung bis hin zur Anwendungsschicht, definiert. Entsprechend Eckert [52, S. 59] hat hierbei jede Schicht die Aufgabe, Daten der darüber liegenden Schicht für die darunter liegende Schicht vorzubereiten und an diese weiterzuleiten. In der untersten Schicht werden die Daten daraufhin von einem Computer zu einem anderen übertragen. Auf diesem werden die Daten nun von Protokollschicht zu Protokollschicht nach oben durchgereicht und zurücktransformiert.

Die weitaus meisten Dienste im Internet basieren auf dem Internet Protocol, kurz IP. Das Internet Protocol wird in RFC 791 [1] spezifiziert und implementiert Schicht drei des ISO/OSI-Referenzmodells (vgl. Fuhrberg, [56, S. 9]). Über weitere Zwischenschichten können Anwendungsprotokolle wie beispielsweise FTP² oder HTTP³ auf IP aufsetzen.

Aus Sicht eines Benutzers gestaltet sich der Zugriff auf einen Dienst heutzutage meist sehr einfach. Laut Eckert [52, S. 58] gibt es bereits seit 1993 für verschiedene Protokolle Client-Software mit verhältnismäßig einfach bedienbarer graphischer Benutzungsoberfläche. Der Benutzer muß dabei lediglich spezifizieren, welche Informationen er haben möchte. Dies geschieht üblicherweise über sogenannte „Uniform Resource Identifier“ entsprechend RFC 2396 [39], mittels derer ein Dienst, der zu verwendende Server und darüber hinausgehende Informationen angegeben werden können.

¹Das Modell wurde Ende der 70er-Jahre von der International Standardization Organization unter dem Titel „Open Systems Interconnection“, ([10]) veröffentlicht, vgl. hierzu Fuhrberg in [56, S. 7].

²FTP steht für „File Transfer Protocol“, ein häufig verwendetes Protokoll zur Übertragung von Dateien. Das Verfahren ist in RFC 959 [85] spezifiziert.

³Bei HTTP, dem „HyperText Transfer Protokoll“ handelt es sich um ein Verfahren zur Übertragung von Webseiten und vergleichbaren Daten. Das Protokoll ist in RFC 2616 [54] spezifiziert.

2.2 Vertrauen und Sicherheit

Der Hauptgrund für das Nichtzustandekommen digitaler Transaktionen ist, so Petrovic et al. in [82], der Mangel an Vertrauen in die Identität der Transaktionspartner und in die Zuverlässigkeit und Sicherheit des verwendeten Kommunikationssystems. Um diesem Problem zu begegnen, müssen Maßnahmen zur Reduktion der Unsicherheit der Transaktionspartner unternommen werden. Die Transaktionspartner müssen diese natürlich wahrnehmen und ihnen vertrauen [ebd.]. Dennoch wird immer eine Restunsicherheit zurückbleiben, die nur durch weiteren Vertrauensaufbau zwischen den Partnern sowie durch nachweislich korrektes Funktionieren des informationstechnologischen Systems kompensiert werden kann. Petrovic et al. schreibt weiter, daß derzeit vor allem durch rechtliche und technische Maßnahmen versucht wird, die Unsicherheit digitaler Transaktionen zu reduzieren, um damit die Bereitschaft der Anwender zu erhöhen, diese durchzuführen.

In Tabelle 2.1 auf Seite 13 sind die Hauptfaktoren der Unsicherheit zwischen Transaktionspartnern dargestellt⁴. Diese lassen sich in zwei Faktorengruppen, nämlich den Bereich der *Systemunsicherheit* und den der *Partnerunsicherheit*, unterteilen. Bei der Systemunsicherheit handelt es sich um auf ein konkretes oder abstraktes informationstechnologisches System bezogene Unsicherheiten für den Benutzer: Das System könnte beispielsweise ausfallen, fehlerhaft sein oder über keine klaren rechtlichen Regelungen verfügen [ebd.]. Darüber hinaus existieren *Partnerunsicherheiten*. Diese entstehen, so Petrovic et al., wenn ein Transaktionspartner einen Informationsvorsprung gegenüber dem anderen besitzt, üblicherweise also dadurch, daß ein Partner keine Möglichkeit hat, sich zweifelsfrei der Identität und der Vorgehensweisen des anderen zu versichern.

Diese Arbeit befaßt sich hauptsächlich damit, digitale Transaktionen durch die Reduzierung von System- und Identitätsunsicherheit sicherer zu machen. Die konkreten Ansatzpunkte hierfür sind in Kapitel 3 dargelegt.

⁴Die Darstellung ist einer Graphik aus Petrovic et al., [82] entlehnt.

Systemunsicherheit	Partnerunsicherheit
<ul style="list-style-type: none"> • Fehler in Hard- oder Software • Sicherheitslücken • Rechtsunsicherheit • Technologische Entwicklung 	<ul style="list-style-type: none"> • Mangelnder Datenschutz • Identitätsunsicherheit • ungewisse Qualität der Lieferanten und Produkte

Tabelle 2.1: Formen der Unsicherheit bei digitalen Transaktionen

2.2.1 Reduzierung von Systemunsicherheit: Open-Source

Schneier erläutert in [98, S. 334-342, 344], daß die Sicherheit eines Systems nichts mit seiner Funktionalität zu tun hat. Die Evaluation der Sicherheit eines Systems, so Schneier, ist lediglich über Langzeitanalysen des Systems durch Sicherheitsexperten realisierbar. Die einzige Möglichkeit, unabhängigen Experten eine Chance zur Analyse des Systems zu geben, besteht wiederum in der Offenlegung aller verwendeten Verfahren und implementierungstechnischen Details [ebd.].

Als Alternative zu diesem Vorgehen führt Schneier [98, S. 344] die Methode *security by obscurity*, also die Geheimhaltung aller Spezifikationen und Quellen des Systems an, wodurch diese zum Fundament der Sicherheitslösung werden: Werden sie, ob nun beabsichtigt oder durch Zufall, offengelegt, bricht das Sicherheitssystem möglicherweise schlagartig zusammen. Da für nach dieser Methode konzipierte Systeme eine objektive Analyse der tatsächlich gebotenen Sicherheit nicht möglich ist, werden sie nicht helfen, Anwender zur Durchführung digitaler Transaktionen zu ermutigen. Auch von Petrovic et al. [82] wird die Offenlegung der Funktionsweisen von Sicherheitssystemen als ein entscheidendes Kriterium zum Aufbau von Vertrauen angesehen.

Das Konzept der *Open-Source-Software* ist nicht neu, insbesondere seit der Gründung der *Free Software Foundation*⁵ 1984 finden organisierte Bestrebungen statt, Software mit offenen Quellen und ohne Weitergabebeschrän-

⁵Siehe hierzu <http://www.fsf.org>.

kungen zu veröffentlichen. Neben dem oben dargestellten Sicherheitsaspekt gibt es hierfür eine Reihe weiterer Gründe – Meretz gibt die Wichtigsten davon in [79] wieder. Eine umfassende Darstellung der gesamten Problematik würde den Rahmen dieser Arbeit sprengen.

Daß eine freie Weitergabe von Dokumentation und Quellcode für den Werdegang eines Projektes sehr vorteilhaft sein kann, zeigen verschiedene erfolgreiche Open-Source-Projekte. Resultierend aus der Offenlegung der Quellen können Fehler oder Sicherheitslücken in Open-Source-Software in der Regel schneller gefunden werden als in proprietären Entwicklungen. Belegt werden diese Fakten beispielsweise von Raymond et al. in [89] und von Wheeler in [108]. Insbesondere Wheeler und Schneier sehen in [107, S. 8-13] sowie in [97] die Offenlegung des Quellcodes und der Dokumentation der verwendeten Verfahren als zwingende Voraussetzung für die Konzeption sicherer kryptographischer Verfahren und die Entwicklung korrekter Software an.

2.2.2 Identität von Benutzern und Benutzerverwaltung

Das Internet hat sich von einem Informationsmedium für Wissenschaftler zu einem weltumspannenden Wirtschafts- und Unterhaltungsbetrieb gewandelt (vgl. Fuhrberg, [56, S. 1 f.]). Die Betreiber von Anwendungen⁶ im Internet müssen wissen, wer der Benutzer ist, der Informationen abrufen oder ein Produkt bestellt, sei es auch nur, um eine Bestellung korrekt ausführen zu können (vgl. Schmidetzki, [94]). Auch in institutionsinternen Netzen ist die Identifizierung von Benutzern zwingend erforderlich, beispielsweise zur Abbildung der unterschiedlichen Funktionen und Tätigkeitsbereiche der Mitarbeiter eines Unternehmens auf dessen Datenbestände oder Applikationen, wie es von Tannenbaum et al. in [104, S. 20] und der *Entrust Inc.* in einem Whitepaper über Identity Management ([66]) dargestellt wird. Aus diesem Grund implementiert ein großer Teil der Anwendungen in Netzwerken jeweils eine ei-

⁶Kostenlos bereitgestellte, frei zugängliche Informationen werden an dieser Stelle nicht berücksichtigt.

gene Benutzerverwaltung, aus deren Datenbestand die Benutzer identifiziert werden können (vgl. Schmidetzki, [94]). Konkrete für die Identifizierung verwendbare Verfahren werden in Abschnitt 2.3 vorgestellt.

Problematisch ist dieses Vorgehen in dreierlei Hinsicht: Zum einen müssen sich die Benutzer Identifizierungsinformationen für eine Vielzahl von Internet-Applikationen merken, was häufig dazu führt, daß die Benutzer für alle Systeme ein und dieselben Daten verwenden (vgl. Schneier, [98, S. 255-269], Kapitel 17: *The Human Factor*). Im Falle der Kompromittierung eines Systems sind damit auch alle anderen Konten des Benutzers kompromittiert. Viel zu häufig vertrauen Benutzer in diesem Punkt auch leichtgläubig Anwendungsbetreibern dahingehend, daß diese mit den übergebenen Daten korrekt umgehen und diese nicht möglicherweise gezielt für Angriffe nutzen (vgl. Schneier zitiert von Mann, [76]). Schneier weist auch in [98, S. 27 f.] explizit auf die Gefahr des Identitätsraubs, *Identity Theft* hin: „Why steal from someone when you can just become that person? It’s far easier, and can be much more profitable“. In [98, S. 50 f.] stellt er den Identitätsraub als eine schnell wachsende, organisierte Form des modernen Verbrechens dar.

Zum anderen ist es so, daß Benutzer in der Regel keine Möglichkeit haben, sich einen Überblick über die tatsächlich über sie gespeicherten Informationen, wo diese gelagert werden und wer auf sie zugreifen kann, verschaffen können⁷. Entsprechend dem Bundesdatenschutzgesetz (vgl. BDSG, [25, §6]) ist es jedoch vorgeschrieben, daß Benutzer über ihre Daten jederzeit Auskunft erhalten können müssen. Oben beschriebenes Vorgehen ist also aus datenschutzrechtlicher Sicht sehr bedenklich, selbst wenn der Schutz personenbezogener Daten vor unberechtigten Zugriffen (vgl. BDSG, [25, §5]), zwar möglicherweise gewährleistet ist, Anwender dies jedoch nicht nachvollziehen können.

Aus Sicht der Applikationsbetreiber entsteht bei der oben beschriebenen dezentralen Lösung natürlich auch immer das Problem, daß Daten an

⁷Es handelt sich hierbei um persönliche Erfahrungen des Autors.

mehreren verschiedenen Stellen gepflegt werden müssen. Der administrative Aufwand hierfür ist kaum abschätzbar, insbesondere vor dem Hintergrund, daß sich personenbezogene Benutzerdaten durch Umzüge, Heiraten und Ähnliches häufig ändern und Benutzer darüber hinaus dazu tendieren, ihre Identifizierungs- und Authentifizierungsdaten zu vergessen (vgl. Entrust Inc, [66]).

Zum Abbau von Unsicherheiten der Benutzer bezüglich des Umgangs mit personenbezogenen Daten bei digitalen Transaktionen, müssen Möglichkeiten existieren, Benutzer zuverlässig zu identifizieren und ihnen verlässliche Einsicht in alle über sie gespeicherten personenbezogenen Daten zu gewähren (vgl. Petrovic et al., [82]). Darüber hinaus müssen sie natürlich auch in die Lage versetzt werden, selbst darüber entscheiden zu können, ob überhaupt nichtanonymisierte Daten über sie gespeichert werden dürfen – zumindest, sofern diese nicht zur Geschäftsabwicklung zwingend erforderlich sind (vgl. Petrovic et al., [ebd.]).

2.2.3 Identity-Management

Ein verhältnismäßig neues Konzept zur Verwaltung von personalisierten Datenbeständen, insbesondere auch von Benutzerdaten stellt das *Identity-Management* dar. Unter diesem Begriff werden entsprechend Yasin [111] Lösungen zusammengefaßt, welche die Verwaltung von Benutzerdaten insbesondere durch die im Folgenden aufgeführten Anforderungen entsprechen.

Access-Management Grundsätzlich erlauben Identity-Management-Systeme die Verwaltung großer und schnell wachsender Benutzerdatenbestände. Dabei ist es möglich, unterschiedliche Zugriffsrechte zu vergeben und sowohl den internen als auch den externen Bestand⁸ an personenbezogenen Daten einer Firma oder Institution zu verwalten. Weil Identity-Management-

⁸Die Formulierung stammt aus Yasin, [111] – es sind sowohl Mitarbeiter (intern) als auch Kunden oder Partner (extern) gemeint.

Systeme dafür sehr schnell auf Datenbestände zugreifen müssen, beispielsweise um große Mengen von Autorisierungsanfragen zu bewältigen, basieren sie häufig auf einem Verzeichnisdienst^(Glossar).

Passwort-Reset Insbesondere bei der Durchsetzung strikter Passwort-Richtlinien zur Vermeidung schwacher, einfach zu erratender oder durch Wörterbuchangriffe zu brechender Passworte (vgl. Schneier, [96, S. 171 f.], entstehen häufig Kosten durch den Verlust von Zugangsdaten durch Benutzer und das darauffolgende manuelle Festlegen eines neuen Passwortes durch Service-Mitarbeiter. Identity-Management-Systeme bieten Benutzern die Möglichkeit, ihre Passworte im Verlustfall selbstständig zurückzusetzen. Üblicherweise erreicht ein Benutzer dies über einen Web-Browser oder ein interaktives Anruf-Beantwortungssystem. Der Benutzer wird hierbei durch die Beantwortung einer Menge von (möglicherweise vorher festgelegten) Fragen authentifiziert.

Passwort-Synchronisierung Darüber hinaus bieten Identity-Management-Systeme Möglichkeiten zur Synchronisierung von Authentifizierungsdaten verschiedener Anwendungen. Verstanden wird darunter ein Datenabgleich zwischen den einzelnen Applikationen mit dem Identity-Management-System, so daß der Benutzer anschließend bei allen Applikationen beispielsweise das gleiche Passwort verwenden kann. Betrachtet man die Sicherheit solcher Verfahren, ergeben sich dabei ähnliche Probleme wie bei der Verwendung des im nächsten Abschnitt beschriebenen Konzeptes *Single-Sign-On*. Es gilt zu beachten, daß die Authentifizierungsdaten hierbei zusätzlich während der Übertragung vom Identity-Management-System zu den Applikationen sowie durch die dezentrale Speicherung zusätzlichen Gefahren ausgesetzt sind.

Single-Sign-On Als die nächsthöhere Ebene der einem Benutzer zu bietenden Vereinfachungen kann Single-Sign-On verstanden werden. Single-Sign-On erlaubt einem Benutzer nach erfolgreicher Authentifizierung bei einem Authentifizierungsserver den Zugriff auf alle Applikationen, bei denen er zugriffsberechtigt ist, ohne sich jeweils einzeln anmelden zu müssen (vgl.

Yasin, [111] und Gerbich, [57]). Eine brauchbare Beschreibung der Ansätze und Ziele sowie die Dokumentation eines der ersten funktionierenden Single-Sign-On-Systeme, entworfen von Netscape Inc., ist unter [12] zu finden. Auch Fuhrberg erläutert in [56, S. 323] die Grundzüge des Konzeptes „Single-Sign-On“ kurz. Schneier weist in [98, S. 149 f.] darauf hin, daß Single-Sign-On Lösungen, vergleichbar mit dem in Abschnitt 2.2.2 dargestellten Problem der Verwendung gleicher Identifizierungs- und Authentifizierungsdaten eines Benutzers bei vielen verschiedenen Applikationen, einen *Single Point of Failure* schaffen: Wird der primäre Zugang eines Benutzers zu seinem Identity-Management-System kompromittiert, stehen dem erfolgreichen Angreifer auf einen Schlag sehr viele Türen offen. Schneier schreibt „It’s the difference between losing a single credit card and losing your entire wallet“. Diese Gefahr läßt sich aus Sicht des Autors lediglich, wenn überhaupt, durch die Verwendung entsprechend sicherer Verfahren zur Authentifizierung eingrenzen.

2.3 Verfahren zur Autorisierung

Zu den größten Herausforderungen in einem offenen IT-System gehört die korrekte und zweifelsfreie *Legitimierung*⁹ seiner Benutzer (vgl. RFC 1704, [58, 1, 7-13]). Entsprechend Petrovic et al. [82] ist sie von enormer Wichtigkeit für einen verbindlichen elektronischen Geschäftsverkehr oder die Verwaltung von Zugriffsrechten auf physikalische Ressourcen oder Informationen in Rechnernetzen.

2.3.1 Identifikation und Authentifizierung

Nach Schneier [99, S. 181-183] ist die Legitimierung als Resultat einer vorhergehenden und erfolgreich abgeschlossenen *Identifizierung* und *Authentifizierung* zu sehen. Dabei wird im Rahmen der Identifizierung dem zu legitimierenden Benutzer eine Frage der Art „Wer sind sie?“ gestellt. Der Benutzer beantwortet diese Frage durch das Übersenden von Identifizierungsdaten, beispielsweise seinen Namens oder der Nummer seines Personalausweises. Während der Authentifizierung muß der Benutzer nun beweisen, daß er tatsächlich der ist, für den er sich ausgibt. Computergestützte Systeme werden hierfür üblicherweise Passworte verwenden, im „realen Leben“ stellt häufig der Vergleich des Paßfotos auf dem Personalausweis mit dem Gesicht der sich ausweisenden Person den Authentifizierungsvorgang dar. War die Authentifizierung erfolgreich, d.h. konnte die zur identifizierende Person ihre Identität beweisen, wird während der nun folgenden Autorisierung festgestellt, was der Authentifizierte tun darf, ob und in welcher Weise er berechtigt ist, auf Datenbestände zuzugreifen oder ob er einen Staat verlassen und in einen anderen einreisen darf.

Schneier führt in [99, S. 185, 188] Geldautomaten, wie sie häufig in Bankfilialen zu finden sind, als ein Beispiel für ein Autorisierungssystem, in dem Identifizierung und Authentifizierung voneinander getrennt sind, an: Beim Einführen der Bankkarte in den Automaten liest dieser die Bankdaten und

⁹Die Begriffe „Legitimierung“ und „Autorisierung“ werden im Rahmen dieser Arbeit als gleichbedeutend behandelt.

die Kontonummer aus dem Magnetstreifen der Karte aus. Diese Ziffernfolgen identifizieren die Bank und den jeweiligen Kunden eindeutig. Die nachfolgende Authentifizierung basiert in diesem Fall auf zwei Faktoren, zum einen wird der bereits festgestellte Fakt, daß der Benutzer die Bankkarte besitzt berücksichtigt, zum anderen muß er die korrekte Geheimzahl wissen und eingeben. In diesem Zusammenhang werden die Begriffe der *something he has*- und der *something he knows*-Authentifizierung verwendet.

Im realen Leben werden Identifizierung und Authentifizierung oft miteinander vermischt. Sowohl Schneier [99, S. 184] als auch Behrens et al. [38, S. 9] weisen darauf hin, daß das wichtigste Autorisierungssystem der Spezies Mensch auf persönlichem Kontakt basiert – eine Person wird anhand physiologischer Merkmale identifiziert und gleichzeitig authentifiziert.

Ebenso gibt es auch Systeme, in denen die Authentifizierung gleichzeitig mit der Legitimierung abläuft. Beispielsweise ist der Schlüssel zu einem mechanischen Schloß gleichzeitig Authentifizierungszeichen¹⁰ und Autorisierung. Der Schlüssel öffnet schließlich eine Tür und autorisiert den Besitzer damit, den Raum zu betreten (vgl. Schneier, [99, S. 188]).

2.3.2 Textuelle Authentifizierungsverfahren

Textuelle Authentifizierungsmethoden kamen mit der Verbreitung informationstechnologischer Systeme nahezu überall zum Einsatz (vgl. Claus et al., [45, S. 64]). Sie zeichnen sich dadurch aus, daß die zu authentifizierenden Benutzer ihre Identität durch die Eingabe von nur ihnen bekannten Informationen beweisen, wobei lediglich die Eingabe selbst, nicht die Art in der sie erfolgt (siehe Abschnitt 2.3.3), ausgewertet wird. Textuelle Authentifizierungsverfahren sind folglich *something he knows*-Verfahren.

Beispiele für diese Verfahrensklasse sind die Eingabe von Passwörtern, Geheimzahlen oder vergleichbaren Daten. Schneier weist in [99, S. 188] darauf

¹⁰Es handelt sich hierbei um eine *something he has*-Authentifizierung.

hin, daß Systeme, die bei der Verwendung textueller Authentifizierungsverfahren die Identifizierung mit der Authentifizierung vermischen und nicht-geheime Daten wie den Mädchennamen der Mutter, Geburtsdaten oder die letzten vier Stellen der Sozialversicherungsnummer als Authentifizierungsdaten verwenden, oftmals signifikante Unsicherheiten aufweisen und zu leichtfertig mit personenbezogenen Daten umgehen.

Aus Sicht des Autors sind an dieser Stelle insbesondere die „Telefonische Notenabfrage“, beschrieben von Remus in [90] und die Online-Bibliotheksanmeldung der Fachhochschule Brandenburg (nicht dokumentiert) als abschreckende Beispiele zu erwähnen. Die Identifizierung erfolgt in beiden Fällen über nicht geheime Identifizierungsnummern, nämlich die Matrikel- oder die Bibliothekskontennummer, die Authentifizierung über einen Teil des Geburtsdatums des Kontoinhabers. Im Fall der Bibliotheksanmeldung haben die Benutzer nicht einmal die Möglichkeit, dieses „Passwort“ zu ändern.

Das generelle Problem der textuellen Authentifizierungsverfahren besteht vor allem darin, daß sowohl Identifizierungs- als auch die zugehörigen Authentifizierungsdaten dem Authentifikator bekannt sein müssen. In der Praxis werden diese Daten meist auf einem Datenträger zusammenhängend gespeichert (vgl. Single UNIX Specification Version 3, [24]). Wenn sich ein Benutzer anmeldet, werden die eingegebenen und im Klartext^(Glossar) übertragenen mit den bereits gespeicherten Daten verglichen und bei Übereinstimmung eine Erfolgsmeldung zurückgegeben. Gelingt es bei diesem Vorgehen jemandem, die Kommunikationsverbindung abzuhören oder sich direkten Zugriff auf die Datenbasis zu verschaffen, ist das Sicherheitssystem kompromittiert (vgl. RFC 1704, [58, S. 1, 3 f.]). Textuelle Authentifizierungsverfahren sind dementsprechend anfällig für passive Angriffe^(Glossar) und Wiederholungsangriffe^(Glossar).

Laut Fuhrberg [56, S. 373-391] werden, um diese Verwundbarkeiten der textuellen Authentifizierungsverfahren zu umgehen, insbesondere im Bereich des Online-Banking, also der Abwicklung von Bankgeschäften über das In-

ternet, häufig Einmal-Passworte verwendet. Der Benutzer authentifiziert sich hierbei zum einen durch die Eingabe eines statischen Passwortes, zusätzlich aber noch mit einem nur für jeweils eine Transaktion gültigen Passwort ([ebd.]). Ein gebräuchliches Verfahren zur Erzeugung und Verwendung von Einmal-Passwörtern wird in RFC 2289 ([59]) vorgestellt, auf mögliche aktive Angriffen^(Glossar) wird in Fuhrberg, [56, S. 322] eingegangen.

2.3.3 Biometrische Authentifizierungsverfahren

Bereits in Abschnitt 2.3.1 wurde darauf hingewiesen, daß Menschen einander üblicherweise über physiologische Merkmale identifizieren und gleichzeitig authentifizieren. Bereits in der Bibel wird ein Beispiel hierfür angegeben: In [6, Richer:12; 1-7] besiegen die Gileaditen die Efraimiten und besetzen daraufhin die Jordanübergänge in das Land Efraim. Daraufhin zwangen sie jeden, der die Übergänge benutzen wollte, das Wort „Schibbolet“ (heb. für Ähre oder Wasserflut) auszusprechen. Weil die Efraimiten kein „sch“ in ihrer Sprache hatten und lediglich einen „s“-Laut aussprechen konnten, wurden sie identifiziert¹¹.

Bei biometrischen Authentifizierungsverfahren handelt es sich also um Verfahren, bei denen Eigenschaften eines Benutzers, die untrennbar¹² mit diesem verbunden sind, ausgewertet werden (vgl. Schneier, [99, S. 187]) – es wird also eine *something he has*-Authentifizierung durchgeführt. Nach Behrens et al. [38, S. 13] sind beispielsweise die Fingerbildererkennung, die Handerkennung, die Gesichts- und Iriserkennung, aber auch die Sprecher- und Schrifterkennung mögliche Verfahren.

Bei der Verwendung biometrischer Merkmale zur Identifizierung oder Authentifizierung gilt es immer zu beachten, daß die Auswahl an vorhandenen

¹¹Die Autorisierung bestand in diesem Fall darin, daß die „Benutzer“ erschlagen wurden: „Auf diese Weise fielen damals an den Jordananfurten 42000 Männer aus Efraim.“ Vgl. [6, Richer:12; 7].

¹²In vielen Fällen ist eine Abtrennung des merkmalsbehafteten Körperbereichs natürlich zumindest physikalisch möglich.

biometrischen Merkmalen im Gegensatz zu Passwörtern oder Schlüsseln begrenzt ist (vgl. Behrens et al. [38, S. 3]). Schlimmer noch: Biometrische Merkmale sind keine Geheimnisse. Viele von ihnen können nicht einmal verborgen werden, sondern werden zwangsläufig offengelegt oder hinterlassen ([ebd.]) – Beispiele hierfür sind Fingerabdrücke, Gesichtsmerkmale oder personenspezifische Eigenschaften der Stimme.

indexBiometrie!Schriftwerkennung Hinsichtlich der Zuverlässigkeit biometrischer Verfahren finden sich in der aktuellen Literatur kaum verlässliche oder überhaupt nur aussagekräftige Angaben, Behrens et al. [38, S. 111] schreibt beispielsweise zur Gesichtserkennung: „Über die Zuverlässigkeit im Sinne geringer Fehlerraten lassen sich leider keine allgemeingültigen Aussagen treffen. Es ist derzeit praktisch unmöglich, hier verlässliche Werte zu finden“.

In informationstechnologischen Systemen werden die jeweils verwendeten biometrischen Merkmale zu Datensätzen, die möglicherweise gestohlen und mißbräuchlich verwendet werden können, so Behrens et al. in [38, S. 3]. Biometrische Verfahren sind entsprechend RFC 1704 [58, S. 3 f.] ebenfalls anfällig gegenüber passiven Angriffen und Wiederholungsangriffen.

Schneier weist in [99, S. 189] darauf hin, daß die biometrische Merkmalerkennung häufig zur automatisierten Identifizierung von Personen mißbraucht wird: Während bei der Authentifizierung das verwendete informationstechnologische System lediglich die Frage beantworten muß, ob ein biometrischer Datensatz ausreichend Ähnlichkeit¹³ mit dem zu einer bestimmten Person gespeicherten Datensatz hat, muß es bei der Identifizierung überprüfen, ob der biometrischer Datensatz ein gewisses Maß an Ähnlichkeit mit irgendeinem Datensatz aus einer möglicherweise sehr großen Datenbank aufweist.

¹³Die Datensätze werden einander niemals gleichen, weil die verwendeten Digitalisierungsverfahren verhältnismäßig ungenau arbeiten und sich biometrische Merkmale verändern, man denke nur an Bartwuchs oder Makeup bei der Gesichtserkennung (vgl. Behrens et al. [38, S. 14-18]).

2.3.4 Kryptographische Authentifizierungsverfahren

Um das Ausspähen von Authentifizierungsdaten zu erschweren, werden diese bei modernen Systemen verschlüsselt, also durch kryptographische Verfahren^(Glossar) geschützt abgelegt (vgl. Fuhrberg, [56, S. 441 f.] und die Single UNIX Specification, [24]). Vorzugsweise werden dafür Einweg-Funktionen^(Glossar) genutzt, die eine Wiederherstellung der eigentlichen Authentifizierungsdaten unmöglich machen (vgl. Schneier, [98, S. 148]). Jedoch ist diese Methode kein Allheilmittel gegen das Ausspähen der Daten, in Abhängigkeit von der verwendeten Einweg-Funktion sind vielfach Angriffe zur Erzeugung von Nachrichten, die den gleichen Rückgabewert wie das tatsächliche Passwort geben, bekannt (vgl. Schneier, [96, Kapitel 18]). Generelle Methoden zum Brechen von Einweg-Funktion beschreibt Schneier in [96, S. 165 f.].

Auch können kryptographische Verfahren die Vertraulichkeit der über eine Kommunikationsverbindung übertragenen Daten sichern. Für Authentifizierungszwecke ist dies insbesondere dann sinnvoll, wenn statische Authentifizierungsdaten wie Passwörter und biometrische Merkmale übertragen werden müssen: Das Verfahren kann durch die Sicherung der Übertragenen Daten gegen Wiederholungsangriffe gesichert werden (vgl. RFC 2246, [50]). Hierbei werden die Daten vom Sender verschlüsselt^(Glossar), im Geheimtext^(Glossar) übertragen und vom Empfänger entschlüsselt^(Glossar). Problematisch bei diesem Vorgehen ist immer der Austausch des zu verwendenden Schlüsselmaterials^(Glossar), da dieses unverschlüsselt übertragen werden muß (vgl. Schneier, [96, S. 26]).

Verschlüsselungsverfahren

Grundsätzlich unterscheidet man zwei Gruppen von Verschlüsselungsverfahren danach, wie in dem jeweiligen Verfahren mit dem Schlüsselmaterial umgegangen wird.

Symmetrische Verfahren zeichnen sich dadurch aus, daß für die Ver- und Entschlüsselung Schlüssel verwendet werden, die auseinander ableitbar sind. Bei vielen Verfahren sind die zum Ver- und Entschlüsseln verwendeten Schlüssel auch gleich (vgl. Schneier, [96, S. 3]). Mathematisch läßt sich dieser Sachverhalt unter Verwendung der von Burrows et al. in [42] vorgeschlagenen Syntax folgendermaßen darstellen:

$$C = \{M\}_K; M = \{C\}_K$$

Dabei ist M eine Nachricht (engl. *Message*) im Klartext und C das Chiffre dieser Nachricht. Der verwendete Schlüssel wird mit K bezeichnet. eine Operation der Art $\{M\}_K$ stellt die Verschlüsselung von M mit K als Schlüssel dar.

Asymmetrische Verfahren (engl. *Public-Key Cryptography*) verwenden im Gegensatz dazu jeweils ein Schlüsselpaar pro Kommunikationspartner. Diese sind derart miteinander verknüpft, daß eine Nachricht, die mit dem einen Schlüssel verschlüsselt wurde, nur mit dem anderen entschlüsselt werden kann. Die dem zugrundeliegenden Prinzipien wurden von Diffie et al. in [51] beschrieben, auch Schneier erläutert das Verfahren in [96, S. 31-44, 461-502]. Die zwei Teile des Schlüsselpaares werden unterschiedlich behandelt. Ein Teil wird vom Benutzer geheimgehalten, dies ist der sogenannte „geheime Schlüssel“ (auch „privater Schlüssel“ oder „Secret Key“). Der andere Teil wird derart hinterlegt, daß Kommunikationspartner auf ihn zugreifen können. Dieser wird als „öffentlicher Schlüssel“ oder „Public Key“ bezeichnet. Nach Burrows et al. ([42]) läßt sich die Verwendung asymmetrischer Kryptographie wie folgt darstellen:

$$M = \{\{M\}_{K_A}\}_{K_A^{-1}}; M = \{\{M\}_{K_A^{-1}}\}_{K_A}$$

Hierbei bezeichnet K_A den öffentlichen Schlüssel eines Teilnehmers A und K_A^{-1} analog seinen privaten Schlüssel.

Symmetrische Verfahren haben in der Anwendung insbesondere den Vorteil, daß sie sich durch eine einfache und kompakte, besser überschaubare

Gesamtstruktur auszeichnen, wodurch Probleme oder Schwachstellen möglicherweise einfacher gefunden werden könnten. Darüber hinaus erreichen sie mit verhältnismäßig kurzen Schlüsseln eine vergleichbare Sicherheit wie asymmetrische Verfahren mit signifikant längeren Schlüsseln (vgl. Schneier, [96, S. 166 f.]). Resultierend daraus sind Benutzer in der Regel durchaus in der Lage, sich symmetrische Schlüssel zu merken. Das primäre Problem der symmetrischen Kryptographie liegt in der hohen Anzahl der benötigten Schlüssel – immerhin ein Schlüssel pro Kommunikationsverbindung zwischen zwei Teilnehmern – und deren Verbreitung (vgl. Schneier, [96, S. 176 f.]).

Beide Probleme können durch den Einsatz asymmetrischer Verfahren gelöst werden: Jeder Benutzer muß sich lediglich ein Schlüsselpaar erzeugen, den öffentlichen Schlüssel für seine Kommunikationspartner zugänglich aufbewahren und den privaten Schlüssel vor allen verbergen. Damit kann ihm jeder eine mit dem öffentlichen Schlüssel verschlüsselte Nachricht schicken, die nur er nach der Entschlüsselung unter Verwendung des privaten Schlüssels lesen kann. Darüber hinaus kann das Verfahren auch die Authentizität und Integrität einer Nachricht gewährleisten: Der Benutzer *A* muß die Nachricht lediglich mit seinem privaten Schlüssel verschlüsseln, die Nachricht kann daraufhin von jedem gelesen werden. Die Rezipienten werden darüber hinaus darauf vertrauen, daß die Nachricht wirklich von *A* stammt, da nur er seinen privaten Schlüssel kennt (vgl. Schneier, [96, S. 4 f., 34-46]). Ganz offensichtlich steht und fällt die Sicherheit des Systems mit der Geheimhaltung des privaten Schlüssels. Das Problem dabei ist jedoch, daß asymmetrisches Schlüsselmaterial in der Regel aus einigen hundert Bytes arbiträrer Daten besteht, die sich ein Mensch kaum merken kann. Aus diesem Grund wird das Schlüsselmaterial häufig auf Datenträgern gespeichert und durch symmetrische Kryptographie mit „merkbaaren“ Schlüsseln geschützt (vgl. Schneier, [96, S. 181]).

Einsatz von Verschlüsselungsverfahren zur Authentifizierung

Die oben vorgestellten Verfahren lassen sich auch zur Authentifizierung eines Benutzers bei einem Authentifizierungsserver einsetzen. Verschiedene kryptographische Protokolle^(Glossar) werden von Schneier in [96, S. 53-65] beschrieben.

Ein zwischen zwei Parteien einsetzbares und auf symmetrischer Kryptographie basierendes Protokoll könnte beispielsweise wie im Folgenden¹⁴ dargestellt, ablaufen. Hierbei versucht ein Benutzer (Prover, P) sich bei einem Authentifizierungsserver (Verifier, V) zu authentifizieren. Beide haben vorher untereinander einen geheimen Schlüssel¹⁵ K , vergleichbar mit einem Passwort, verabredet.

- (1): V erzeugt eine P unbekannte Nachricht M und sendet sie an P .
- (2): P berechnet $C = \{M\}_K$ und sendet es an V .
- (3): V kann nun $M' = \{C\}_K$ entschlüsseln und mit dem von ihm gesendeten M vergleichen. Gilt $M = M'$ weiss er, daß P den Schlüssel K kennt.

Das Verfahren ähnelt in einigen Eigenschaften sehr stark den weiter oben beschriebenen textuellen Authentifizierungsverfahren: Es müssen vorher Schlüssel ausgetauscht werden. Es hat auch eine ähnliche Schwachstelle wie diese Verfahren, nämlich die Notwendigkeit zur Speicherung von Klartext-Passworten auf dem Authentifizierungsserver. Der entscheidende Vorteil des Verfahrens besteht darin, daß P dem V sein Passwort bei der Authentifizierung nicht verraten muß, was den Erfolg von Angriffen, bei denen sich ein

¹⁴Das Protokoll wird vom Autor als ein selbst erdachtes Beispiel angeführt, eine Quelle für dieses simple Vorgehen konnten nicht gefunden werden.

¹⁵Es handelt sich hierbei um einen symmetrischen Schlüssel.

Angreifer als V auszugeben versucht, stark erschwert. Auch ist es von enormer Wichtigkeit, daß M dem P vorher wirklich unbekannt ist, das Verfahren wäre sonst anfällig für Wiederholungsangriffe.

Weitere Beispiele für auf symmetrischer Kryptographie basierende Authentifizierungsverfahren sind beispielsweise das von Needham et al. in [81] publizierte und im Kerberos-Authentifizierungsframework (vgl. RFC 1510, [71]) in einer leicht modifizierten Form verwendete Protokoll¹⁶.

Mit der Entwicklung der asymmetrischen Kryptographie wurden Verfahren verfügbar, durch deren Einsatz sich die Speicherung und Verwaltung symmetrischer Schlüssel erübrigt, wodurch die oben angeführten Probleme reduziert werden. Dementsprechend existieren inzwischen verschiedene, auf asymmetrischen Verfahren basierende Authentifizierungsprotokolle (vgl. Schneier, [96, S. 61-64]). Auf diese wird in Kapitel 4, Abschnitt 4.2.2 noch näher eingegangen.

Bewertung kryptographischer Authentifizierungsverfahren

Die formale Analyse kryptographischer Authentifizierungsprotokolle gestaltet sich im Allgemeinen als sehr schwierig. Entsprechend Schneier ([96, S. 65 f.]) gibt es hierzu verschiedene Verfahren, das am häufigsten verwendete ist die BAN-Logik, beschrieben in Burrows et al. ([42]). Ein Teil der von Burrows et al. vorgeschlagenen Syntax zur Darstellung kryptographischer Verfahren wurde bereits weiter oben verwendet. Eine erschöpfende Darstellung der auch in Kapitel 4 dieser Arbeit verwendeten Logik sei der interessierte Leser auf Burrows et al., [42] und Abadi et al., [31] verwiesen. Kurzdarstellungen finden sich in Schneier, [96, S. 65-68] und Eckert, [52, 436-444].

¹⁶Needham et al. beschreibt in [81] auch ein asymmetrisches Verfahren. Auf dieses wird in Abschnitt 4.2.2 kurz eingegangen.

2.4 Bestehende Lösungen und deren Probleme

In der Praxis vernetzter Arbeitsumgebungen werden bereits seit einigen Jahren verteilte Autorisierungsdienste und Identity-Management-Systeme entwickelt bzw. eingesetzt. Dieser Abschnitt gibt einen Überblick über zwei vom Autor aufgrund der subjektiven Einschätzung ihrer Verbreitung ausgewählte Systeme und geht auf deren Vorzüge und Probleme ein.

2.4.1 Microsoft Passport

Passport ist Bestandteil von Microsofts .NET-Strategie und wird seit 1999 eingesetzt (vgl. Microsoft, [27]). Nach eigenen Angaben hat Microsoft derzeit 200 Millionen Benutzerkonten und führt monatlich 3.5 Milliarden Authentifizierungen durch ([ebd.]). Das Ziel des Systems ist es, Internet-Benutzern eine Online-Identität zu verschaffen, mit der sich diese dann bei Internet-Applikationen ausweisen können (vgl. Eckert, [52, S. 423]).

Realisierung von Single-Sign-On Das System bietet seinen Benutzern über einen Web-Browser bedienbare Single Sign-On Funktionalität an. Hat der Benutzer sich einmal authentifiziert, kann er auf weitere, *Passport* verwendende Web-Applikationen zugreifen, ohne sich bei diesen erneut anmelden zu müssen. Realisiert wird dies über die Verwendung von Cookies¹⁷ und HTTP-Redirects¹⁸ (vgl. Microsoft, [21, S. 7]).

¹⁷Cookies (dt. Kekse) enthalten Daten, die von einer Web-Applikation gesendet und vom Web-Browser auf dem Arbeitsplatz-System des Benutzers angelegt wurden. Cookies können von der Applikation auch wieder ausgelesen werden und erlauben es damit, beim eigentlich anonymen Zugriff auf einen Web-Server verschiedene Aktionen einem Benutzer zuzuordnen (vgl. Claus et al., [45, S. 147 f.].)

¹⁸Das HTTP-Protokoll spezifiziert entsprechend RFC 2616 ([54]) Möglichkeiten zur automatischen Weiterleitung von Anfragen an eine andere Zieladresse.

Identifizierung und Authentifizierung Der zur Identifizierung und Authentifizierung von Benutzern verwendete Datenbestand muß sich auf von Microsoft betriebenen *Passport*-Servern befinden (vgl. Eckert, [52, S. 422 f.]). Laut Microsoft ([21, S. 8-16]) unterstützt *Passport* drei verschiedene Anmeldeverfahren: Beim *Standard Sign-In* wird der Benutzer über seine E-Mail-Adresse identifiziert, die Authentifizierung erfolgt über ein Passwort. Sowohl die E-Mail-Adresse als auch das Passwort werden zur Übertragung mittels SSL (vgl. Netscape Inc., [55]) verschlüsselt. Eine weitere Methode ist das *Secure Channel Sign-In*. Der Unterschied zum Standard Sign-In besteht hierbei darin, daß alle zwischen dem Benutzer, der Applikation und dem Authentifizierungsserver übertragenen Daten mittels SSL verschlüsselt werden. Die dritte Anmeldeverfahren ist das *Strong Credential Sign-In*, die gesamte Datenübertragung erfolgt, ebenso wie beim Secure Channel Sign-In, verschlüsselt, der Benutzer muß neben dem Passwort noch einen *Security Key* angeben. Dieser Security Key ist eine vierstellige Ziffer, er muß innerhalb von fünf Versuchen korrekt eingegeben werden und wird andernfalls gesperrt (vgl. Microsoft, [21, S. 15]). *Passport* unterstützt also lediglich textuelle Authentifizierungsverfahren.

Probleme beim Einsatz von Passport Eckert führt in [52, S. 429] den Einsatz von Cookies als eines der primären Probleme von *Passport* an. Cookies dürfen üblicherweise nur von dem HTTP-Server gelesen werden, der sie geschrieben hat (vgl. Fuhrberg, [56, S. 6 f.]). Dieses Verhalten des Web-Browsers verhindert das Ausspähen möglicherweise personenbezogener, in den Cookies gespeicherter Daten durch dazu nicht berechnete Web-Applikationen. Beim Single Sign-On über *Passport* muß der Sicherheitsmechanismus jedoch ausgeschaltet werden, damit die vom *Passport*-Server geschriebenen Cookies von den zu verwendenden Web-Applikationen gelesen werden können (vgl. Eckert, [52, S. 429]). Insbesondere bei der Verwendung des Standard Sign-In bietet sich hierbei, so Eckert ([ebd.]) sehr einfach die Möglichkeit zur Vortäuschung einer falschen Identität. Weite Angriffsmög-

lichkeiten stellen beispielsweise das Maskieren des *Passport*-Servers¹⁹ oder das Einschleusen von Proxies, auf denen die Kommunikationsverbindung zwischen Server und Benutzer abgehört werden kann, dar. Eckert beschreibt in [52, S. 428-435] diese und verschiedene weitere Angriffe ausführlich. Auch die Tatsache, daß *Passport* ein System ist, dessen Spezifikationen und Implementierungstechnische Details nicht öffentlich zugänglich sind, also auch nicht, wie in Abschnitt 2.2.1 dargestellt, von unabhängigen Experten evaluiert werden können, wird das Vertrauen der Benutzer in *Passport* nicht erhöhen. Eckert führt in [52, S. 422] die hohen Benutzerzahlen darauf zurück, daß jeder *Hotmail*²⁰-Benutzer automatisch ein *Passport*-Konto erhält.

Insbesondere die Tatsache, dass personenbezogene Daten zur Identifizierung und Authentifizierung der Benutzer auf einem zentralen Server gespeichert werden müssen, der dadurch wiederum zu einem sehr attraktiven Angriffsziel wird (vgl. Eckert, [52, S. 433]), führte zur Konzeption des *Liberty Alliance Projects*.

2.4.2 Das Liberty Alliance Project

Als direkte Konkurrenz zu *Microsoft Passport* begannen 2001 insgesamt 33 Firmen mit der Begründung des *Liberty Alliance Projects*. Ziel der Bestrebungen, inzwischen getragen von über 150 kooperierenden Unternehmen ist es, einen industrieweit einsetzbaren Single Sign-On Standard zu entwickeln. Die Grundgedanken des derzeit erst partiell einsetzbaren Systems sind von der *Liberty Alliance* in [26] aufgeführt. Danach funktioniert das Single Sign-On nach einem ähnlichen Konzept wie bei *Passport*: Der Benutzer muß sich zuerst bei einem *Identity-Provider* identifizieren und mittels eines textuellen Verfahrens authentifizieren, danach kann er auf von einem *Service Provider* angebotene Dienste in Anspruch nehmen. Die Weitergabe eines Autori-

¹⁹Der Computer eines Angreifers gibt sich hierbei als ein *Passport*-Server aus und bekommt von sich anmeldenden Benutzern Zugangsdaten, die anschließend weiterverwendet werden können (vgl. Eckert, [52, S. 429]).

²⁰*Hotmail* ist ein von Microsoft betriebener E-Mail-Dienst. *Hotmail*-Benutzerkonten werden mittels *Microsoft Passport* verwaltet.

sierungszeichens erfolgt über HTTP-Weiterleitungen, Cookies oder Formularfelder²¹ in Webseiten.

Datenspeicherung Fundamental anders als bei *Passport* ist die Art der Speicherung von Identifizierungs- und Authentifizierungsinformationen (vgl. *Liberty Alliance*, [26]). Diese werden nicht an einer zentralen Stelle, sondern dezentral gespeichert. Das Konzept des *Liberty Alliance Project* sieht dafür eine große Anzahl von Identity-Providern, vertrauenswürdigen Organisationen, die ähnlich wie Trust Center für die Identität von Benutzern bürgen, vor. Diese dezentrale Speicherung reduziert die mit *Passport* verbundenen Risiken des Diebstahls großer Mengen personenbezogener Daten durch den Einbruch in ein einzelnes Computersystem beträchtlich. Auch haben Benutzer die freie Wahl zwischen verschiedenen Identity-Providern und sind nicht zwangsläufig an die Weitergabe ihrer Daten an einen bestimmten Konzern gebunden. Die Rolle des Identity-Providers kann entsprechend der Spezifikation eine Institution übernehmen, die ohnehin über personenbezogene Daten des Benutzers verfügt – beispielsweise Banken oder gar die Einwohnermeldebehörden. Auch unterstützt das System die Verwaltung verschiedener *Identity-Profiles* eines Benutzers, um unterschiedliche Situationen zu repräsentieren. In [26] wird hierfür häufig das Beispiel des *Home Profile* und des *Work Profile* verwendet. In dem einen Fall wird der Benutzer vorrangig Zugriff auf private E-Mail benötigen oder Einkäufe erledigen, während er im anderen Fall im Auftrag seiner Firma handelt.

Offener Standard Ebenfalls für Das *Liberty Alliance Project* spricht seine offene Spezifikation. Die an der Erarbeitung des Konzeptes beteiligten Firmen publizierten Arbeitsplanung und Entwürfe von Anfang an im Internet unter <http://www.projectliberty.org>, wodurch unabhängige Experten bereits sehr früh Einblick in die Spezifikation erhielten.

Probleme beim Einsatz von Liberty Durch die Verwendung des HTTP-Protokolls zum Austausch von Identifizierungs-, Authentifizierungs-

²¹Die Verwendung von Formularfeldern ist ebenfalls in RFC 2616 ([54]) spezifiziert.

und Autorisierungsinformationen sind beim Einsatz des *Liberty Alliance Projects* ähnliche Schwachstellen wie bei *Passport* zu erwarten. Auch sind die bereits im Abschnitt 2.2.3 aufgeführten grundsätzlichen Bedenken bezüglich Single Sign-On Technologien nicht außer Acht zu lassen. Da dem Autor keine umfassende Analyse des Systems vorlag und diese auch nicht Gegenstand der vorliegenden Arbeit ist, wird an dieser Stelle auf weitere Ausführungen verzichtet.

Kapitel 3

Das

Identity-Management-System

CIDAS

OMNIS ENIM RES, QUAE DANDO NON DEFICIT, DUM HABETUR
ET NON DATUR, NONDUM HABETUR, QUOMODO HABENDA EST.¹

— St. Augustinus, DE DOCTRINA CHRISTIANA ([33])

¹Sinngemäße Übersetzung: „Jede Sache, die dadurch, daß man sie weitergibt nicht verloren geht, wird nicht auf richtige Weise besessen, wenn man sie nur besitzt, aber nicht weitergibt“.

Das Identity-Management-System CIDAS

An der Fachhochschule Brandenburg wird der „Configurable Internet Directory and Authentication Service“ CIDAS als ein modernes Identity-Management-System entwickelt, das für viele gängige Betriebssysteme und eine Vielzahl von Internet-Anwendungen unterschiedliche und differenziert nutzbare Authentifizierungsdienstleistungen erbringt (vgl. Holl et al., [61]). Damit ist es möglich, Autorisierung und Benutzerverwaltung von den eigentlichen Aufgaben der Anwendungen abzukoppeln. Die Grundlagen hierfür wurden von Schäfer in einer Diplomarbeit mit dem Thema „Konzeption und Prototypische Umsetzung eines Benutzermanagement-Systems mit Transaktionskomponente“ [93] dokumentiert. Dieses Kapitel bezieht sich vorrangig auf diese Arbeit, sowie auf das von Holl et al. in [61] publizierte Konzept von CIDAS. Insbesondere die letzte Arbeit wird häufig wörtlich zitiert. Da der Autor an Holl et al. [61] selbst mitgearbeitet hat, wird an dieser Stelle auf explizite Hinweise auf zitierte Stellen verzichtet.

Da CIDAS als ein freies Open-Source-Projekt entsprechend der *GNU General Public License* ([8]) entwickelt und veröffentlicht werden soll, wird es für jeden und weltweit² einsetzbar sein (vgl. Schäfer, [93, S. 73 f.]). Firmen, aber auch Institutionen wie Hochschulen und Verwaltungseinrichtungen sollen durch den Einsatz von CIDAS in die Lage versetzt werden, dem zu verwaltenden Personenbestand, also Kunden, Studenten oder Einwohnern, Möglichkeiten zur Selbstverwaltung bzw. zur selbstständigen Pflege ihrer Daten zu geben und gleichzeitig den heute geltenden hohen Ansprüchen an den Schutz personenbezogener Daten und der eingesetzten Systeme gerecht zu werden. Auch sollen Entwickler und Anwender die Möglichkeit erhalten, alle Bereiche von CIDAS im Quellcode einzusehen, zu erweitern oder ihren eigenen Bedürfnissen anzupassen. Darüber hinaus wird auf diese Weise auch die Überprüfung einer CIDAS-Implementierung durch Dritte hinsichtlich ihrer Korrektheit ermöglicht. Siehe hierzu auch Abschnitt 2.2.1. Dem Autor

²In Bezug auf die verwendeten kryptographischen Verfahren zur Benutzerauthentifizierung wird der Einsatz von CIDAS u. U. durch die jeweils geltenden rechtlichen Bestimmungen eingeschränkt.

sind keine weiteren derartig frei verfügbaren Identity-Management-Systeme bekannt.

Für den verwalteten Personenbestand soll die Situation überschaubarer gestaltet werden: Benutzer sollen, natürlich immer in Abhängigkeit von der Art der Anwendung, jederzeit Auskunft bezüglich der über sie gespeicherten Informationen erhalten können und werden auch die Möglichkeit haben, die Nutzung von Informationen zu verweigern.

3.1 Das Komponentenmodell von CIDAS

Zur Realisierung der Authentifizierung ist CIDAS als ein Server-basiertes System konzipiert: Es gibt einen Authentifizierungsserver und zugehörige Clients. Implementierungen der CIDAS-Client-Software sind für gängige Betriebssysteme wie moderne Unixe, MacOS X und Windows geplant. In Abhängigkeit von der von diesen Systemen unterstützten Hardware gibt es unter Umständen Einschränkungen hinsichtlich der Funktionalität des jeweiligen Clients.

Das Identity-Management-System setzt sich aus verschiedenen Modulen zusammen. Diese wurden größtenteils bereits von Schäfer in [93, S. 86-91] beschrieben. In Abbildung 3.1 auf Seite 37 wird das Gesamtsystem graphisch dargestellt. Der große gestrichelte Kasten stellt hierbei den Prozeßraum des CIDAS-Servers, also jene Komponenten, die sich innerhalb der Server-Software befinden, dar. Die Verbindungen zwischen diesen Komponenten repräsentieren dabei Kommunikationswege. Das System verfügt also über eine Reihe Server-interner und -externer Kommunikationsschnittstellen.

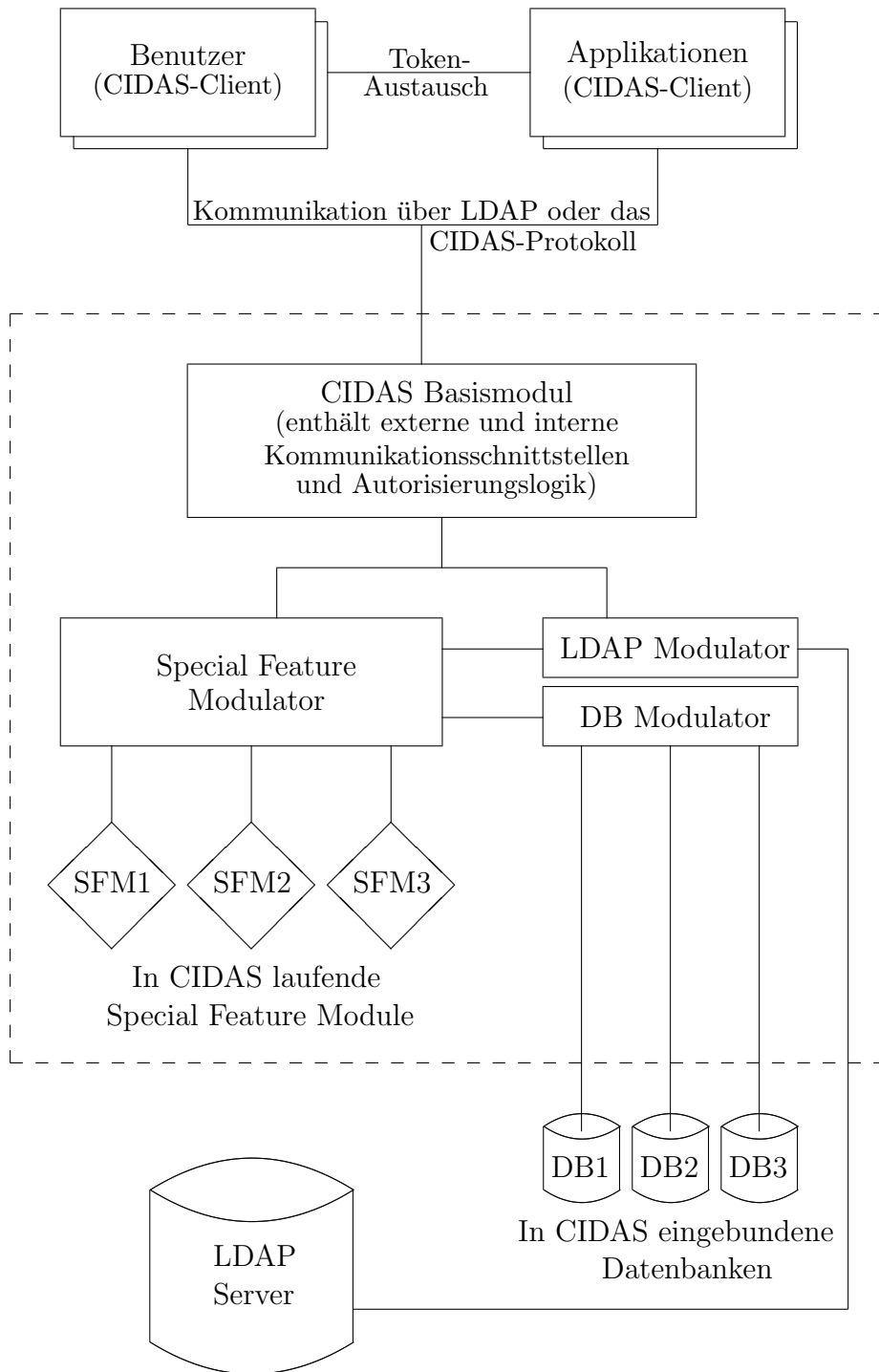


Abbildung 3.1: Modularisierung von CIDAS

Die Kommunikation zwischen Server und Clients³ kann über unsichere Kommunikationswege ablaufen, die übertragenen Daten werden durch kryptographische Verfahren geschützt. Das zur Kommunikation verwendete Protokoll wird in Kapitel 5 dieser Arbeit spezifiziert.

3.1.1 Unterstützte Anwendungen und

CIDAS unterstützt auf Anwendungsseite jede Anwendung, die entweder LDAP⁴ oder das CIDAS-eigene Protokoll verwenden bzw. einen CIDAS-Client bedienen kann. Dies schließt die Anmeldung an einem Arbeitsplatz-Rechner ebenso wie das Login bei parameterorientierten Anwendungen wie etwa Internet-Shopsystemen ein. Auch erlaubt die Verwendung von LDAP die verhältnismäßig unkomplizierte Anbindung vieler bereits existierender und auf LDAP aufbauender Applikationen.

Neben LDAP und dem CIDAS-Protokoll sieht Schäfer in [93, S. 87 f.] eine HTTP-basierte Schnittstelle für den CIDAS-Server vor. Diese ist bislang unspezifiziert, könnte jedoch Liberty-konforme (siehe [26]) Zugriffsmöglichkeiten zu CIDAS bieten.

3.1.2 Single-Sign-On

Bei Verwendung von CIDAS als Authentifizierungssystem meldet sich ein Benutzer im Gegensatz zu herkömmlichen Systemen nicht mehr bei jeder Anwendung einzeln an, vielmehr muß er sich lediglich einmal bei einem CIDAS-Server authentifizieren. Nach der Anmeldung erhält der Benutzer vom Server ein Autorisierungszeichen, im Folgenden als *Token* bezeichnet. Anwendungen, die ebenfalls bei diesem Server authentifiziert sind, können ein ihnen von einem Benutzer übergebenes Token zum Server zurücksenden und darüber die Authentizität des Benutzers zu prüfen. Die Anwendung bekommt

³In Abbildung 3.1 werden die Clients durch die Benutzer und Applikationen symbolisiert.

⁴LDAP: „Lightweight Directory Access Protocol“, siehe RFC 1777 [113].

hiernach auch automatisch eine Benachrichtigung, wenn sich der Benutzer beim CIDAS-Server abmeldet.

Auch die Bereitstellung eines universell einsetzbaren Authentifizierungs-Frameworks, das neben textuellen Authentifizierungsmethoden auch hochwertige kryptographische und biometrische Verfahren, gekoppelt mit Single-Sign-On Funktionalität, anbietet, stellt eine zukunftssträchtige Innovation im IT-Sektor dar.

3.1.3 Autorisierungsmethoden in CIDAS

Entsprechend Schäfer [93, S. 8-10, 66-68] erfolgt die Benutzerlegitimierung im Identity-Management-System CIDAS nach dem in Abschnitt 2.3 dargestellten Schema: In der ersten Stufe identifiziert sich ein Benutzer bei einem CIDAS-Server, indem er durch Angabe einer bestimmten Menge persönlicher Daten seine angebliche Identität eindeutig bestimmt. Durch die Abfrage weiterer Daten versucht der CIDAS-Server in der zweiten Stufe, die Korrektheit der angegebenen Identifizierungsdaten zu überprüfen.

CIDAS bietet neben der Unterstützung von textuellen Authentifizierungsmethoden auch die Authentifizierung mittels asymmetrischer Verschlüsselung und die Verwendung biometrischer Merkmale an. Bei einer auf asymmetrischer Verschlüsselung basierenden Authentifizierung kann das Schlüsselmaterial auf verschiedenste Art und Weise bereitgestellt werden, sowohl als Datei⁵ als auch auf einem identifizierbaren Datenträger. Das hierfür verwendete Authentifizierungsverfahren ist in Kapitel 4 vorgestellt. Zusätzlich wird es natürlich auch möglich sein, bereits etablierte Authentifizierungsgeräte wie Smart-Cards heranzuziehen.

Neben diesen generischen Authentifizierungsmethoden ist es bei Nutzung von CIDAS auch möglich, zur Authentifizierung eines Benutzers kombinierte

⁵Denkbar sind beispielsweise X.509-Zertifikat entsprechend [44] oder OpenPGP-Schlüssel nach RFC 2440 [43].

Verfahren einzusetzen. So bietet sich beispielsweise im Bereich der sicherheitskritischen Anwendungen die Nutzung einer Kombination aus auf asymmetrischer Kryptographie und biometrischen Authentifizierungsverfahren basierenden Authentifizierung an.

Die Festlegung, welche Kombinationen von Authentifizierungsverfahren zum Einsatz kommen, kann auf verschiedene Weise erfolgen. Zum einen kann der Betreiber des CIDAS-Servers entsprechende Mindestanforderungen bezüglich der für den Zugriff auf einzelne Datenbestände oder Applikationen notwendigen Sicherheit bestimmen.

Zum anderen ist es auch Benutzern möglich, im Rahmen dieser Festlegung ihr eigenes Sicherheitsbedürfnis durch die Auswahl gleich- oder höherwertiger Authentifizierungsverfahren umzusetzen. Unterschreitet ein Benutzer die Sicherheitsanforderungen einzelner Applikationen, kann er diese entweder nicht nutzen oder muß sich mit einem höherwertigeren Verfahren erneut authentifizieren.

Betreiber legen immer nur den für sie unbedingt notwendigen minimalen Schutzbedarf fest, so daß von den Benutzern jederzeit höherwertigere Verfahren eingesetzt werden können. Ist der notwendige Schutz ein sehr hoher, erübrigt sich natürlich diese Option.

3.1.4 Speicherung personenbezogener Daten

Viele Anwendungen und auch der Identifikations- und Authentifizierungsdienst selbst kommen nicht ohne Zugriffe auf personenbezogenen Daten aus. Deren Speicherung und vor allem der oftmals aus Sicht des Benutzers nicht nachvollziehbare Umgang mit diesen Daten ist einer der primären Kritikpunkte an bestehenden Anwendungen. Mit CIDAS wird versucht, eine aus Sicht des Benutzers maximal transparente Datenerfassung und sichere Datenspeicherung zu realisieren. Das bedeutet zum einen, daß in Abhängigkeit von der Applikation bzw. der grundsätzlichen Vereinbarung über die CIDAS-

Speicherung nur die unbedingt notwendigen Daten erfasst werden. Zum anderen kann der Benutzer jederzeit Auskunft über die gespeicherten Daten erhalten und hat zudem die direkte Kontrolle über diese Daten. Somit kann ein größtmöglicher Schutz gegen die eventuelle Kompromittierung der Daten realisiert werden.

Bezogen auf die zur Identifikation eines Benutzers bzw. zur Abwicklung von Transaktionen mit einem Benutzer benötigten Daten werden vorrangig dessen E-Mail-Adresse, CIDAS-Benutzername, Namen, die Postanschrift, unter Umständen sein Alter sowie seine Bankverbindung verwendet. Die Speicherung dieser Daten kann laut Schäfer [93, S. 70] im Klartext oder verschlüsselt erfolgen. Je nach Applikation kann sogar akzeptiert werden, daß, sollte ein Benutzer besonders viel Wert auf dem Schutz seiner persönlichen Daten legen, an Stelle der eigentliche Benutzerdaten lediglich Daten gespeichert werden, mit denen sich die Korrektheit eingegebener Benutzerdaten nachweisen läßt, aus denen jedoch selbst keine Rückschlüsse auf den Benutzer gezogen werden können. In diesem Falle bliebe die sichere Speicherung der eigentlichen personenbezogenen Daten dem Benutzer selbst überlassen⁶.

Die zur Identifikation und Authentifizierung eines Benutzers benötigten Daten werden in einem LDAP-Server gespeichert. Der Zugriff auf diesen erfolgt über den in Abbildung 3.1 dargestellten *LDAP-Modulator*. Weitere personenbezogene Daten werden von CIDAS in explizit dafür vorgesehenen, besonders geschützten Datenbanken gespeichert. Auch hier können kryptografische Verfahren als Schutzmechanismen eingesetzt werden. Insbesondere durch das Verwenden von nur innerhalb von CIDAS bekannten Datensatzidentifikatoren wird ein Ausspähen, vor allem aber die Zuordnung der Daten zu den zugehörigen Benutzern durch dazu nicht Berechtigte zusätzlich erschwert. Die Anonymisierung der Daten wird von der Komponente *DB Modulator* realisiert.

⁶Auf die Speicherung der E-Mail-Adresse oder des CIDAS-Benutzernamens kann nicht verzichtet werden, da zumindest ein eindeutiges Datum zur Identifizierung des Benutzers nötig ist.

3.1.5 Special-Feature-Module

Eine wichtige Eigenschaft von CIDAS ist die einfache Erweiterbarkeit des Servers. Durch so genannte Special-Feature-Module läßt sich sicherheitskritische Anwendungslogik in den Authentifizierungsserver auslagern. Die von Special-Feature-Modulen gebotene Funktionalität ist entweder mittels eines konventionellen Internet Browsers oder über den CIDAS-Client erreichbar. Auch sind aus einem Special-Feature-Modul heraus Zugriffe auf speziell geschützte Datenbanken möglich. Special-Feature-Module können derzeit lediglich in Java⁷ entwickelt werden, Schnittstellen zu anderen Sprachen sind denkbar. Die in den Server integrierte Logik zur Modifikation der persönlichen Daten eines authentifizierten Benutzers durch eben diesen wird bereits in Form eines Special-Feature-Moduls implementiert werden.

3.2 Betrachtung der Sicherheit von CIDAS

Wie bei anderen Single-Sign-On (siehe Abschnitt 2.2.3) Systemen ergibt sich natürlich auch beim Einsatz von CIDAS das Problem, daß die Kompromittierung des Single-Sign-On-Systems möglicherweise sehr weitreichende Folgen hat und dem Angreifer Zugang zu sehr vielen Daten eines Benutzers bietet. Wie Schneier in [98, S. 255-269] in [76] ausführt, existiert dieses Problem auch in Arbeitsumgebungen, in denen keine Single-Sign-On-Verfahren eingesetzt werden, weil Benutzer viel zu häufig dazu tendieren, ähnliche oder gar identische Passworte für verschiedene Dienste zu verwenden.

Im Gegensatz zu den in Abschnitt 2.4 beschriebenen Systemen hat CIDAS jedoch den Vorteil, neben textuellen Authentifizierungsverfahren auch kryptographische- und hardwarebasierte⁸ Verfahren anzubieten. Diese sind entsprechend RFC 1704 ([58]) resistent gegen passive, aktive und auf Wiederholungen basierende Angriffe.

⁷Java ist eine von Sun Microsystems Inc. entwickelte Programmiersprache. Informationen hierzu sind unter <http://java.sun.com/> erhältlich.

⁸Es ist die Unterstützung von Chipkarten und vergleichbaren Authentifizierungsgeräten gemeint.

Auch wird der Austausch der Autorisierungszeichen nicht über von jedem auslesbare Cookies oder Ähnliches durchgeführt, vielmehr ist selbst das Autorisierungszeichen für einen Angreifer noch völlig wertlos – wie in Kapitel 5 erläutert, bedarf dessen Einsatz nochmals einer Zustimmung seitens des jeweiligen Benutzers.

In Abschnitt 2.2.1 stellte der Autor bereits die Notwendigkeit der Offenlegung von Spezifikationen und Quellen als eine zwingende Voraussetzung für die Entwicklung sicherer informationstechnologischer Systeme dar. CIDAS erfüllt auch diese Voraussetzung, alle Entwicklungsarbeiten am Projekt sind im Internet dokumentiert und für jeden zugänglich. Ob das System einer Analyse durch Security-Experten und der Evaluation durch Benutzer standhalten kann, wird die Zukunft zeigen.

Kapitel 4

Verwendung passiver mobiler Speichermedien und asymmetrischer Kryptographie zur Benutzerauthentifizierung

*A colleague once told me that the world was full of bad
security systems designed by people who read Applied Cryptography.
— Bruce Schneier, *Secrets and Lies* ([98, S. xii])*

Verwendung passiver mobiler Speichermedien und asymmetrischer Kryptographie zur Benutzerauthentifizierung

In diesem Kapitel soll ein kryptographisches Protokoll zur Authentifizierung von Benutzern konzipiert werden.

4.1 Anforderungen an die Authentifizierung

Das zu konzipierende Verfahren muß im Besonderen den in diesem Abschnitt aufgeführten Anforderungen genügen. Diese betreffen vorrangig die verwendbaren Speichermedien, die Art der zu speichernden Daten sowie die Sicherheit der Authentifizierung.

Die hier wiedergegebene Zusammenstellung von Anforderungen entstand im Verlauf mehrerer Treffen sowie im Austausch von E-Mails zwischen dem Autor und den Leitern des Projektes CIDAS, F. Holl und I. Schäfer. Da eine vollständige Dokumentation des Zustandekommens der Anforderungen den Rahmen der Arbeit bei weitem sprengen würde, verzichtet der Autor an dieser Stelle darauf. Bei Interesse können zusätzliche Informationen hierzu beim Autor bzw. bei oben genannten Mitarbeitern der Fachhochschule Brandenburg erfragt werden.

4.1.1 Verwendbare Speichermedien

Als im Sinne dieser Arbeit verwendbares Speichermedium soll nach Möglichkeit jedes Medium gelten, das genügend Speicherkapazität zur Aufnahme von für den Authentifizierungsprozeß benötigten Daten bietet. Dies schließt passive Speicher wie Festplatten und Disketten ebenso ein wie aktive Chipkarten. Das Wort „passiv“ bezieht sich hierbei auf das Nichtvorhandensein von Schaltkreisen oder Software zur Durchführung einer Authentifizierung. Chipkarten gelten im Sinne dieser Arbeit als aktive Datenträger. Sie integrie-

ren entsprechend Rankl et al. [88, S. 35-37] bereits starke Sicherheitslogik und können damit verschiedene Authentifizierungsverfahren intern durchführen – ein Bedarf zur Konzeption eines neuen Verfahrens besteht an dieser Stelle nicht. Besonderer Wert soll daher auf moderne, passive mobile Speichermedien wie CD-ROMs nach ISO 9660 [5] bzw. ECMA-119 [3]¹ und USB-Memory-Sticks entsprechend der USB-Spezifikation [19] gelegt werden. Diese sind inzwischen allgemein verfügbar, preiswert und werden von jedem modernen Computersystem unterstützt.

Neben diesen ist auch eine Verwendung von passiven Transpondern^(Glossar) angedacht. Auch sie sind äußerst preiswert, universell einsetzbar und enthalten genügend Speicher für das benötigte Schlüsselmaterial. Verschiedene Hersteller integrieren in ihre Transponder bereits Sicherheitslogik, so daß die hierdurch gebotene Sicherheit vergleichbar mit der von SmartCards sein könnte. Leider sind für den Einsatz von Transpondern spezielle Lesegeräte nötig.

Bei den eben erwähnten Authentifizierungsmedien handelt es sich teilweise um identifizierbare Speichermedien.

Identifizierbare Speichermedien Unter einem identifizierbaren Speichermedium wird ein Medium verstanden, das durch das Vorhandensein von durch den Benutzer nicht oder nur mit großem Aufwand verfälschbaren Markierungen eindeutig zuordenbar ist. Solche Markierungen sind beispielsweise fest im Gerät bzw. im Medium eingebrachte Produkt- und Seriennummern. Diese Identifikationsmerkmale der jeweiligen Speichermedien sollen genutzt werden, um ein Medium eindeutig als ein von einem autorisierten Benutzer, dem Besitzer des Mediums, erstelltes Authentifizierungsmedium zu identifizieren und damit die Verwendung nichtautorisierter Kopien auszuschließen.

¹Datenträger- und Dateistruktur von CD-ROMs werden im Allgemeinen entsprechend den Spezifikationen in ISO 9660 ([5]) angegeben. Da dieser Standard dem Autor nicht vorlag, basieren die in diesem Dokument gemachten Angaben bezüglich CD-ROMs auf der zweiten Auflage von ECMA-119 ([3]). Die Angaben in ECMA-119 sind den Verfassern zufolge technisch identisch mit denen in ISO 9660.

4.1.2 Sicherheit der Daten auf einem Speichermedium

Entsprechend Schäfer [93, S. 70] müssen personenbezogene Daten der Benutzer nicht zwangsläufig auf dem CIDAS-Server gespeichert sein. So soll bei Verwendung mobiler Speichermedien die Möglichkeit bestehen, lediglich Metadaten über den eigentlichen Benutzerdaten auf dem Server abzulegen. Die personenbezogenen Benutzerdaten verbleiben hierbei auf dem zur Authentifizierung verwendeten Speichermedium des Benutzers.

Daher sind alle auf einem Speichermedium befindlichen und möglicherweise schützenswerten Daten durch die Verwendung starker Kryptographie zu schützen. Dies bezieht sich neben den personenbezogenen Daten auch auf die privaten Schlüssel (siehe Abschnitt 4.1.4) der Benutzer. Zum Schutz letzterer kommen standardisierte symmetrische Algorithmen wie 3DES², AES³ oder Blowfish nach Schneier [95] zum Einsatz. Die personenbezogenen Daten können sowohl durch symmetrische als auch durch asymmetrische Kryptographie geschützt werden.

4.1.3 Einsetzbarkeit in heterogenen Umgebungen

Das Authentifizierungssystem muß, wie von Schäfer in [93, S. 64] gefordert, in heterogenen Netzwerkumgebungen einsetzbar sein. Es müssen sowohl Workstation-Logons als auch Logons für Web-Portale im Internet realisierbar sein. Daher müssen clientseitig möglichst viele gängige Betriebssysteme unterstützt werden.

4.1.4 Verwendung von asymmetrischer Kryptographie

Um die in Abschnitt 2.3.4 beschriebenen Probleme beim Austausch von symmetrischem Schlüsselmaterial zu vermeiden, soll eine jede Authentifizierung

²3DES ist eine auf DES entsprechend FIPS 46 [105] basierende Chiffre im EDE-Modus. Vgl. hierzu Schneier, [96, S. 362 f.].

³Gemeint ist Rijndael nach Daemen et al. [48], standardisiert in FIPS 197 [20].

auf Basis von asymmetrischer Kryptographie erfolgen. Weil asymmetrisches Schlüsselmaterial in der Regel zu lang ist, als daß eine Person es sich einfach merken und gegebenenfalls eingeben kann, (siehe ebenfalls Abschnitt 2.3.4), soll es auf dem Authentifizierungsmedium gespeichert werden. Für die Durchführung der Authentifizierung wird sowohl der öffentliche als auch der private Schlüssel eines Benutzers benötigt. Beide sollten sich auf dem zur Authentifizierung verwendeten Datenträger befinden und mit diesem derart „verbunden“ sein, daß nur der zugehörige Benutzer⁴ in der Lage ist, einen neuen zur Authentifizierung verwendbaren Datenträger zu erstellen. Ein von einem nicht dazu berechtigten Benutzer erstellter Datenträger soll vom System als solcher erkannt werden. Siehe hierzu auch Abschnitt 4.1.1.

Public-Key-Infrastructure Die Verwendung asymmetrischer Kryptographie bietet sich insbesondere deshalb an, weil CIDAS auf einem Verzeichnisdienst basiert und damit hervorragende Voraussetzungen zum Aufbau einer *Public-Key-Infrastructure* (PKI), also einer hierarchischen Infrastruktur zur Speicherung von beglaubigten öffentlichen Schlüsselmaterials (vgl. Eckert, [52, S. 329]), bietet. Schlüsselmaterial wird hierbei nur dann in den Datenbestand aufgenommen, wenn entsprechende Regelungen, die Policies der PKI, erfüllt sind. Diese Regelungen könnten beispielsweise den persönlichen Kontakt zwischen einem Benutzer und der Registrierungsstelle des Authentifizierungsservers verlangen (vgl. Pohlmann, [83]). Betreiber von CIDAS-Servern werden damit in die Lage versetzt, Trust-Center Funktionalitäten anbieten zu können. Konkrete Konzepte zur Aufnahme von Benutzern und deren Authentifizierungsdaten werden im Rahmen einer weiteren Diplomarbeit in Kürze an der Fachhochschule Brandenburg vorgestellt werden.

Standardisierte Verfahren und Formate Um die Funktionsweise des Systems nachvollziehbar und mit bestehenden Infrastrukturen kompatibel zu halten, müssen standardisierte Verschlüsselungsverfahren einge-

⁴Damit ist der Benutzer gemeint, dem das betreffende Schlüsselmaterial im CIDAS-Server zugeordnet ist.

setzt werden. Schlüsselmaterial kann entweder als X.509^(Glossar)/PEM^(Glossar)-Zertifikat oder in OpenPGP^(Glossar)-konformer Form vorliegen.

4.1.5 Bindung der Authentifizierung an eine Interaktion des Benutzers

Ferner muß, um ein hohes Maß an Sicherheit zu gewährleisten, eine korrekt durchgeführte Authentifizierung immer an eine direkte Interaktion mit dem jeweiligen Benutzer gebunden sein. Die bloße Präsenz eines validen Datenträgers mit validem Schlüsselmaterial darf nicht ausreichen, um den Authentifizierungsprozeß erfolgreich zu durchlaufen. Dies resultiert unter anderem daraus, daß die Validität des Schlüsselmaterials erst nach seiner Entschlüsselung, die wiederum mit einer Passwordeingabe durch den Benutzer verbunden ist, überprüfbar ist.

Die Durchführung einer Authentifizierung ausschließlich über die Nutzung der auf dem Speichermedium vorhandenen Identifizierungsdaten ist durchaus möglich, jedoch nicht Gegenstand dieser Arbeit. Der Ansicht des Autors nach wäre ein solches Vorgehen von der gebotenen Sicherheit her maximal vergleichbar mit einer Passwordeingabe. Es sei darauf hingewiesen, daß ein Angreifer in diesem Fall nur einige Sekunden Zugang zum verwendeten Speichermedium benötigt, um alle relevanten Authentifizierungsdaten zu erhalten.

4.1.6 Einfache Bedienbarkeit, Single-Sign-On

Das zu konzipierende Verfahren soll aus Sicht des Benutzers eine einfache, klar erkennbare und strukturierte Abfolge von Arbeitsschritten erfordern. Komplizierte, undurchsichtige Vorgehensweisen entsprechen nicht den von Schäfer in [93, S. 63-70] spezifizierten Grundgedanken des Identity Management Systems CIDAS. Insbesondere sollte die Eingabe mehrerer verschiedener Passworte vermieden werden. Samar et al. weist in OSF-RFC ([92]) explizit darauf hin, daß ein Sicherheitssystem, soll es erfolgreich sein, einfach

bedienbar sein muß. Anwender sind andernfalls dazu gezwungen, die Funktionsweise des Systems erst zu erlernen. Dies führt, wie Whitten et al. in [109] erläutert, häufig zu erheblichen Problemen, schlimmstenfalls sogar dazu, daß eine Security-Lösung nicht eingesetzt werden kann.

Das Verfahren soll im Rahmen einer Single-Sign-On Umgebung entsprechend Abschnitt 2.2.3 und Kapitel 3 einsetzbar sein.

4.1.7 Offene Spezifikationen und Quellen

Der Quellcode einer Implementierung des im Folgenden konzipierten Verfahrens soll, ebenso wie die Verfahrensspezifikation selbst, grundsätzlich von jedem einsehbar sein. Gründe hierfür sind in Abschnitt 2.2.1.

4.2 Konzeption des Authentifizierungsverfahrens

Im Folgenden sollen Ansätze beschrieben werden, die dem Autor geeignet scheinen, die obige Anforderungen umzusetzen.

4.2.1 Identifizierung von Datenträgern

Wie in Abschnitt 4.1.4 dargestellt, sollen Schlüsselmaterial und Datenträger auf geeignete Weise miteinander „verbunden“ werden. Um dies zu realisieren bieten sich zwei Möglichkeiten an, die beide auf der Erstellung von digitalen Signaturen^(Glossar) über zur Datenträgeridentifizierung geeignete Daten wie Hersteller-, Produkt- und Seriennummern basieren. Welche Daten hierfür konkret verwendet werden, ist abhängig von der Art des verwendeten Authentifizierungsmediums. In den Abschnitten 4.2.5, 4.2.6, 4.2.7 wird hierauf näher eingegangen.

Clientseitige Datenträgeridentifizierung Zum einen kann die Identifizierung eines Authentifizierungsmediums über das einmalige Signieren der betreffenden Identifizierungsdaten realisiert werden. Die Signatur wird vom autorisierten Benutzer unter Verwendung seines zur Authentifizierung verwendeten privaten Schlüsselmaterials selbst erzeugt und kann im Nachhinein von einem CIDAS-Client mit einem vom Authentifizierungsserver abgerufenen öffentlichen Schlüssel des Benutzers überprüft werden. Schlägt die Überprüfung fehl, wird ein korrekter⁵ CIDAS-Client dem Benutzer mitteilen, daß er einen möglicherweise nicht von einem autorisierten Benutzer erzeugten Datenträger verwendet, worauf der Benutzer seinem Sicherheitsbedürfnis entsprechend reagieren kann. Die konkrete Art und Weise der Erstellung der

⁵Unter einem korrekten CIDAS-Client wird in diesem Zusammenhang die Implementierung der clientseitigen Funktionalität von CIDAS entsprechend Schäfer [93] und dieser Arbeit verstanden, die nicht durch zusätzliche, bewußt schadhafte Funktionalität die Sicherheit der Authentifizierungsverfahren und damit die Sicherheit der Benutzer gefährdet. Siehe auch Kowalk, [72].

Signatur wird in Abhängigkeit von der Art des verwendeten Schlüsselmaterials, entweder in RFC 2440 ([43]) bzw. in X.509 ([44]) spezifiziert.

Der Authentifizierungsserver muß bei der Verwendung dieses Verfahrens über keine Daten bezüglich der verwendeten Authentifizierungshardware verfügen. Tatsächlich ist die Durchführung dieser Art von Datenträgeridentifizierung aus seiner Sicht irrelevant. Der Server kommuniziert über ein Netzwerk mit dem entfernten Client, ausschließlich dieser hat Zugriff auf die zur Authentifizierung verwendete Hardware. Da ein Authentifizierungsserver keine Möglichkeit hat, die Korrektheit einer Client-Implementierung zu überprüfen, hat er auch keinen Grund, auf die von einem Client übertragenen Daten zu vertrauen, es sei denn, er kann deren Korrektheit über kryptographische Verfahren beweisen. Resultierend daraus ist es irrelevant, ob ein Client die Daten der Hardware an den Server überträgt und dieser sie dann überprüft oder ob der Client die Auswertung selbst durchführt – der Server wird ihm weder das eine noch das andere glauben.

Serverseitige Datenträgeridentifizierung Um eine serverseitige Datenträgeridentifizierung zu realisieren, benötigt der Authentifizierungsserver die entsprechenden Identifizierungsdaten des verwendeten Mediums. Sofern diese nicht „aus erster Hand“, also über einen direkten Zugriff auf das Medium, oder durch eine vertrauenswürdige Person in den Server eingepflegt werden, muß er die Korrektheit der Daten ohne einen Beweis voraussetzen oder, was hierbei als logischer erscheint, sie als eine arbiträre Datenmenge ohne jeden Bezug zum autorisierten Benutzer und dessen Authentifizierungsmedium ansehen. Selbst wenn die Daten von einem authentifizierten Benutzer mit der Bemerkung „Dies sind die Identifikationsdaten meines Speichermediums“ übertragen wurden, werden sie aus Sicht des Authentifizierungsservers nicht glaubhafter. Zum einen sagt auch ein authentifizierter Benutzer nicht zwangsläufig immer die Wahrheit, zum anderen weiß der Server nicht, wie die Daten zustande kamen, sie könnten korrekt oder auch die Ausgabe einer absichtlich manipulierten CIDAS-Client-Software sein – selbst der ehrlichste

Benutzer wird in der Regel nicht die Serien- und Herstellernummer seines USB-Sticks kennen und mit den zu übertragenden Daten vergleichen.

Im Folgenden soll nun davon ausgegangen werden, daß dem Authentifizierungsserver die Identifizierungsdaten des verwendeten Mediums zweifelsfrei bekannt sind. Es wäre beispielsweise denkbar, daß der Betreiber des CIDAS-Servers USB-Memory-Sticks an seine Kunden ausgegeben hat. Unter dieser Voraussetzung kann das Authentifizierungsmedium möglicherweise durch das Übersenden einer Signatur über den aktuell aus dem Medium ausgelesenen Identifizierungsdaten identifiziert werden. Die Aktualität dieser Signatur kann über das Auswerten des enthaltenen Zeitstempels realisiert werden.

Dieser auf den ersten Blick vielversprechende Ansatz bietet bei genauerer Betrachtung leider keinerlei zusätzliche Sicherheit: Es ist aus Sicht des Authentifizierungsservers nicht nachprüfbar, ob die Daten tatsächlich von einem eingelegten bzw. angeschlossenen Medium stammen, sie könnten ebenso gut aus einer Datei gelesen worden sein, schließlich weiß der Server nicht, ob sich die verwendete Client-Software korrekt verhält. Auch ist die Korrektheit der Signatur nicht wirklich nachprüfbar, sie könnte bereits vor längerer Zeit an einem Arbeitsplatz mit verstellter Systemzeit⁶ ohne Wissen des Benutzers erstellt worden sein.

Auch wenn sich diese Zweifel durch die Einbindung der Identifizierungsdaten des Speichermediums in das in Abschnitt 4.2.3 konzipierte Authentifizierungsprotokoll beseitigen ließen, bleibt der Zweifel am tatsächlichen Vorhandensein des bezeichneten Speichermediums im Clientsystem bestehen. Lediglich ein direkter Zugriff auf die Hardware des Clients seitens des Authentifizierungsservers könnte an dieser Stelle Vertrauen schaffen. Dieser ist jedoch aufgrund der physikalischen Trennung beider Systeme nicht möglich.

⁶Dem Autor ist bewußt, daß dieser Angriff sehr viel Planungsarbeit, exaktes Timing und Zugriff auf einen vom Opfer benutzten Arbeitsplatz erfordert. Er dient jedoch lediglich dazu, zu zeigen, daß das erläuterte Vorgehen dem Authentifizierungsserver keinen wirklichen Grund zu der Annahme, die empfangenen Daten würden einen Datenträger identifizieren, bietet.

Fazit Im Ergebnis dieser Betrachtung läßt sich feststellen, daß die Identifizierung der Authentifizierungsmedien lediglich durch den CIDAS-Client, nicht aber durch den Authentifizierungsserver durchgeführt werden kann. Sie eignet sich trotzdem dazu, Angriffe zu erschweren und ermöglicht es dem Benutzer, Authentifizierungsmedien eindeutig als von ihm erstellt zu erkennen.

4.2.2 Auswahl des Authentifizierungsverfahrens

Als äußerst schwierig erwies sich die Wahl eines geeigneten Public-Key-Authentifizierungsverfahrens unter Berücksichtigung der im vorherigen Abschnitt aufgestellten Anforderungen. Der Autor mußte feststellen, daß auf asymmetrischer Kryptographie basierende Verfahren zur *User-To-Host*-Authentifizierung derzeit sehr selten eingesetzt werden. Darüber hinaus ergab eine Recherche in zu den wenigen vorhandenen Protokollen durchgeführten Analysen verschiedene Fehler und Probleme in den jeweiligen Verfahren. Diese werden im Folgenden vorrangig am Beispiel der Public-Key-Version des Needham-Schroeder Protokolls entsprechend Needham et al. [81] und des in CCITT X.509 ([44]) verwendeten Authentifizierungsprotokolls erläutert. Schneier weist in [96, S. 61-64] auf verschiedene weitere Verfahren mit ähnlichen Problemen hin.

Ungeeignete Verfahren

In Hinblick auf den Bestimmungszweck der jeweiligen Verfahren erwiesen diese sich immer als teilweise ungeeignet.

Das **Needham-Schroeder-Protokoll** ist, ebenso wie die meisten anderen untersuchten Verfahren, primär für den sicheren Schlüsselaustausch konzipiert (vgl. Needham et al., [81]; Burrows et al., [42]). Es soll eingesetzt werden, um eine wechselseitige Authentifizierung zwischen zwei Kommunikationspartnern und die gesicherte Übertragung von symmetrischen Schlüsselmaterial zu realisieren. Dies ist wesentlich mehr als im Rahmen von CIDAS benötigt wird. Entsprechend Schäfer [93, S. 90 f.] muß lediglich eine Authen-

tifizierung des Benutzers, nicht das CIDAS-Servers erfolgen⁷. Auch ist ein Schlüsselaustausch nicht erforderlich – der Beweis der Identität des Benutzers genügt dem CIDAS-Server.

Das **CCITT X.509-Protokoll** wiederum wickelt parallel zur Authentifizierung zweier Kommunikationspartner einen Nachrichtenaustausch zwischen zwei Kommunikationspartnern ab (vgl. X.509, [44]; Burrows et al., [42]). Auch dies ist beim Einsatz in CIDAS nicht erforderlich: Der Nachrichtenaustausch erfolgt nach dem von Schäfer in [93, S. 90-96] bzw. in Kapitel 5 dieser Arbeit spezifizierten Kommunikationsprotokoll, welches nicht zwangsläufig die Verwendung von asymmetrischem Schlüsselmaterial voraussetzt.

Fehler und Sicherheitslücken in den Verfahren

Darüber hinaus wiesen alle untersuchten Protokoll gravierende Sicherheitsmängel auf. Zum Needham-Schroeder-Protokoll sind beispielsweise verschiedene Angriffsmöglichkeiten von Lowe in [74] und [75], sowie von Burrows et al. in [42] beschrieben. Burrows et al. zeigt in [42] auch eine Sicherheitslücke im CCITT X.509-Protokoll auf, die es einem Angreifer erlaubt, sich ohne die Kompromittierung des Schlüsselmaterials einer Partei als diese auszugeben. Schneier beschreibt in [96, S. 63] einen ähnlichen Angriff auf das Public-Key-Authentifizierungsverfahren Denning-Sacco, auch beschrieben von Yasinsac in [112]. Zu verschiedenen weiteren Protokollen lagen dem Autor keine Analysen vor.

Obige Überlegungen und Recherchen resultierten in der Idee zur Konzeption eines eigenen Verfahrens, welches im folgenden Abschnitt (4.2.3) ausführlich beschrieben wird. Dem Autor ist bewußt, daß dieses Vorgehen zwar zum einen die Vermeidung der in bestehenden Lösungen vorhandenen Probleme erlaubt, zum anderen jedoch mit dem Risiko des Hervorrufens neuer Probleme und Sicherheitslücken verbunden ist.

⁷Der Server wird über andere Verfahren authentifiziert. Laut Schäfer [93, S. 90 f.] kommen hierfür beispielsweise die VPN-Technologie oder das SSL-Protokoll zum Einsatz.

4.2.3 Ablauf einer Authentifizierung

Eine Authentifizierung, egal ob mittels OpenPGP- oder X.509-Schlüsselmaterial, läuft immer nach genau dem selben Verfahren ab, bei dem Client (P) und der Authentifizierungsserver (V) miteinander Nachrichten austauschen. Dabei wird davon ausgegangen, daß P und V sich über das zu verwendende Public-Key-Verschlüsselungssystem und ein symmetrisches Verschlüsselungsverfahren einig sind⁸. Darüber hinaus muß P im Besitz eines validen Schlüsselpaares, also eines geheimen Schlüssels K_P^{-1} und eines öffentlichen Schlüssels K_P sein. Den öffentlichen Schlüssel muß V im Voraus, beispielsweise bei der Eintragung des Benutzers ins System, erhalten haben und auf seine Korrektheit vertrauen. Es wird im Folgenden die bereits aus Kapitel 2 bekannte Notation verwendet.

Es sei darauf hingewiesen, daß V im Laufe des im Folgenden beschriebenen Verfahrens seine Identität nicht nachweisen muß. Da das Authentifizierungsprotokoll für den in die Kommunikation zwischen CIDAS-Client und -Server eingebetteten Einsatz gedacht ist, kann davon ausgegangen werden, daß eine Authentifizierung des Servers bereits vorher ablief und der Client dem Server nun vertraut. Auch wurden Identifizierungsdaten bereits im Voraus ausgetauscht, wodurch dem Server die Identität des Benutzers in Form einer Behauptung bekannt ist. Der Server kann dementsprechend zum Benutzer gehörendes Schlüsselmaterial für den Authentifizierungsprozeß auswählen und verwenden, sofern ihm dieses aus einer zuverlässigen Quelle⁹ zur Verfügung steht. Problematisch ist das jedoch für den Fall, daß dem Server inkorrekte Identifizierungsdaten¹⁰ vorliegen – in Abschnitt 4.3 finden sich zusätzliche Erläuterungen hierzu.

⁸Die Standards RFC 2440 ([43]) und X.509 ([44]) spezifizieren jeweils die Art der Verwendung verschiedener symmetrischer Verschlüsselungsverfahren, weshalb hierrauf nicht weiter eingegangen wird.

⁹Hierbei ist die Verwendung einer Public-Key-Infrastructure mit restriktiven Regeln zwingend erforderlich.

¹⁰Das weiter unten beschriebene Protokoll könnte unter diesen Umständen nicht durchgeführt werden, weil der Authentifizierungsserver hierfür über das öffentliche Schlüsselmaterial verfügen muß.

Nach Ablauf des Protokolls ist V entweder davon überzeugt, daß P tatsächlich im Besitz des zu K_P gehörigen geheimen Schlüssels ist, womit aus der Sicht des V seine Identität bewiesen wäre, oder, daß P nicht über dieses Schlüsselmaterial verfügt und damit nicht der ist, als der er sich auszugeben versucht. Es wird folglich nicht direkt bewiesen, daß der sich authentifizierende Benutzer tatsächlich der ist, als der er sich ausgibt, sondern lediglich, daß er über das zu der entsprechenden Identitätsbeschreibung passende Schlüsselmaterial verfügt. Ausgehend davon, daß ein Benutzer seinen privaten Schlüssel unter allen Umständen geheim hält und das verwendete asymmetrische Verschlüsselungsverfahren keine Fehler enthält, die beispielsweise eine Rekonstruktion des geheimen Schlüssels aus dem öffentlichen erlauben, ist die Annahme, daß der authentifizierte Benutzer tatsächlich auch der identifizierte ist, mehr als berechtigt.

Das folgende Protokoll ist einem von Schneier in [96, S. 54.] publizierten Beispielverfahren zur Authentifizierung mittels Public-Key-Kryptographie entlehnt. Es wurde dahingehend modifiziert, daß P niemals gezwungen wird, von V erzeugte oder möglicherweise aktiv manipulierte Zeichenketten mit seinem privaten Schlüssel zu verschlüsseln. Das Protokoll wird in vier Schritten durchlaufen. Ferner dienen auch das von Needham et al. in [81] beschriebene Public-Key-Verfahren sowie verschiedene Analysen existierender Verfahren, insbesondere in Burrows et al. ([42]) und Yasinsac et al. ([112]) als Beispiele.

(1): P erzeugt zwei Mengen Zufallsdaten M_P und K . Anschließend verschlüsselt P die erste Zufallszahlenmenge M_P mittels eines symmetrischen Verfahrens. Als Schlüssel werden die ersten n Bits von K verwendet, wobei n gleich der benötigten Schlüssellänge des symmetrischen Verfahrens ist. Das Ergebnis dessen wird von P mit einem Zeitstempel¹¹ t_{M_P} versehen und von P

¹¹Der Zeitstempel wird vom CIDAS-Client des zu authentifizierenden Benutzers erzeugt. Dieses Vorgehen ist vergleichbar mit dem Kerberos-Protokoll entsprechend RFC

mit seinem privaten Schlüssel unter Verwendung eines Public-Key-Verfahrens verschlüsselt. Das daraus resultierende $C_P = \{t_{M_P}, \{M_P\}_K\}_{K_P^{-1}}$ überträgt P an V .

(2): V kann die empfangenen Daten nur teilweise entschlüsseln und ihre Aktualität überprüfen: $t_{M_P}, \{M_P\}_K = \{C_P\}_{K_P}$. M_P steht ihm aufgrund des Fehlens von K erst einmal noch nicht zur Verfügung. V erzeugt selbst eine Menge Zufallsdaten M_V , bildet $C_V = \{M_V\}_{K_P}$ und sendet es an P .

(3): Falls also P tatsächlich im Besitz von K_P^{-1} ist, kann er $M_V = \{C_V\}_{K_P^{-1}}$ entschlüsseln. Anschließend berechnet er $\Delta M = M_P \oplus M_V$ und verschlüsselt ΔM und K mit seinem geheimen Schlüssel. Das Resultat $C_{\Delta M} = \{t_{\Delta M}, \Delta M, K\}_{K_P^{-1}}$ sendet er an V .

(4): Im letzten Schritt kann sich V von der Identität des P überzeugen: $t_{\Delta M}, \Delta M, K = \{C_{\Delta M}\}_{K_P}$; mit K kann M_P aus Schritt 1 entschlüsselt werden und falls $\Delta M = M_V \oplus M_P$ gilt, ist P tatsächlich im Besitz von K_P^{-1} und damit aus Sicht des V der, der er zu sein vorgibt.

Das Protokoll verwendet bei mehreren Nachrichten Zeitstempel, bezeichnet mit $t_{\langle \text{Nachrichtennamenname} \rangle}$. Diese dienen als Überprüfungsmöglichkeit bezüglich der Aktualität einer Nachricht für den Kommunikationspartner und müssen ausgewertet werden. Zeitstempel bezeichnen den aktuellen Zeitpunkt der Erzeugung der jeweiligen Nachricht. Sie werden vom CIDAS-Client des zu authentifizierenden Benutzers erzeugt. Dieses Vorgehen ist vergleichbar mit dem Kerberos-Protokoll entsprechend RFC 1510 ([71]), es werden synchrone Systemuhren zwischen Client und Server vorausgesetzt.

Die verwendeten Zufallsdaten werden vom CIDAS-Client erzeugt oder aus einer dafür vorgesehenen Betriebssystem-Ressource, beispielsweise dem 1510 ([71]), es werden synchrone Systemuhren zwischen Client und Server vorausgesetzt.

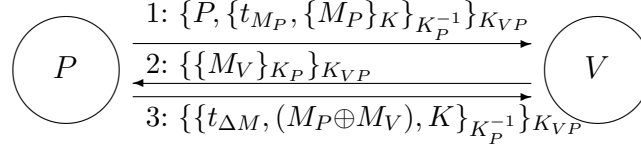


Abbildung 4.1: Nachrichtenaustausch im vorgestellten Protokoll.

/dev/random-Gerät ausgelesen (vgl. Single Unix Specification, [24]). Für eine Software-Implementierung eines guten Pseudo-Zufallsgenerators werden in Schneier, [96, S. 420-428] verschiedene Hinweise gegeben und Verfahren erläutert.

Die Analyse des Protokolls

Um das Verfahren auf seine Korrektheit und Vollständigkeit hin zu überprüfen und unnötige Aktionen zu finden, wird es im Folgenden entsprechend der BAN-Logik nach Burrows et al. [42] notiert. Es werden drei Nachrichten übermittelt. Abbildung 4.1 auf Seite 59 verdeutlicht den Nachrichtenfluß nochmals:

Nachricht 1¹² $P \rightarrow V: \{P, \{t_{M_P}, \{M_P\}_K\}_{K_P^{-1}}\}_{K_{VP}}$

Nachricht 2 $V \rightarrow P: \{\{M_V\}_{K_P}\}_{K_{VP}}$

Nachricht 3 $P \rightarrow V: \{\{t_{\Delta M}, (M_P \oplus M_V), K\}_{K_P^{-1}}\}_{K_{VP}}$

¹²Die Übertragung von P in Nachricht 1 ist eine Art Kunstgriff. Sie repräsentiert die Identifizierung, also den Teil der Kommunikation zwischen Client und Server, der bereits vorher ablief und in dem P seine Identifizierungsdaten an den Authentifizierungsserver übermittelte. Ähnliches gilt für die Verschlüsselung der gesamten Nachricht mit dem Schlüssel K_{VP} . Dieser ist ein nur zwischen P und V bekannter Schlüssel, er drückt die Sicherung der zwischen den Kommunikationspartnern übertragenen Daten durch SSL oder vergleichbare Verfahren aus.

Aus der Ausgangssituation bzw. den einzelnen Protokollschritten kann nun nach den von Burrows et al. in [42] aufgestellten *Message-Meaning-Rules* abgeleitet werden:

P und V kennen beide den geheimen Sitzungsschlüssel K_{VP} . Siehe auch Fußnote 12:

$$P \equiv P \xleftrightarrow{K_{VP}} V \quad V \equiv V \xleftrightarrow{K_{VP}} P$$

P kennt seinen eigenen Secret- und Public-Key, V kennt den Public-Key von P , die Verwendung einer Publik-Key-Infrastruktur wird vorausgesetzt.

$$\begin{array}{ll} P \equiv \vdash^{K_P} P & P \equiv \vdash^{K_P} V \\ V \equiv \vdash^{K_P} P & V \equiv \vdash^{K_P} V \end{array}$$

Des Weiteren glauben beide an die Aktualität der eigenen Zufallsdaten, sie haben diese schließlich selbst erzeugt. Sie glauben ebenso an die Aktualität der Zeitstempel des Kommunikationspartners, was aus der Verschlüsselung des Zeitstempels mit K_P^{-1} resultiert. Das Verfahren setzt dementsprechend synchrone Systemzeiten zwischen P und V voraus.

$$\begin{array}{ll} P \equiv \sharp(M_P) & V \equiv \sharp(M_V) \\ P \equiv \sharp(K) & V \equiv \sharp(t_{M_P}) \\ & V \equiv \sharp(t_{\Delta M}) \end{array}$$

Aus Nachricht 1 kann nun abgeleitet werden:

$$V \equiv P \sim \{M_P\}_K \quad V \equiv P \equiv \{M_P\}_K$$

Es sei darauf hingewiesen, daß $V \triangleleft M_P$ hier nicht ableitbar ist, weil V den Schlüssel K nicht kennt und P prinzipiell weder M_P noch K besitzen muß, um $\{M_P\}_K$ mit K_P^{-1} zu verschlüsseln.

Aus Nachricht 2 wird abgeleitet:

$$P \equiv V \sim M_V \qquad P \triangleleft M_V$$

P kann also unter Verwendung des geheimen Sitzungsschlüssels und seines privaten Schlüssels M_V entschlüsseln.

Mit Nachricht 3 erhält V zum einen K , zum anderen $\Delta M = M_P \oplus M_V$ daher läßt sich ableiten:

$$\begin{array}{ll} V \triangleleft K & V \triangleleft M_P \\ V \equiv P \sim K & V \equiv P \equiv K \\ V \equiv P \sim M_P & V \equiv P \equiv M_P \\ V \triangleleft \Delta M & V \equiv P \sim \Delta M \end{array}$$

Nach der Verifikation der Korrektheit von ΔM durch V ergibt sich ferner:

$$V \equiv \#(\Delta M) \qquad V \equiv P \triangleleft M_V$$

Aus Sicht des Authentifizierungsservers ist damit die Identität von P bewiesen.

Selbstverständlich mußte P dafür keinerlei nichtarbiträre Daten mit seinem privaten Schlüssel verschlüsseln. Selbst wenn sich V nicht an das Protokoll hält und ein bereits benutztes M_V verwendet, ergeben sich keine Sicherheitseinbußen für P , weil dieser lediglich $M_P \oplus M_V$ verschlüsselt, wobei M_P dem V bei der Erstellung von M_V nicht bekannt war. Probleme ergeben sich an dieser Stelle lediglich, wenn V in der Lage ist, M_P ohne K zu entschlüsseln, er könnte dann ein M_V derart wählen, daß $M_P \oplus M_V$ eine Nachricht ergibt, die er von P verschlüsselt haben möchte.

Das vorgestellte Verfahren kann jedoch nicht garantieren, daß der dem Clientsystem vorliegende und zur Authentifizierung genutzte Datenträger mit Schlüsselmaterial tatsächlich ein vom autorisierten Benutzer erzeugter Datenträger ist. Dies kann ausschließlich der Client selbst überprüfen. Da der

Authentifizierungsserver jedoch keinen Grund hat, einem Client in dieser Frage zu vertrauen, er selbst auch keinen direkten Zugriff auf die Hardware des Clients durchführen kann, ist dieses Problem aus Sicht des Autors nicht zu lösen.

Auch kann nicht sichergestellt werden, daß das zur Authentifizierung verwendete Schlüsselmaterial von der tatsächlich autorisierten Person verwendet wird. Vielmehr wird davon ausgegangen, daß Benutzer ihre Authentifizierungsdatenträger und insbesondere das darauf gespeicherte Schlüsselmaterial durch symmetrische Kryptographie und vor dem Zugriff durch Dritte geschützt aufbewahren.

4.2.4 Auf einem Authentifizierungsmedium zu speichernde Daten

Unabhängig von der Art des verwendeten Speichermediums müssen bestimmte Daten darauf gespeichert werden. Hierbei handelt es sich vorrangig um das zur Authentifizierung benötigte Schlüsselmaterial und vom verwendeten Public-Key-Verschlüsselungssystem verwendete Metadaten: Asymmetrisches Schlüsselmaterial besteht, wie bereits in Abschnitt 2.3.4 dargestellt aus einer größeren Menge arbiträrer Zeichen. Benutzer werden sich diese in der Regel nicht merken können, müssen sie jedoch um obiges Protokoll zu durchlaufen, verfügbar haben. Darüber hinaus bietet CIDAS nach Schäfer ([93, S. 70]) noch die Möglichkeit, personenbezogene Daten des Benutzers auf dem Medium abzulegen, falls dem Benutzer eine Speicherung auf dem Authentifizierungsserver zu unsicher ist. Insbesondere das geheime Schlüsselmaterial und diese Daten müssen auf geeignete Weise, beispielsweise durch kryptographische Verfahren, vor unautorisiertem Zugriff und Vervielfältigung geschützt und an einer von einem CIDAS-Client auffindbaren, bestenfalls immer gleichen Stelle auf dem Datenträger gespeichert werden.

Das notwendigerweise zu speichernde Datenvolumen beläuft sich auf maximal 10 KiB¹³ pro Schlüsselpaar. Dies entspricht in etwa einem vom Autor unter Verwendung der Open-Source-Software *GNU Privacy Guard*¹⁴ erzeugten OpenPGP-Schlüsselpaar, bestehend aus jeweils einem 1024-Bit-DSA-Schlüsselpaar und einem 4096-Bit-ElGamal-Schlüsselpaar¹⁵. Der öffentliche Schlüssel enthielt hierbei zehn Signaturen und 80 Bytes Identifizierungsinformationen wie einen Benutzernamen und eine zugehörige E-Mail Adresse. Das Schlüsselmaterial wurde in ASCII-Armored, Radix-64-kodierter Form entsprechend RFC 2440 [43, S. 7, 41-48] gespeichert. Bei der Verwendung von PEM/X.509 werden vergleichbare Speicherkapazitäten benötigt, was vom Autor durch die Erzeugung entsprechenden Schlüsselmaterials mit der Open-Source Software *OpenSSL*¹⁶ überprüft wurde.

In der Regel¹⁷ werden jedoch 5 KiB für das Schlüsselmaterial genügen. Für Anwendungen, bei denen Schlüsselmaterial nur für einen verhältnismäßig kurzen Zeitraum, im Bereich von Tagen oder Stunden, existiert und eingesetzt wird, werden auch Schlüssel mit einer Länge von weniger als 1024 Bit ausreichend sein. Werden dabei keine zusätzlichen Signaturen über dem Schlüsselmaterial benötigt, böten sich bei Verwendung von OpenPGP-Schlüsselmaterial auch Datenträger mit etwa einem KiB Speicherplatz als Authentifizierungsmedien an. Es bräuchte dann nämlich nur das *Secret Key Package*, das entsprechend RFC 2440 [43, S. 35] eine Kopie des öffentlichen Schlüssels einschließlich der Identifizierungsdaten und der primären (self-) Signatur enthält, auf dem Datenträger gespeichert werden. Mit

¹³Als Einheit für Speicherkapazitäten von Datenträgern, Datendurchsatzraten, etc. werden die in IEC 60027-2 [18] vorgeschlagenen verwendet. 2^{10} Bytes sind dementsprechend 1 KiB, gesprochen „ein Kibibyte“, oder analog 8 Kib, gesprochen „acht Kibibit“.

¹⁴Der *GNU Privacy Guard* ist eine vom Bundesministerium für Wirtschaft und Technologie bezuschusste Implementierung von OpenPGP entsprechend RFC 2440 [43]. Die Software ist im Internet unter <http://www.gnupg.org> erhältlich.

¹⁵Es werden für unterschiedliche Aufgaben auch unterschiedliche Schlüssel verwendet: Der DSA-Schlüssel entsprechend FIPS 186 [106] dient zum Signieren und Prüfen von Signaturen, der ElGamal-Schlüssel nach ElGamal [53] dementsprechend zum Ver- und Entschlüsseln von Nachrichten.

¹⁶Siehe hierzu <http://www.openssl.org>.

¹⁷Beispielsweise werden laut Schneier [96, S. 162] bis zum Jahr 2015 Schlüssel mit einer Länge von 2048 Bit ausreichend Sicherheit gegen derzeit bekannte Angriffe bieten.

X.509-Zertifikaten ist dieses Vorgehen nicht möglich.

In Abschnitt 4.3.5 wird eine Möglichkeit zur Verbesserung der durch das System gebotenen Sicherheit beschrieben, bei der mehrere Schlüsselpaare, nämlich ein primäres Schlüsselpaar und möglicherweise mehrere kurzlebige Sekundärschlüssel, verwendet werden. Soll dieser Ansatz umgesetzt werden, müssen Datenträger genügend Speicherkapazität für weitere Schlüsselpaare aufweisen.

Neben den Authentifizierungsdaten werden auf dem verwendeten Medium möglicherweise noch personenbezogene Daten des Benutzers, beispielsweise seine Kontaktdaten, Bankverbindungen oder vergleichbares, gespeichert. Da es sich hierbei lediglich um verschlüsselt gespeicherte¹⁸ textuelle Informationen handelt, sollten 2 KiB ausreichend sein. In Abhängigkeit von der Speicherkapazität des verwendeten Datenträgers können natürlich weitere Daten abgelegt werden.

In Abschnitt 4.2.1 wird erläutert, wie Authentifizierungsmedien über die Bildung einer Signatur über bestimmten, nur sehr schwer fälschbaren Daten des Mediums eindeutig identifiziert werden können. Diese Signatur muß natürlich ebenfalls auf dem verwendeten Datenträger abgelegt werden. Sie wird in der Regel weit weniger als ein KiB einnehmen. OpenPGP-Signaturen z.B. haben üblicherweise eine Länge von weniger als 300 Bytes.

Insgesamt ergibt sich damit ein minimaler Bedarf an Speicherkapazität von etwa 1.3 KiB bei Verwendung eines Schlüsselpaares mit nicht mehr als 1024 Bit Schlüssellänge und einer Signatur zur Identifizierung des Authentifizierungsmediums. Soll das Medium längerfristig eingesetzt werden, eine hohe Sicherheit und Speicherplatz für personenbezogene Daten bieten, sollte es mindestens über eine Kapazität von 10 KiB verfügen. Für darüber

¹⁸Die verschlüsselte Speicherung personenbezogener Daten auf dem Authentifizierungsmedium des Benutzers wird von Schäfer in [93, S. 70] erläutert bzw. gefordert.

hinausgehende Funktionalitäten, beispielsweise zur Verwendung von sekundärem Schlüsselmaterial, sind größere Datenmengen zu speichern. Werden Datenträger mit einer Kapazität von mehr als 50 KiB genutzt, wird die gesamte Funktionalität der hier beschriebenen Authentifizierungsinfrastruktur nutzbar sein.

4.2.5 USB-Memory-Sticks

Der Universal Serial Bus, kurz USB, erfreut sich seit seiner Spezifikation 1998 in [13] nicht zuletzt aufgrund der Unterstützung durch Computer- und Peripheriegerätehersteller wachsender Beliebtheit (vgl. Axelson [34, S. 15-30]). Wie es der Name bereits vermuten läßt, ist der USB ein externes und universell einsetzbares Bussystem für Computer.

USB-Memory-Sticks haben allesamt einen sehr ähnlichen Aufbau. Sie bestehen aus einem USB-Controller, der über entsprechende physikalische Verbindungen mit dem USB-Host-Controller in der Workstation kommuniziert. Aufbau und Funktionalität handelsüblicher USB-Controller werden von Axelson in [34, S. 135-140] ausführlich erläutert. Die Kommunikation läuft in Abhängigkeit vom USB-Controller des Memory-Sticks und vom USB-Host-Controller mit einer Geschwindigkeit zwischen 1.5 Mib/s und 480 Mib/s¹⁹. Zugriffe auf den eigentlichen Datenspeicher werden über weitere Controller im Memory-Stick realisiert. Als Speicher kommen entsprechend den Produkt-Informationen vieler Hersteller von Memory-Sticks meist Flash-Speicher²⁰ zum Einsatz, es werden jedoch auch kleine externe Festplatten mit USB-Schnittstelle angeboten. Die Speicherkapazitäten für Flash-Sticks liegen derzeit zwischen 4 MiB und 1 GiB, Geräte mit Festplatte erreichen bereits Kapazitäten von mehreren hundert GBytes. Die Geräte sind verhältnismäßig

¹⁹In Version 1.1 der USB-Spezifikation [13] werden die Übertragungsgeschwindigkeiten 1.5 Mib/s und 12 Mib/s spezifiziert, in Version 2.0 [19] der Highspeed-Modus mit 480 Mib/s. Sowohl USB-Controller als auch USB-Host-Controller sind abwärts kompatibel.

²⁰Ein elektronisch löscher- und beschreibbarer Speicherbaustein mit kurzen Zugriffszeiten, siehe auch Axelson [34, S. 136].

preiswert und können an jedem Computer eingesetzt werden, der über ein PCI-Bussystem und einen USB-Host-Controller verfügt. Sie verfügen üblicherweise über keinerlei kryptographische Funktionalität oder sonstige Sicherheitslogik und entsprechen damit der im Rahmen dieser Arbeit geltenden Definition von einem „passiven Datenträger“.

Wie bereits unter 4.1.1 erläutert wurde, soll das hier vorgestellte Verfahren als zusätzlichen Sicherheitsfaktor Speichermedien identifizieren und damit Medien, die vom ehrlichen Benutzer erzeugt wurden, von ungültigen Kopien unterscheiden können. Bei USB-Memory-Sticks werden hierfür die Felder *idVendor*, *idProduct*, *bcdDevice*, *iManufacturer*, *iProduct* und *iSerialNumber* aus dem Geräte-Deskriptor entsprechend Tabelle 4.1 auf Seite 67 verwendet. Die Felder *iManufacturer*, *iProduct* und *iSerialNumber* sind hierbei Verweise auf String-Deskriptoren, also auf Zeichenketten variabler Länge. Diese sind zwar optional, jedoch wird zumindest die Seriennummer von allen dem Autor bekannten Herstellern angegeben. Dies liegt vor allem daran, daß die Seriennummer für die Verwendung mehrerer gleichartiger Geräte an einem Host-Controller zwingend erforderlich ist. Seriennummern haben der Erfahrung des Autors entsprechend eine Länge von mindestens 48 Bit.

Da die oben genannten Datenfelder auf USB-Medien üblicherweise²¹ in ROMs^(Glossar) gespeichert werden, können sie von Benutzern später nicht mehr modifiziert werden und identifizieren das Speichermedium eindeutig (vgl. Axelson, [34, S. 106]). Dementsprechend würde, wie in Abschnitt 4.2.1 beschrieben, eine digitale Signatur über diesen Daten ein Medium einem Benutzer eindeutig zuordenbar machen. Existieren tatsächlich Speichermedien ohne Seriennummer²², sollten diese von einer CIDAS-Implementierung aus Sicherheitsgründen abgelehnt werden. Die Überprüfung eines USB-Sticks auf seine Validität geschieht durch erneutes Auslesen der Hardware-Informationen und Überprüfen der auf dem Stick gespeicherten Signatur mit

²¹Die Konstruktion von Geräten mit nachträglich modifizierbaren Deskriptoren ist natürlich technisch möglich.

²²Ob die Seriennummer möglicherweise modifizierbar ist, wird nicht feststellbar sein. Die USB-Spezifikation sieht keine Operationen zum Beschreiben der Deskriptorfelder vor.

Offset (dezimal)	Feld	Größe (Bytes)	Beschreibung
0	bLength	1	Größe des Deskriptors in Bytes
1	bDescriptorType	1	die Konstante DEVICE (01h)
2	bcdUSB	2	Release-Nummer der USB-Spezifikation
4	bDeviceClass	1	Klassencode
5	bDeviceSubclass	1	Unterklassencode
6	bdeviceProtocol	1	Protokollcode
7	bMaxPacketSize(0)	1	maximale Paketgröße für den Endpunkt 0
8	idVendor	2	Anbieter-ID
10	idProduct	2	Produkt-ID
12	bcdDevice	2	Release-Nummer des Geräts
14	iManufacturer	1	Index des String-Deskriptors für den Hersteller
15	iProduct	1	Index des String-Deskriptors für das Produkt
16	iSerialNumber	1	Index des String-Deskriptors mit der Seriennummer
17	bNumConfigurations	1	Anzahl möglicher Konfigurationen

Tabelle 4.1: Der Geräte-Deskriptor eines USB-Gerätes nach Axelson [34, S. 105 f.].

dem ebenfalls dort vorhandenen öffentlichen Schlüsselmaterial des Benutzers. Konkret wird die Datenträgersignatur wie folgt gebildet:

(1): Der Device-Deskriptor wird über entsprechende Betriebssystem-Funktionen aus dem USB-Gerät ausgelesen.

(2): Die im Device-Deskriptor über die String-Deskriptoren *iManufacturer*, *iProduct* und *iSerialNumber* referenzierten Zeichenketten *Manufacturer*, *Product* und *SerialNumber* werden ebenfalls ausgelesen.

(3): Die ausgelesenen Daten werden in der folgenden Reihenfolge ohne Verwendung von Delimitern konkateniert: *idVendor*, *idProduct*, *bcdDevice*, *Manufacturer*, *Product*, *SerialNumber*.

(4): Unter Verwendung des vom Benutzer bereitgestelltem Schlüsselmaterials wird eine abgetrennte digitale Signatur²³ über der aus Schritt (3) resultierenden Zeichenkette errechnet:

(4.1): Wird OpenPGP-Schlüsselmaterial verwendet, ist die Signatur entsprechend RFC 2440 [43, S. 17-32, 52] zu bilden.

(4.2): Wird X.509-Schlüsselmaterial verwendet, ist die Signatur entsprechend RFC 2630 [64], Abschnitt 5: „Signed-data content type“ zu bilden.

(5): Die Signatur wird auf dem Datenträger gespeichert. Da auf USB-Memory-Sticks üblicherweise ein Dateisystem, also eine fixe Vorschrift zur Anordnung von Daten auf dem Medium, existiert, können die zur Authentifizierung und zur Identifizierung des Mediums benötigten Daten in Form von Dateien abgelegt werden. In Abschnitt 6.3 werden hierfür konkrete Vorgehensweisen angegeben.

²³In den referenzierten RFCs wird von „detached signatures“ gesprochen. Abgetrennte Signaturen unterscheiden sich von anderen Signaturtypen dadurch, daß die signierten Daten im Signatur-Paket nicht enthalten sind.

Die Signatur kann überprüft werden, indem die Schritte **(1)** bis **(3)** obigen Verfahrens erneut durchgeführt werden und anschließend die in RFC 2440 [43] bzw. RFC 2630 [64] spezifizierten Verfahren zur Verifikation von Signaturen über den aus Schritte **(3)** resultierenden Daten und der auf dem Datenträger gespeicherten Signatur angewandt werden.

Aufgrund des Größe des verfügbaren Speicherplatzes auf einem USB-Memory-Stick wird bei Verwendung dieser Geräte als Authentifizierungsmedien auch die Speicherung von personenbezogenen Daten und Sekundärschlüsseln auf dem Medium möglich sein. Benutzer werden zusätzlich noch weitere Nutzdaten darauf ablegen können.

Probleme hinsichtlich der gebotenen Sicherheit ergeben sich beim Einsatz von USB-Memory-Sticks unter Umständen durch die Möglichkeit der Kompromittierung von Client-Systemen durch das Einschleusen modifizierter CIDAS-Client Software, die die Verwendung von Medien ohne oder mit falscher Signatur nicht unterbinden. Auch ist es denkbar, daß Angreifer Speichermedien mit dem Medium des Opfers identischen Deskriptoren verwenden oder sich auf andere Weise Zugang zu einem validen Authentifizierungsmedium verschaffen. Dies ermöglicht einem Angreifer jedoch noch nicht den Zugang zu CIDAS, da die Authentifizierung beim Server ausschließlich über das auf dem Stick gespeicherte Schlüsselmaterial entsprechend Abschnitt 4.2.3 durchgeführt wird. Auf diese und weitere Angriffsmöglichkeiten gegen das Authentifizierungsmedium oder den Authentifizierungsprozeß wird im Abschnitt 4.3 eingegangen.

4.2.6 CD-ROMs

Bei „Compact Disc – Read-Only Memories“, kurz CD-ROMs, handelt es sich um optische Speichermedien. Ihr physischer Aufbau wird beispielsweise von Messmer in [80, S. 927-933] erläutert. Laut Stan [101, S. 5] ist die physische und logische Struktur für ihren Einsatz als digitaler Audio-Datenträger seit

1981 standardisiert. In ECMA-119 [3] bzw. ISO 9660 [5]²⁴ wird der Einsatz von CD-ROMs als universell einsetzbare Datenträger spezifiziert. Seitdem werden die Datenträger mit einer Kapazität zwischen 16 MiB für Datenträger im Kreditkartenformat und 900 MiB für 120 mm-Discs vielfach eingesetzt, was nicht zuletzt an den geringen Preisen der Medien liegt. Entsprechende Lese- und Schreibgeräte sind bereits seit mehreren Jahren preiswert verfügbar bzw. in nahezu jeden verkauften Computer eingebaut. CD-ROMs können von Benutzern selbst beschrieben werden. Hierfür sind im Handel sogenannte CD-Rs²⁵ und CD-RWs²⁶ erhältlich. Obige Angaben entstammen [101, S. 1-13] und [80, S. 927-933], CD-ROMs im Kreditkartenformat sind über verschiedene Hersteller zu beziehen.

Seit einigen Jahren werden CD-ROMs zunehmend durch „Digital Versatile Discs“, DVDs, verdrängt (siehe Stan [101, S. 2]). Diese haben eine höhere Speicherkapazität als CD-ROMs, ansonsten jedoch sehr ähnliche Eigenschaften. Auch können DVD-Laufwerke in der Regel CD-ROMs lesen – alle im Folgenden gemachten Angaben gelten gleichwohl für CD-ROMs als auch für DVDs.

Durch die hohen Speicherkapazitäten von CD-ROMs und die allgemeine Verfügbarkeit der Datenträger und Laufwerke eignen sich diese durchaus als Authentifizierungsmedien. Auch können personenbezogene Daten in nahezu unbegrenztem Umfang auf den Datenträgern abgelegt werden. Auf CD-ROMs wird, ebenso wie auf USB-Memory-Sticks üblicherweise ein Dateisystem verwendet. Die von CIDAS benötigten Daten können dementsprechend in Form von Dateien gespeichert werden.

In Bezug auf die Identifizierbarkeit der Datenträger bieten CD-ROMs jedoch Nachteile: Sie verfügen über keinerlei Seriennummern oder vergleichbare Identifizierungsmerkmale. Aus dem ATIP²⁷ können lediglich Informa-

²⁴ISO 9660 lag dem Autor nicht vor.

²⁵CD-Rs sind einmalig beschreibbar, vielfach auslesbar (vgl. Stan [101, S. 8]).

²⁶CD-RWs können mehrmals beschrieben werden (vgl. Stan [101, S. 8]).

²⁷ATIP – „Absolute Time In Pre-groove“, Informationen hierzu sind im „Orange Book“,

tionen bezüglich der Art des Datenträgers sowie dessen Hersteller ermittelt werden, die für eine Identifizierung entsprechend Abschnitt 4.2.1 nicht ausreichen bzw. nicht eindeutig genug sind. Alle weiteren Informationen auf den Medien können beliebig dupliziert werden.

Da Identifizierung des Mediums jedoch nicht zwingende Voraussetzung für den Erfolg der Authentifizierung ist, lassen sich CD-ROMs trotzdem als Authentifizierungsmedien verwenden. Lediglich von einem Einsatz in „unsicheren“ Arbeitsumgebungen, siehe hierzu Abschnitt 4.3.1, wird abgeraten.

4.2.7 Passive Transponder

Transponder, sogenannte „Radio Frequency Identification Device“ mit einer kontaktlosen Schnittstelle entsprechend ISO 14443 [15] bzw. ISO 15693 [16] eignen sich insbesondere dann zur Identifizierung und Authentifizierung von Personen oder Gegenständen, wenn eine möglichst schnelle, automatische Aufnahme und Verarbeitung der Identifizierungs- und Authentifizierungsdaten erforderlich ist: Die Geräte sind sehr klein, die Größen liegen zwischen einigen Quadratzentimetern bis hin zu wenigen Quadratmillimetern und können auch in extremen Arbeitsumgebungen, beispielsweise bei großer Hitze, eingesetzt werden. Darüber hinaus sind sie preiswert und verfügen über einen für viele Anwendungen ausreichend großen²⁸ EEPROM-Speicher^(Glossar), der innerhalb weniger Sekunden²⁹ auch auf einige Meter Entfernung und ohne Sichtkontakt fehlerfrei ausgelesen oder beschrieben werden kann (vgl. *Escort Memory Systems* [30]). Ein Nachteil gegenüber den anderen hier aufgeführten Medien besteht darin, daß zum Auslesen der Transponder spezielle Hardware

Abschnitt 2 zu finden. Das Buch ist jedoch nur gegen eine hohe Schutzgebühr von Philips Electronics erhältlich. Das Wissen des Autors beruht auf dem Studium des Quellcodes einer Open-Source-Software namens *cdrecord*, verfügbar unter <http://www.fokus.fhg.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html>.

²⁸In einem White Paper der Firma *Escort Memory Systems* [30] werden maximal 736 Bytes für passive Geräte angegeben, die *Infineon Technologies AG* bietet bereits RFIDs mit 1.25 KiB (siehe [22]) an.

²⁹Die Lesegeschwindigkeiten liegen etwas unterhalb von 30 Kib/s, ein 10 Kib großer Speicher kann damit unter optimalen Bedingungen in etwa 1/3 s vollständig ausgelesen werden.

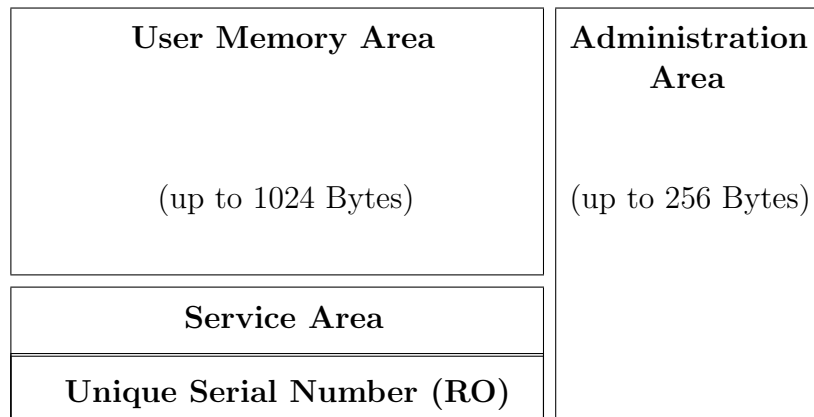


Abbildung 4.2: Üblicher Speicheraufbau in RFIDs

nötig ist. Passive Transponder verfügen über keine eigene Stromversorgung, sie werden kontaktlos durch elektromagnetische Induktion über eine Antenne vom Lesegerät mit Energie versorgt.

Abbildung 4.2 auf Seite 72 ist einer Abbildung aus einer Produktbeschreibung der *Infineon Technologies AG* [22, S. 6] entlehnt und zeigt eine häufig verwendete Speicherstruktur für RFID-Geräte. Der Speicher ist üblicherweise in drei Bereiche unterteilt, eine *User Memory Area* zur Speicherung von Nutzdaten, eine *Administration Area*, häufig für Metadaten verwendet, und eine *Service Area*, in der unter anderem die Seriennummer des Gerätes bzw. des Speicherbausteins herstellerseitig gespeichert³⁰ wird. Zur Identifizierung des Datenträgers, also zur Bildung einer Signatur über fixen Daten des Mediums, bietet sich diese Seriennummer an. Es wird ein ähnlicher Algorithmus wie in Abschnitt 4.2.5 verwendet:

- (1): Die *Unique Serial Number* wird aus dem Gerät ausgelesen.
- (2): Unter Verwendung des vom Benutzer bereitgestellten Schlüsselmaterials wird eine abgetrennte digitale Signatur über der aus Schritt (1) resultierenden Zeichenkette errechnet:

³⁰ „RO“ steht für „Read Only“, die Seriennummer kann nur ausgelesen, nicht modifiziert werden.

(2.1): Wird OpenPGP-Schlüsselmaterial verwendet, ist die Signatur entsprechend RFC 2440 [43, S. 17-32, 52] zu bilden.

(2.2): Wird X.509-Schlüsselmaterial verwendet, ist die Signatur entsprechend RFC 2630 [64], Abschnitt 5: „Signed-data content type“ zu bilden.

(3): Die Signatur wird auf dem Datenträger gespeichert. Die Datenspeicherung auf RFIDs erfolgt in der Regel ohne das Vorhandensein eines Dateisystems. Im Abschnitt 6.3 werden daher gesonderte Vorschläge zur Strukturierung der notwendigerweise zu speichernden Daten gemacht.

Problematisch bei der Verwendung von RFIDs ist der geringe insgesamt verfügbare Speicherplatz. Aus diesem Grund sollten hier vorrangig OpenPGP-Schlüssel und -Signaturen eingesetzt werden. Auch wird es nicht möglich sein, Schlüsselmaterial mit mehr als 1024 Bit Länge zu verwenden. Damit bieten sich RFIDs vorrangig als kurzzeitig eingesetzte Authentifizierungsgeräte an.

Es ist davon auszugehen, daß auf den Datenträgern nicht ausreichend Platz zur Speicherung personenbezogener Daten verfügbar sein wird. Sollten in der Zukunft Geräte mit 2 oder mehr KiB verfügbar sein, erweitern sich die Möglichkeiten entsprechend.

Auch wenn die RFID-Technologie in Abhängigkeit von den Fähigkeiten der verwendeten Lesegeräte die Möglichkeit bietet, mehrere Datenträger gleichzeitig auszulesen (vgl. *Escort Memory Systems* [30]), wird sich, sollen viele Benutzer in möglichst kurzer Zeit authentifiziert werden, das Problem ergeben, daß die Passworteingabe³¹ eine relativ lange Zeit in Anspruch nimmt. Möglicherweise sind RFIDs wesentlich besser als Autorisierungs-

³¹Siehe hierzu Abschnitt 4.1.5.

Tickets, im Güterverkehr oder zur Fluggastabfertigung, als als Authentifizierungsmedien geeignet.

Es sei darauf hingewiesen, daß verschiedene Hersteller von RFID-Technologie mit Produkten wie beispielsweise dem von der *Infineon Technologies AG* produzierten und in [23] beschriebenen Gerät, durch die Integration von aus Chipkarten bekannter Sicherheitslogik sehr einfach anwendbare Authentifizierungsgeräte anbieten.

4.3 Angriffsmöglichkeiten und zusätzliche Sicherungsmechanismen

Bei der Durchführung einer Authentifizierung mittels asymmetrischer Kryptographie und mobiler Datenträger ergeben sich insbesondere auf nicht vertrauenswürdigen Client-Systemen, wie bereits in den vorangestellten Abschnitten erläutert, Probleme bei der Geheimhaltung des privaten Schlüssels der Benutzer. Da die Sicherheit der Authentifizierung, der personenbezogenen Daten der Benutzer und damit die Sicherheit des Gesamtsystems primär davon abhängen, daß dieses Schlüsselmaterial nicht kompromittiert ist, sind zusätzliche Sicherungsmechanismen zwingend erforderlich.

Auch führt, wie in Abschnitt 4.2.3 angedeutet, die Angabe inkorrekt identifizierender Daten zu einem ungewollten Systemverhalten: Da zu einem nicht vorhandenen Benutzerdatensatz auch keine Authentifizierungsdaten vorhanden sind, müßte der Authentifizierungsserver den Authentifizierungsvorgang abbrechen, was dem von Schäfer in [93, S. 92] spezifizierten Verhalten widerspricht.

Mehrere Ansätze zur Einführung zusätzlicher Sicherungsebenen sollen in diesem Abschnitt aufgezeigt und diskutiert werden.

Auf Angriffe gegen die verwendeten Verschlüsselungsverfahren selbst wird in dieser Arbeit nicht eingegangen. Entsprechende Hinweise sind beispielsweise bei Schneier ([96]) oder in den jeweiligen Verfahrensbeschreibungen, beispielsweise bei El Gamal ([53]) und in weiterführenden Analysen der Verfahren.

4.3.1 Angriffe und Angriffsszenarien

Unter einem Angriff wird jeder absichtliche manuelle oder automatisiert durchgeführte Versuch eines Benutzers (des Angreifers) verstanden, Identifizierungs- bzw. Authentifizierungsdaten eines Dritten zu kompromittieren

oder sich als dieser auszugeben. Um sich erfolgreich zu legitimieren, muß ein Benutzer im hier vorgestellten Authentifizierungssystem zum einen über Identifizierungsdaten, zum anderen über seinen öffentlichen und den zugehörigen privaten Schlüssel verfügen. Will sich ein Angreifer als derselbe Benutzer ausgeben, muß er über die gleichen Daten verfügen. Da selbst bei bekannten Identifizierungsdaten das Brechen der Verschlüsselung bei Verwendung der in OpenPGP oder X.509 üblichen Verfahren extrem zeitaufwendig ist, kann das System ohne entsprechende Daten höchstens aufgrund von Fehlern in einer Implementierung kompromittiert werden. Dennoch sollten auch Identifizierungsdaten geeignet geschützt werden, insbesondere, weil diese unter Umständen die Identifizierung der zugehörigen Person direkt ermöglichen³².

Ohne zusätzliche Sicherungsmechanismen haben Angreifer durch simples Durchprobieren von Identifizierungsdaten die Möglichkeit, Teile des Benutzerbestand eines CIDAS-Servers zu identifizieren: Für korrekte Identifizierungsdaten wird der Server eine Authentifizierung anbieten, andernfalls wird er dazu nicht in der Lage sein, weil er über kein zu den eingegebenen Daten passendes öffentliches Schlüsselmaterial verfügt.

Darüber hinaus sind Angriffe denkbar, bei denen mittels modifizierter Clientsoftware oder -Hardware versucht wird, zusammengehörige Authentifizierungs- und Identifizierungsdaten zu erhalten. Für Benutzer, die aus einer gesicherten Umgebung heraus, beispielsweise ihrer Wohnung, CIDAS benutzen, stellen derartige Angriffe kein sehr großes Risiko dar. Dies liegt insbesondere daran, daß der Benutzer hier die Möglichkeit hat, beglaubigt korrekte Clientsoftware zu installieren. Auch wird er nachträgliche Manipulationen an seiner Hard- und Softwarekonfiguration in der Regel bemerken.

³²Es sei nochmals darauf hingewiesen, daß CIDAS neben verhältnismäßig anonymen Identifizierungsdaten wie einer E-Mail-Adresse beispielsweise auch eine Postanschrift zur Benutzeridentifizierung verwenden kann.

Viel problematischer ist die Situation für Benutzer, die regelmäßig ihren Arbeitsplatz wechseln oder CIDAS von absolut nicht vertrauenswürdigen Rechnern, z.B. in Internet-Cafés oder Internet-Terminals an Bahnhöfen, aus nutzen. Benutzer haben hier aus Sicht des Autors keine Möglichkeit, das Kopieren von auf Authentifizierungsmedien gehaltenen Daten oder das Mithören der Tastatureingaben zu verhindern oder auch nur zu bemerken. Verfügt ein Angreifer über diese Daten, kann er das auf dem Authentifizierungsmedium enthaltene Schlüsselmaterial entschlüsseln und sich vor dem CIDAS-Server als der Benutzer ausgeben, dem die Daten ursprünglich gehörten. Diese Angriffsstrategie ist verhältnismäßig alt und funktioniert bei den meisten heute im Internet verwendeten Authentifizierungsmechanismen: Diverse Webseiten-Logins, Online-Banking-Portale und Zugänge zu Firmennetzen sind kompromittierbar, sobald die Benutzer von nicht vertrauenswürdigen Systemen aus arbeiten. Da derartige Angriffe mit einfachsten Mitteln³³ realisierbar sind, besteht akuter Bedarf an Lösungen für dieses Problem.

4.3.2 Zusätzliche Sicherheitsmechanismen

Zusätzliche Sicherheitsmechanismen zum in dieser Arbeit beschriebenen Authentifizierungsverfahren sollten es zum einen ermöglichen, ein vom Benutzer als sicher eingestuftes Client-System als solches zu identifizieren, also eine Möglichkeit zur Herstellung einer Beziehung zwischen einer Client-Installation und einem bzw. einer Reihe von Benutzern, die diese Installation als sicher erachten, bieten. Da es aus Sicht des Autors keine Möglichkeiten gibt, das Mithören von Tastatureingaben bzw. das Kompromittieren von Schlüsselmaterial bei der Verwendung von nicht vertrauenswürdiger Client-Installationen auszuschließen, sollen im Weiteren Maßnahmen zur Reduzierung des im Falle eines erfolgreichen Angriffs entstehenden Schadens vorge-

³³Das Implementieren einer Software, die alle Eingaben des Benutzers abfängt und speichert, erfordert nach Einschätzung des Autors nur wenige Stunden Entwicklungsarbeit. Noch weitaus einfacher ist das automatische Kopieren von Datenträgern realisierbar. Eine Hardwarelösung für beides kann ebenfalls mit vergleichsweise geringem Aufwand umgesetzt werden (vgl. Kersten, [70, S. 62]).

stellt werden. Nicht zuletzt sind Mechanismen zum Schutz der Identifizierungsdaten vorzusehen.

4.3.3 Identifizierung von Client-Installationen

Verhältnismäßig einfach gestaltet sich die Identifizierung von Client-Installationen: Entsprechend der Spezifikation des CIDAS-Protokolls in Kapitel 5 werden zur Verbindungssicherung und Authentifizierung des Servers die kryptographischen Protokolle SSL/TLS³⁴ genutzt. Beide erlauben eine beidseitige Authentifizierung der Kommunikationspartner, also sowohl des Authentifizierungsservers als auch des -clients. Ausgehend davon, daß im Server neben den primären Authentifizierungsdaten noch eine Liste von Zertifikats-Fingerprints³⁵ vertrauenswürdiger Client-Installationen für jeden Benutzer gespeichert werden kann, kann ein Benutzer sein Vertrauen auf die Sicherheit einer Installation durch einfaches Aufnehmen des betreffenden Fingerprints in diese Liste definieren.

Offensichtlich ist, daß bei diesem Verfahren für jeden CIDAS-Client ein Zertifikat existieren muß. Dieses sollte während des Installationsprozesses erzeugt und signiert³⁶ werden. Effektiv gibt es an dieser Stelle hinsichtlich der erreichten Sicherheit keine Unterschiede zwischen der Verwendung von selbst- oder von einer CA³⁷ signierten Zertifikaten. Dies liegt daran, daß die

³⁴SSL steht für „Secure Socket Layer“, TLS bedeutet „Transport Layer Security“. Beide Protokolle implementieren kryptographische Verfahren zur Absicherung von Vertraulichkeit und Authentizität von über unsichere Netze transferierten Daten. Die Protokolle sind von der *Netscape Inc.* in [55] bzw. in RFC 2246 [50] spezifiziert.

³⁵SSL und TLS benutzen zur Authentifizierung und zur Verbindungssicherung ebenfalls asymmetrisches Schlüsselmaterial. Dieses wird ebenso wie das möglicherweise zur Benutzerauthentifizierung verwendete Schlüsselmaterial in Zertifikaten entsprechend X.509 [44] gespeichert. Zertifikate können über eine eindeutige Nummer, den „Fingerprint“ identifiziert werden.

³⁶Zertifikate werden erst durch eine digitale Unterschrift benutzbar. Die Unterschrift kann mit dem selben (self-sign) oder einem anderen Zertifikat erstellt werden. Durch sie garantiert der Unterschreiber für die Korrektheit der im Zertifikat enthaltenen Informationen über seinen Besitzer. Weiterführende Informationen findet der interessierte Leser in X.509 [44].

³⁷CA steht für „Certificate Authority“ und bezeichnet eine vertrauenswürdige Instanz, die durch ihre Unterschrift für die Identität von Zertifikatsinhabern bürgt

von SSL/TLS gebotene Funktionalität hier lediglich zur Identifizierung einer Client-Installation über den Fingerprint des jeweils erzeugten Zertifikates und nicht zur Authentifizierung verwendet wird. Tatsächlich böte die clientseitige Bildung einer genügend langen Zufallszahl anstelle des Zertifikates, die dann zwar nicht bereits im SSL/TLS-Handshake, sondern erst über das CIDAS-Protokoll ausgetauscht werden könnte, immer noch die gleichen Möglichkeiten und die gleiche Sicherheit. Die Verwendung von Zertifikaten bietet jedoch insbesondere für den kommerziellen Einsatz von CIDAS Vorteile: Beispielsweise könnten CIDAS-Server derart konfigurierbar sein, daß sie nur Clients mit einem von einer zum jeweiligen Server gehörenden CA signierten Zertifikat akzeptieren oder diesen Clients besondere, sonst nicht zugängliche Funktionalitäten anbieten. Weiterhin spricht für den Aufbau einer CIDAS-CA, daß über diese auch alternatives Schlüsselmaterial³⁸ der registrierten Benutzer, beispielsweise für die Verschlüsselung von E-Mails, signiert werden könnte womit CIDAS-Server in der Lage wären, TrustCenter-Funktionalität anzubieten.

Der private Schlüssel zu einem Zertifikat wird aus Sicherheitsgründen üblicherweise durch symmetrische Kryptographie geschützt abgespeichert. Für obiges Verfahren hätte dies zur Folge, daß Benutzer im Laufe der CIDAS-Sitzung mehrere Passworte eingeben müßten. Diese Passworte wäre zum einen das Passwort zum Entschlüsseln des Client-Zertifikates, zum anderen das zum Entschlüsseln des für zur Benutzerauthentifizierung verwendeten Schlüsselmaterials nötige Passwort. Zwar würde hierdurch die Sicherheit des Verfahrens erhöht werden, jedoch würde es Benutzer unter Umständen überfordern: Wie Whitten et al. in [109] zeigt, haben Benutzer in der Regel bereits Probleme, auf asymmetrischer Kryptographie Basierende Sicherheitssysteme mit einem Schlüsselpaar zu verstehen. Da die Nutzung clientseitiger Zertifikate jedoch primär für vom Benutzer als sicher klassifizierte Client-Systemen gedacht ist, kann hier möglicherweise auf die Verschlüsselung der Zertifikate

³⁸Die Verwendung von ein und demselben Schlüsselmaterial für verschiedene Aufgaben gilt als schlechte kryptographische Praxis, vgl. Schneier [96, S. 186].

verzichtet werden. Letztlich ist es der Entscheidung des Benutzers überlassen, der Autor empfiehlt die Verwendung verschlüsselter Zertifikate.

4.3.4 Signierte Client-Software

Eine weitere Möglichkeit, die Korrektheit einer Client-Software verifizierbar zu machen, bestuende darin, diese zu signieren. Es müßte also eine digitale Signatur über dem Client und allen zugehörigen Bibliotheken mit einem Vertrauenswürdigen Schlüssel, beispielsweise dem des CIDAS-Servers, erstellt werden. Ein Benutzer könnte diese Signatur vor der Benutzung der Client-Software überprüfen. Hierbei ergeben sich die folgenden Probleme: Zum einen muß der Benutzer die Überprüfung manuell vornehmen oder entsprechende Software für alle unterstützten Hardware-Plattformen Betriebssysteme auf seinem Authentifizierungsmedium mitbringen – er hat schließlich keinen Grund auf die Korrektheit irgendeiner auf dem System installierter Software zu vertrauen. Zum anderen soll die CIDAS-Client-Software auch im Quellcode angeboten werden, so daß Anwender sich diesen selbst selbst übersetzen können. Hierbei werden die Kompilate in Abhängigkeit von verwendeten Compiler und der Hard- und Software-Umgebung stark variieren. Auch wird die tatsächliche Korrektheit eines Kompilates nur unter sehr großem Aufwand nachprüfbar sein. Durch einen Anwender übersetzte CIDAS-Clients können aus diesem Grund nicht über kryptographische Prüfsummen und Signaturen geschützt werden. Auch schützt ein signierter CIDAS-Client nicht vor anderweitigen Manipulationen am verwendeten Betriebssystem oder der Hardware. Daraus schlußfolgernd stellt der Autor fest, daß signierte Client-Software kein wirkliches Mehr an Sicherheit bieten kann.

4.3.5 Maßnahmen zur Schadensreduzierung

Insbesondere zur Vermeidung des Problems der Kompromittierbarkeit der verwendeten Authentifizierungsdaten bei Public-Key-Authentifizierungsverfahren, bieten Sicherheitsspezialisten wie *RSA Security Inc.* mit verschiedenen Produkten zur Realisierung einer Zwei-Komponenten-Authentifizierung, bei der ein Benutzer neben seinen Identifizierungsdaten zum einen ein

Passwort, zum anderen ein zeitabhängiges Einmalpasswort eingeben muß, eine gangbare und hochsichere Alternative³⁹ zur Verwendung von asymmetrischem Schlüsselmaterial an. Jedoch sind derartige Mechanismen derzeit noch sehr kostenaufwendig – es wird hierbei für jeden Benutzer ein teures⁴⁰ Hardware-Token mit integrierter Sicherheitslogik benötigt – und werden aus diesem Grund für Web-Applikationen nicht oder nur sehr selten eingesetzt. Eine Einbindung solcher Verfahren in CIDAS ist durchaus möglich und würde die gebotene Sicherheit um ein Vielfaches erhöhen. Insbesondere weil die Verfahren die hier besprochenen Public-Key-Authentifizierungsverfahren vollständig ersetzen könnten, eignen sie sich nicht als *zusätzliche* Sicherheitsmaßnahme und sind nicht Gegenstand dieser Arbeit. Es sei dennoch darauf hingewiesen, daß sicherheitskritische Anwendungen und Daten auch durch angemessene Sicherheitstechnik geschützt werden sollten. Die Entscheidung über die im Einzelnen zu verwendenden Verfahren obliegt dem Benutzer bzw. dem Anwendungsbetreiber.

Vor allem im Bereich des Home-Banking wird eine andere Art von Einmalpassworten seit langem unter der Bezeichnung TAN⁴¹ eingesetzt. Hierbei wird für jede Transaktion, z.B. eine Überweisung, eine neue TAN verwendet, die anschließend ihre Gültigkeit verliert (siehe Fuhrberg, [56, S. 319-326]). Die hierfür nötigen TANs müssen dem Kunden im Voraus mitgeteilt werden. Dies geschieht üblicherweise persönlich oder auf dem Postweg. Diese Art der Einmalpassworte kann im Zusammenhang mit dem in dieser Arbeit vorgestellten Authentifizierungsverfahren durchaus eingesetzt werden.

Eine weitere Möglichkeit zur Erhöhung der gebotenen Sicherheit stellt die Verwendung von Sekundärschlüsseln dar. Entsprechend Schäfer [93, S.

³⁹Obige Angaben beziehen sich auf das Produkt „RSA SecurID“, bei dem Benutzer neben ihrem Passwort einen „Passwort-Generator“ im Chipkartenformat besitzen, der für jede Minute ein neues, auch nur für diese Minute gültiges Passwort anzeigt. Weitere Informationen sind über <http://www.rsasecurity.com> zu beziehen.

⁴⁰Eine persönliche Anfrage bei der *RSA Security Inc.* ergab Stückpreise zwischen 50 und 100 €, Die Geräte können ausschließlich zur Authentifizierung, nicht auch zur Datenspeicherung verwendet werden.

⁴¹TAN steht für „Transaktionsnummer“.

100-102] können Benutzer nach erfolgreicher Anmeldung ihre eigenen personenbezogenen Daten und natürlich auch die Authentifizierungsdaten ändern. Hier kann ihnen auch die Möglichkeit geboten werden, sekundäres Schlüsselmaterial und diesbezügliche Sicherheitsanpassungen vorzunehmen. Sekundäres Schlüsselmaterial ist ebenso aufgebaut wie das Primäre, hat jedoch eine zeitlich eng begrenzte Validitätsdauer im Bereich von Tagen oder Wochen. Darüber hinaus sollten nach der Anmeldung mit diesem Schlüsselmaterial bestimmte Funktionen von CIDAS oder auch externe Applikationen nicht zugänglich sein. Sinnvoll erscheint es an dieser Stelle, vor allem die Einsicht und Modifikation in personenbezogene Daten sowie den Zugriff auf hochgradig sicherheitskritische Anwendungen einzuschränken. Wird zum Schutz des sekundären Schlüsselmaterials ein anderes Passwort als für das primäre Schlüsselmaterial verwendet, kann ein Benutzer dieses zum Login von einem nicht vertrauenswürdigen Arbeitsplatz aus benutzen, ohne sein primäres Schlüsselmaterial der Gefahr der Kompromittierung auszusetzen.

Das System ist ganz offensichtlich ausschließlich zur Schadensbegrenzung geeignet: Es wird prinzipiell das gleiche Verfahren wie für eine „normale“, auf asymmetrischer Kryptographie basierende Authentifizierung verwendet. Damit sind alle unter 4.3.1 aufgeführten Angriffe, natürlich nur in Bezug auf das gerade verwendete Schlüsselmaterial, möglich. Die Kompromittierung des sekundären Schlüsselmaterials erlaubt jedoch nur den sowohl zeitlich als auch funktional eingeschränkten Zugriff auf die vom jeweiligen Benutzer verwendeten Systeme. Unter der Bedingung, daß die nach einer Authentifizierung mit sekundärem Schlüsselmaterial noch zugreifbare Funktionalität vom Benutzer aktiv beeinflusst werden kann, bietet das Verfahren in Bezug auf das verbleibende Restrisiko eine maximal skalierbare Lösung. Auch ist sie für den Benutzer verhältnismäßig komfortabel, weil dieser das ihm vertraute Verfahren weiter verwenden kann und er keine Listen von Einmalpassworten mit sich herumtragen muß. Insbesondere bietet sich an dieser Stelle auch die Möglichkeit an, Schlüsselmaterial für einen bestimmten Zeitraum, beispielsweise wöchentlich wechselnd, im Voraus anzulegen, was selbst längere Aufenthalte ohne Zugang zu vertrauenswürdigen Arbeitsplätzen planbar macht.

4.3.6 Umgang mit inkorrekten Identifizierungsdaten

Nicht unproblematisch ist der Umgang mit falschen oder unvollständigen Identifizierungsdaten. Das weiter oben (S. 75) erläuterte Problem läßt sich aus Sicht des Autors wie folgt lösen: Der Authentifizierungsserver fragt vom Benutzer zusätzlich den Fingerprint des zu verwendenden Schlüsselmaterials ab⁴². Verfügt der Server über einen entsprechenden öffentlichen Schlüssel und ist dieser für die verwendeten Identifizierungsdaten als Authentifizierungsdatum zulässig, läuft der Authentifizierungsprozeß entsprechend Abschnitt 4.2.3 ab. Trifft dies nicht zu, muß der Server für Nachricht zwei ($V \rightarrow P: \{M_V\}_{K_P}$) eine äußerlich korrekt mit dem durch den angegebenen Fingerprint bezeichneten öffentlichen Schlüssel verschlüsselte Pseudo-Nachricht an den Client senden. Aufgrund des fehlenden Schlüsselmaterials kann die Nachricht selbst jedoch nur eine Menge zufälliger Daten plausibler Größe enthalten. Die Datenabfrage kann im Anschluß an die Übermittlung dieser Nachricht aus Sicht des Servers weiter fortgesetzt werden, eine Überprüfung der empfangenen Daten erübrigt sich.

Ein Angreifer sollte bei diesem Verfahren keine Möglichkeit mehr haben, Identifizierungsdaten auf ihre Validität prüfen zu lassen. Er wird vom Server lediglich nichtssagende Antworten bekommen.

⁴²Der Benutzer muß diesen natürlich nicht auswendig wissen, Client-Implementierungen können komfortable Schnittstellen zur Auswahl des zu verwendenden Schlüsselmaterials bereitstellen.

Kapitel 5

Konzeption eines Kommunikationsprotokolls für CIDAS

[...] it is chiefly unexpected occurrences which require instant consideration and help all such matters defy communication by fire signal. For it is quite impossible to have a preconcerted code for things which there is no means of foretelling.
— Polybius, *The Histories*, ([84, Book X, Chapter 43])

Konzeption eines Kommunikationsprotokolls für CIDAS

Wie Holzmann in [62] richtig feststellt, ist es in der Regel wesentlich schwieriger, ein gutes und fehlerfreies Kommunikationsprotokoll zu entwickeln, als ein normales sequentielles Programm zu schreiben. Kommunikationsprotokolle wurden, so Holzmann in [63], in den vergangenen Jahren größtenteils von einigen wenigen Spezialisten entworfen, ohne daß dafür grundsätzliche Techniken oder Vorgehensweisen verwendet wurden. Als Alternative zu dieser Entwurfsvariante wird beispielsweise in RFC 822 ([47]) und RFC 2616 ([54]) die Backus-Naur Form entsprechend Backus et al. ([35]) zur Darstellung der Nachrichten und der dem Protokoll zugrundeliegenden Regeln verwendet. Auch Holzmann definiert in [62] und [63] eine Beschreibungssprache für Protokolle, über die der Entwickler mittels einer Analyse-Software die formale Vollständigkeit und Widerspruchsfreiheit eines Protokolls prüfen kann.

Das im folgenden dargestellte Protokoll entstand ohne Zuhilfenahme derartiger Verfahren. Es dient lediglich als Arbeitsgrundlage für erste prototypische Umsetzungen von CIDAS und stellt dementsprechend die gesamte benötigte Funktionalität bereit, Ausnahmen bilden hierbei eine Reihe bislang un spezifizierter Authentifizierungsverfahren sowie die Verwendung des von Schäfer in [93, S. 71] aufgeführten, jedoch nicht weiter erläuterten „Transaktionssystems“ von CIDAS. Die folgende Protokollspezifikation ist dementsprechend vorrangig prosaischer Natur. Der Autor orientierte sich während der Arbeiten am CIDAS-Protokoll vorrangig an den RFCs 2408 und 2409 ([77], [60]), an deren Implementierung er im Rahmen früherer Tätigkeiten beteiligt war. Auch wurden verschiedene Hinweise aus Holzmann ([62], [63]) und Bradley et al. ([40]) aufgegriffen. Ein beispielhafter Ablauf einer CIDAS-Sitzung wird im Anhang A anhand der übertragenen CIDAS-Nachrichten angegeben.

Bemerkt sei, daß der aus Sicht eines Benutzers möglicherweise kryptische Aufbau des Kommunikationsprotokolls für diesen nicht von wesentlicher Relevanz ist. Ein Benutzer wird die vom CIDAS-Server bereitgestell-

te Funktionalität stets über die von einer Cidas-Client-Software angebotene Benutzungsoberfläche nutzen. In Kapitel 6 wird der Autor kurz auf die in der prototypischen Umsetzung verwendete Benutzungsschnittstelle eingehen. Insgesamt betrachtet der Autor die Konzeption einer ergonomischen und aus gestalterischer Sicht ansprechenden Benutzungsoberfläche nicht als Gegenstand dieser Arbeit.

5.1 Designentscheidungen zum Protokoll-entwurf

Viele Teile des in diesem Kapitel dargestellten Protokolls basieren auf individuellen Entscheidungen des Autors, die potentiell auch anders hätten ausfallen können. In diesem Abschnitt sollen einige Designentscheidungen erläutert werden.

Die **Bezeichnerwahl** in diesem Teil der Arbeit ist sehr an die vom Autor bevorzugte Programmiersprache C angepasst. Dies sollte jedoch keine Einschränkung hinsichtlich der Implementierbarkeit des Verfahrens in anderen Sprachen nach sich ziehen – die Identifizierung einer Nachricht wird letztlich über einen numerischen, dem jeweiligen Bezeichner zugeordneten Wert realisiert. Alle Bezeichner beginnen mit der Zeichenkette `CIDAS_`. Dadurch lassen sich Namenskollisionen mit möglicherweise gleichnamigen Nachrichten anderer Protokolle vermeiden.

Wie in den Abschnitten 5.2.8 und 5.2.9 dargestellt, basiert das vorgestellte Kommunikationsprotokoll auf der Übertragung einzelner Nachrichten. Diese **Nachrichtenorientiertheit** ist bei der Verwendung verbindungsorientierter Transportprotokolle wie TCP (vgl. RFC793, [2]) nicht zwingend erforderlich. Der Autor entschied sich trotzdem für einen nachrichtenorientierten Protokollaufbau, insbesondere, weil er diesen als „übersichtlicher“ empfindet und bereits Erfahrungen mit der Implementierung derartiger Protokolle besitzt.

Auch erlaubt dieses Vorgehen im Bedarfsfall relativ einfach den Wechsel auf verbindungslose Transportprotokolle.

Auch der **Nachrichtenaufbau** stellt eine weitestgehend individuelle Entscheidung des Autors dar. Die in Abschnitt 5.2.8 beschriebene Vorgehensweise erlaubt eine sehr schnelle Identifizierung einer Nachricht seitens eines Kommunikationspartners. Auch können aufeinander folgende bzw. aufeinander aufbauende Nachrichten verhältnismäßig einfach als solche identifiziert und entsprechend behandelt werden. Der Nachrichtenkopf ist darüber hinaus auch sehr kompakt, so daß die minimale Länge einer Nachricht lediglich 16 Bytes beträgt. Die Begrenzung des im Rahmen einer Nachricht übertragbaren Payloads auf 16 kiB sollte für alle bislang spezifizierten Anwendungsfälle ausreichend sein; durch das in Abschnitt 5.2.8 beschriebene Zusammenfügen der Payloads mehrerer Nachrichten können letztlich beliebig grosse Datenmengen übertragen werden. Mit dem Verzicht auf Datenkomprimierung bei Nachrichten vom Client zum Server wird versucht, Probleme mit sogenannten „Kompressionsbomben“ (vgl. auch Heise Zeitschriften Verlag, [29]) zu umgehen.

Um Payload-Inhalte strukturiert und menschenlesbar zu halten, entschied sich der Autor für die **Verwendung von XML** entsprechend der XML-Spezifikation des *World Wide Web Consortium* ([14]). Die Definition neuer Datenformate für den Einsatz in CIDAS wäre zwar möglicherweise performanter, für Außenstehende möglicherweise jedoch maximal unverständlich gewesen und hätte damit spätere Analysearbeiten am Protokoll behindert.

Insgesamt stellt die formale Spezifikation des CIDAS-Protokolls eine Weiterführung der von Schäfer in [93, S. 90-96] dargelegten Grundgedanken zur Kommunikation zwischen CIDAS-Server und den Clients dar.

5.2 Formale Spezifikation des CIDAS-Protokolls

Der folgende Abschnitt stellt einen Entwurf für das von CIDAS zur Kommunikation zwischen dem Authentifizierungsserver und den Clients genutzte Protokoll in der Version 1.0 dar.

Grundsätzlich setzt das CIDAS-Protokoll auf eine bestehende TCP- (vgl. RFC 793, [2]) bzw. SSL/TLS-Verbindung¹ auf. Aus diesem Grund müssen Handshakes oder Datenverschlüsselung nicht explizit behandelt werden. Sie werden lediglich im Abschnitt 5.2.2 kurz angerissen und im Weiteren als gegeben vorausgesetzt.

Genau spezifiziert werden die gebotenen Möglichkeiten zur Identifizierung, Authentifizierung² und zur weiteren Nutzung der von CIDAS gebotenen Funktionalität. Insbesondere werden der Aufbau der CIDAS-Nachrichten und die Interpretation ihrer Bestandteile erläutert.

5.2.1 Datenrepräsentation

Die grundlegende Datenblock-Größe in diesem Abschnitt ist das Byte³, bestehend aus 8 Bit. Das höchstwertige Bit ist hierbei immer das erste. Multibyte-Daten werden durch die Konkatenation mehrerer Bytes, im Text mit **Bezeichner0**, **Bezeichner1**, **Bezeichner2**,... bezeichnet, gebildet. CIDAS verwendet für die Repräsentation von multibyte-Daten ausschließlich das *Big-Endian*-Format, auch unter dem Begriff *Network-Byte-Order* bekannt. Das bedeutet, daß das signifikanteste Byte eines multibyte-Datums das erste Byte ist. Die Konkatenation zur Bildung eines multibyte-Datums erfolgt daher mit

¹SSL und TLS sind Protokolle, die die Sicherheit von über unsichere Kommunikationswege übertragene Daten durch kryptographische Verfahren gewährleisten. Die Protokolle basieren auf TCP und sind im Internet-Draft 302 ([55]) bzw. in RFC 2246 ([50]) spezifiziert.

²Authentifizierungsverfahren sind nur soweit beschrieben, wie die notwendigen Details hierfür bereits bekannt waren.

³Der Autor verwendet gelegentlich das Wort *Oktett* als Synonym zu *Byte*.

aufsteigender Bezeichnernummer von links nach rechts bzw. von oben nach unten. In C-Notation ließe sich dies wie folgt darstellen:

```
value = (byte[0] << 8*(n-1)) | (byte[1] << 8*(n-2)) | ...  
| byte[n-1];
```

Die für textuelle Informationen zu verwendende Kodierung ist Unicode entsprechend ISO 10646 ([17]) im UTF8-Format nach RFC 2279 ([114]). XML-Strukturen beinhalten im *XML-Prolog*⁴ immer einen Hinweis auf die verwendete Kodierung.

5.2.2 Verbindungsaufbau

Der Aufbau einer Verbindung zwischen einem CIDAS-Client und dem Authentifikationsserver geschieht mittels SSL oder TLS, wie sie im Internet-Draft „The SSL Protocol Version 3.0“ bzw. in RFC 2246, „The TLS Protocol Version 1.0“⁵ beschrieben sind: Der Client eröffnet die Verbindung zum Server, Client und Server einigen sich bezüglich der zu verwendenden Protokollversion und der kryptographischen Algorithmen, authentifizieren sich gegenseitig und nutzen Public-Key Kryptographie zur Erzeugung von Schlüsselmaterial für den weiteren Verlauf der Sitzung. Danach wird die vorhandene und gesicherte Verbindung zum Austausch von Anwendungsdaten eines darüber liegenden Protokolls, in diesem Fall des CIDAS-Protokolls, genutzt. Sollte der Server aus irgendeinem Grund nicht in der Lage sein, eine Verbindung anzunehmen oder weiter zu bearbeiten, kann er sie zu einem beliebigen Zeitpunkt abbrechen. Dem Client steht das gleiche Verhalten zu. Client und Server prüfen auf Zeitüberschreitungen, um den formalen Status einer Verbindung einzuschätzen. Der durch SSL/TLS geschützte CIDAS-Datenverkehr, im Folgenden mit „CIDAS-Encrypted“-Dienst bezeichnet, wird serverseitig auf Port 755/TCP entgegengenommen.

Neben der durch SSL/TLS geschützten Verbindung können Serverimplementierungen auch einen unverschlüsselten Verbindungsaufbau anbieten. Die

⁴Siehe hierzu die XML-1.0-Spezifikation ([14]).

⁵Siehe hierzu Netscape Inc., [55] und RFC 2246 [50].

Verwendung dessen wird jedoch nur empfohlen, wenn die Verbindung bereits durch externe Verschlüsselungsmechanismen, beispielsweise durch VPN-Gateways, gesichert wird und die zusätzliche Verwendung von SSL/TLS lediglich eine Maximierung des Aufwandes an Rechenzeit bedeuten würde. Der ungeschützte Dienst „CIDAS-Plain“ läuft serverseitig auf Port 756/TCP⁶.

Dieses Dokument beschreibt lediglich Aufbau und Inhalt der von SSL oder TLS bzw. im Rahmen einer TCP-Session im Zusammenhang mit CIDAS transportierten Anwendungsdaten.

5.2.3 Identifizierungsmethoden

Entsprechend Schäfer [93, S. 66 f.] müssen Benutzer vor der Authentifizierung erst identifiziert werden. Dies geschieht durch die Übergabe von Identifizierungsdaten an den CIDAS-Server.

Die in Tabelle 5.1 aufgeführten drei Identifizierungsmethoden sind von Schäfer in [93, S. 66 f.] beschrieben. Der jeweilige Benutzer kann zwischen diesen drei Verfahren frei wählen. Sollte hierbei, insbesondere bei der Abfrage der Postanschrift, eine Person nicht eindeutig identifiziert werden können – denkbar sind beispielsweise mehrere gleichnamige Personen mit gleichem Wohnort – ist die Verwendung einer alternativen Identifizierungsmethode oder die Abfrage weiterer personenbezogener Daten möglich. Entsprechend Absprachen mit der Projektleitung ist die Sicherstellung eines hierfür ausreichenden Datenbestandes nicht Gegenstand dieser Arbeit, sondern kann vorausgesetzt werden.

In Abhängigkeit von der verwendeten Identifizierungsmethode stehen unter Umständen bestimmte Authentifizierungsverfahren nicht zur Verfügung. Das diesbezügliche Systemverhalten wird in Abschnitt 5.2.4 erläutert.

⁶Es sei darauf hingewiesen, daß die beiden bezeichneten Portnummern nicht bei der *Internet Assigned Numbers Authority* registriert wurden. Nachforschungen des Autors entsprechend werden die Ports derzeit nicht anderweitig verwendet.

Abgefragte Daten	Beschreibung der Identifizierungsmethode
Benutzername	CIDAS-Benutzer können sich bei der Registrierung im System einen Benutzernamen frei wählen und diesen im Nachhinein zur Identifizierung verwenden. Benutzernamen sind für den jeweiligen CIDAS-Server eindeutig und können nicht mehrfach verwendet werden.
E-Mail-Adresse	Auch bei der E-Mail-Adresse handelt es sich in der Regel um ein sogar weltweit eindeutig zuordenbares Identifizierungsmerkmal, das von CIDAS verwendet werden kann.
Postanschrift	Weiterhin kann eine Postanschrift, bestehend aus einem Namen, Vornamen und einem Wohnort, beschrieben durch Wohnort, Postleitzahl und Straße mit Hausnummer, als Identifizierungsmerkmal verwendet werden. Entsprechend der Konfiguration des Authentifizierungsservers können auch andere personenbezogene Daten genutzt werden.

Tabelle 5.1: Identifizierungsmethoden für CIDAS entsprechend Schäfer [93, S. 66 f.]

5.2.4 Authentifizierungsmethoden

Authentifizierungsmethoden werden im Rahmen verschiedener CIDAS-Nachrichten benötigt und über die im Folgenden spezifizierten Schlüsselworte referenziert. Sie werden in Klassen von Authentifizierungsmethoden eingeteilt. Die Zuordnung erfolgt aufgrund gemeinsamer Eigenschaften der Methoden. Zur Auswahl einer konkreten Methode wird jeweils die Klasse und die entsprechende Methode in Form einer XML-Struktur entsprechend Abschnitt 5.2.10 angegeben.

Die folgenden Klassen von Authentifizierungsverfahren können verwendet werden:

Klassenname	Beschreibung
NOTHING	Es wird keine Authentifizierung durchgeführt. Eine Autorisierung eines Benutzers mittels dieser Verfahrensklasse ist nach Abschluß der Identifikation beendet.
TEXT	Verfahren, die auf der Abfrage textueller Informationen basieren. Hierzu gehören beispielsweise die Abfrage zufällig ausgewählter Daten oder Passworte.
STORAGE_PASSIVE	Asymmetrische Kryptographie nutzende Verfahren, die passive Speichermedien wie Festplatten, CD-ROMs oder USB-Flash-Speicher als Schlüssel-speicher verwenden. Passive RFIDs werden in einer separaten Klasse behandelt.
STORAGE_ACTIVE	Kryptographische Verfahren, die aktive Speichermedien, also Medien, die bereits Sicherheitslogik integrieren, verwenden, werden in dieser Klasse zusammengefaßt. Chipkarten und aktive RFIDs werden in separaten Klassen behandelt.
RFID_PASSIVE	RFID-Geräte ohne integrierte Sicherheitslogik. Aufgrund der Vielzahl von Produkten und Herstellern werden diese Geräte in einer separaten Klasse zusammengefaßt.
RFID_ACTIVE	RFID-Geräte mit integrierter Sicherheitslogik. Aufgrund der Vielzahl von Produkten, Herstellern und kryptographischen Verfahren werden diese Geräte in einer separaten Klasse zusammengefaßt.
SMARTCARD	Auf Chipkarten basierende Verfahren. Aufgrund der Vielzahl von Produkten, Herstellern und kryptographischen Verfahren werden diese Geräte in einer separaten Klasse zusammengefaßt.
BIOMETRY	Biometrische Verfahren
MISC	Sonstige Verfahren

Tabelle 5.2: Spezifikation der Klassen von Authentifizierungsverfahren.

Wird die Identifizierung über die Postanschrift oder vergleichbare personenbezogene Daten des Benutzers realisiert, kommen zur Authentifizierung nur die Verfahren der Klassen `NOTHING` und `TEXT` zum Einsatz.

Für jede Verfahrensklasse gibt es eine Reihe konkreter Verfahren. Die in Tabelle 5.3 aufgeführten Verfahren werden entsprechend Schäfer [93, S. 67] von CIDAS angeboten, für einige Verfahrensklassen sind die im Einzelnen möglichen Authentifizierungsverfahren noch unspezifiziert:

Methodenname	Beschreibung
Verfahrensklasse <code>NOTHING</code>	
Diese Klasse hat keine Methoden	
Verfahrensklasse <code>TEXT</code>	
<code>NOPASS</code>	Das Wissen, kein Passwort zu haben.
<code>RANDDATA</code>	Abfrage eines zufälligen Datums aus den Benutzerdaten.
<code>PASSWORD</code>	Abfrage eines vorher festgelegten Passwortes.
Verfahrensklasse <code>STORAGE_PASSIVE</code>	
<code>X509</code>	Authentifizierung mittels asymmetrischer Kryptographie. Es wird Schlüsselmaterial entsprechend X.509 verwendet, der Benutzer hält sein Schlüsselmaterial auf einem passiven Datenträger bereit.
<code>OPGP</code>	Authentifizierung mittels asymmetrischer Kryptographie. Es wird Schlüsselmaterial entsprechend RFC 2440 verwendet, der Benutzer hält sein Schlüsselmaterial auf einem passiven Datenträger bereit.
Verfahrensklasse <code>STORAGE_ACTIVE</code>	
Die Methoden dieser Klasse sind in Zusammenarbeit mit Herstellern und Anwendern entsprechender Produkte zu konzipieren.	
Verfahrensklasse <code>RFID_PASSIVE</code>	
Die Methoden dieser Klasse sind in Zusammenarbeit mit Herstellern und Anwendern entsprechender Produkte zu konzipieren.	
Verfahrensklasse <code>RFID_ACTIVE</code>	

Tabelle 5.3: Spezifikation der Authentifizierungsmethoden in Abhängigkeit von den möglichen Verfahrensklassen.

Methodenname	Beschreibung
	Die Methoden dieser Klasse sind in Zusammenarbeit mit Herstellern und Anwendern entsprechender Produkte zu konzipieren.
	Verfahrensklasse SMARTCARD
	Die Methoden dieser Klasse sind in Zusammenarbeit mit Herstellern und Anwendern entsprechender Produkte zu konzipieren.
	Verfahrensklasse BIOMETRY
	Die Methoden dieser Klasse sind in Zusammenarbeit mit Herstellern und Anwendern entsprechender Produkte zu konzipieren. Es wird davon ausgegangen, daß hierfür das vom <i>BioAPI Consortium</i> in [46] spezifizierte Framework verwendet wird.
	Verfahrensklasse MISC
	Die Methoden dieser Klasse sind in Zusammenarbeit mit Herstellern und Anwendern entsprechender Produkte zu konzipieren.

Tabelle 5.3: Spezifikation der Authentifizierungsmethoden in Abhängigkeit von den möglichen Verfahrensklassen.

Authentifizierungsverfahren können beliebig miteinander kombiniert werden, so daß ein Benutzer, wenn er dies wünscht oder der Authentifizierungserver bzw. die verwendete Applikation dies voraussetzt, mehrere Verfahren nacheinander im Rahmen eines Authentifizierungsvorgangs verwenden kann. Der Benutzer gilt hierbei nur dann als erfolgreich authentifiziert, wenn alle Verfahren erfolgreich abgeschlossen wurden.

5.2.5 Sicherheitslevel

Von Schäfer wird in [93, S. 68 f.] wird der Begriff „Sicherheitslevel“ eingeführt und kurz erläutert. Sicherheitslevel geben an, wie sicher die von einem Benutzer verwendeten Authentifizierungsverfahren sind. Beispielsweise wird ein Benutzer, der sich mittels eines kryptographischen und eines biometrischen Authentifizierungsverfahrens von CIDAS hat autorisieren lassen, einen höheren aktuellen Sicherheitslevel haben als ein anderer Benutzer, der die Authentifizierungsklasse **NOTHING** verwendet hat. Sicherheitslevel drücken also

aus, inwieweit Dritte der Identität eines authentifizierten Benutzers Glauben schenken können.

Neben dem aktuellen Sicherheitslevel gibt es für jeden Benutzer ein minimales Sicherheitslevel, das die bei der Anmeldung mindestens zu verwendenden Authentifizierungsverfahren eingrenzt. Darüber hinaus kann er auch für Kommunikationspartner, beispielsweise Applikationen innerhalb von CIDAS, ein minimales Sicherheitslevel setzen, das sich dahingehend auswirkt, als daß er nur solche Applikationen nutzen wird, die diesem Sicherheitslevel entsprechen. Die Applikationen definieren wiederum eigene minimale Sicherheitslevel für die von ihnen akzeptierten Benutzer.

Die Zuordnung der Sicherheitslevel zu Authentifizierungsverfahren oder Kombinationen von Authentifizierungsverfahren ist serverseitig konfigurierbar. Gleiches gilt für die in Abhängigkeit vom aktuellen Sicherheitslevel verfügbare CIDAS-Funktionalität. Im Rahmen mehrerer Gespräche mit der Projektleitung im Juni 2003 entstand der in Tabelle 5.4 auf Seite 96 dargestellte Vorschlag für die Einstufung von Authentifizierungsverfahren in Sicherheitslevel. Die Angaben erfolgen in aufsteigender (unsichere Authentifizierung → sicherere Authentifizierung) Reihenfolge.

5.2.6 Token

Token dienen zur Überprüfung der Authentizität eines Benutzers seitens Dritter, die ebenfalls authentifizierte Clients bei ein und demselben CIDAS-Server sein müssen. Hierzu werden Token auf externem Wege vom Tokeneigentümer dem Kommunikationspartner übertragen. Dieser kann den Token zum CIDAS-Server zurücksenden und von ihm überprüfen lassen. Um den korrekten Server hierfür ausfindig zu machen, enthält der Token einen CIDAS-Server-Identifizierer. Im Rahmen eines `CIDAS_MTYPE_CSRREQ` kann ein Client den CIDAS-Server-Identifizierer jedes Servers, bei dem er authentifiziert ist, abfragen. Die für die Speicherung von Token bzw. zur Übertragung verwendeten XML-Strukturen werden in Abschnitt 5.2.10 vorgestellt.

Sicherheitslevel	Authentifizierungsverfahren bzw. Klassen von Authentifizierungsverfahren
0	Eindeutige Identifizierung ohne Authentifizierung
1	Eindeutige Identifizierung und Verwendung des Authentifizierungsverfahrens TEXT:NOPASS
2	Eindeutige Identifizierung und Verwendung des Authentifizierungsverfahrens TEXT:RANDDATA
3	Eindeutige Identifizierung und Verwendung des Authentifizierungsverfahrens TEXT:PASSWORD
4	Identifizierung über einen Benutzernamen oder eine E-Mail Adresse und Authentifizierung über ein Verfahren der Klasse BIOMETRY
5	Identifizierung über einen Benutzernamen oder eine E-Mail Adresse und Authentifizierung über ein Verfahren der Klasse BIOMETRY und das Verfahren TEXT:PASSWORD
6	Identifizierung über einen Benutzernamen oder eine E-Mail Adresse und Authentifizierung über ein Verfahren der Klassen STORAGE_PASSIVE oder RFID_PASSIVE
7	Identifizierung über einen Benutzernamen oder eine E-Mail Adresse und Authentifizierung über ein Verfahren der Klassen STORAGE_ACTIVE , RFID_ACTIVE oder SMARTCARD
8	Identifizierung über einen Benutzernamen oder eine E-Mail Adresse und Authentifizierung über ein Verfahren der Klassen STORAGE_PASSIVE oder RFID_PASSIVE und ein Verfahren der Klasse BIOMETRY
9	Identifizierung über einen Benutzernamen oder eine E-Mail Adresse und Authentifizierung über ein Verfahren der Klassen STORAGE_ACTIVE , RFID_ACTIVE oder SMARTCARD und ein Verfahren der Klasse BIOMETRY

Tabelle 5.4: Sicherheitslevel in CIDAS

Token werden erst nach Abschluß der Authentifizierung auf Anfrage des Clients vergeben. Sie haben eine zeitlich begrenzte Gültigkeit und müssen, werden sie weiterhin vom Client benötigt, regelmäßig erneuert werden. Der Gültigkeitszeitraum eines Tokens ist durch den Ausstellungszeitpunkt und den Endzeitpunkt seiner Gültigkeitsdauer angegeben. Bei beiden Daten handelt es sich um vorzeichenlose ganze Zahlen in dezimaler Zahlendarstellung, die als Zeitangabe in Form der verstrichenen Sekunden seit Mitternacht, 01. Januar 1970 (UTC) zu interpretieren sind. Die Gültigkeitsdauer der Token kann server- und benutzerseitig konfigurierbar sein.

Ist der Gültigkeitszeitraum eines Tokens zur Hälfte abgelaufen, kann der Client ein weiteres Token anfordern. Ein Client hat damit zu einem beliebigen Zeitpunkt maximal zwei gültige Token von einem Server, sendet er weitere CIDAS_MTYPE_TOKREQ-Nachrichten (siehe Abschnitt 5.2.9), werden ihm keine neuen Token zugestellt.

Das Token selbst ist ein 128 Bytes langes, vom Server zufällig gewähltes Datenpaket. Es ist für den jeweiligen Server während seiner Gültigkeit und kurzfristig auch darüber hinaus eindeutig genau einem Benutzer zuordenbar.

Zur Übertragung des Tokens wird eine XML-Struktur des Typs `token` verwendet. Da ein Token keinerlei Rückschlüsse auf den zugehörigen Benutzer zuläßt und auch nicht ohne dessen Zustimmung verwendet werden kann, kann das Token unverschlüsselt übertragen und gespeichert werden.

5.2.7 Fehler und Systemverhalten bei Fehlern

Mögliche auftretende Fehler bei der Kommunikation zwischen CIDAS Server und Client beschränken sich vorrangig auf das Nichteinhalten der in diesem Dokument beschriebenen Vorschriften für die Kommunikation, auf Zeitüberschreitungen, möglicherweise auf Verbindungszusammenbrüche und auf fehlerhafte Identifizierungs- oder Authentifizierungsdaten.

Sollten falsche oder fehlerhafte Identifizierungs- oder Authentifizierungsdaten zum Fehlschlagen der Authentifizierung geführt haben, kann der Server nach dem Senden einer `AUTHFAIL`-Nachricht und einer implementierungsabhängigen Pause durch das Senden einer Nachricht des Typs `IDDATAREQ` den Authentifizierungsprozeß erneut starten oder die Verbindung schließen. Das gleiche Verhalten wird auch dann verwendet, wenn der Benutzer des Clients bereits authentifiziert war, sich jedoch mit einem höherwertigeren Verfahren erneut authentifizieren will. Serverimplementierungen speichern Informationen über fehlgeschlagene Authentifizierungsversuche und benachrichtigen den Benutzer nach Abschluß der nächsten fehlerfreien Authentifizierung. Hierfür bietet sich die Verwendung des Nachrichtentyps `INFOMSG` an.

Sind die angegebenen Identifizierungsdaten nicht eindeutig oder falsch, wird das Protokoll nach Möglichkeit trotzdem fortgesetzt.

Fehler bei der Verwendung des Protokolls und Zeitüberschreitungen werden mit dem Abbruch der Verbindung quittiert. Sowohl Server- als auch Client-Implementierungen warten nach dem Absenden einer Anfrage mindestens 30 Sekunden auf eine Antwort. Diese Zeit ist implementierungsabhängig bzw. frei konfigurierbar. Bei Anfragen, die eine Interaktion des Benutzers voraussetzen⁷, beträgt diese Zeit mindestens 3 Minuten.

5.2.8 Aufbau einer Nachricht

Alle CIDAS-Nachrichten sind nach dem in Abbildung 5.1 dargestellten Schema aufgebaut. Hierbei entspricht eine Spalte der Abbildung jeweils einem 8-Bit-Byte. Das Gesamtvolumen des Payloads ist variabel. Die in der Abbildung verwendeten Bezeichner `PVer`, `PSubVer`, `MType`, `Flags`, `SeqC`, `Psize` und `SID` haben die in Tabelle 5.5 angegebene Bedeutung, sie werden im Folgenden näher erläutert.

⁷Beispielsweise erfordert die Abfrage von Identifizierungs- und Authentifizierungsdaten unter Umständen eine etwas längere Bearbeitungszeit, da der Benutzer die angefragten Daten evtl. erst per Tastatur eingeben oder Hardware an das System anschließen muß.

Oktett 0	Oktett 1	Oktett 2	Oktett 3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
PVer	PSubVer	MType	Flags
SeqC0	SeqC1	PSize0	PSize1
SID0	SID1	SID2	SID3
SID4	SID5	SID6	SID7
Payload			
Payload			
Payload			
...			

Abbildung 5.1: Aufbau einer CIDAS-Nachricht.

Bezeichner	Beschreibung	Größe in Bytes
PVer	'Protocol Version', Protokoll Version	1
PSubVer	'Protocol Subversion', Protokoll Unterversi- on	1
MType	'Message Type', der Typ der Nachricht	1
Flags	Flags, Bitschalter	1
SeqC	'Sequence Counter', ein Nachrichtenzähler	2
PSize	'Payload Size', Zahl der Bytes im Payload	2
SID	'Session Identifier', Verbindungsnummer	8

Tabelle 5.5: Bedeutung der Bezeichner aus Abbildung 5.1.

Protokoll-Version und -Unterversion

Unter Umständen wird das CIDAS-Protokoll in der Zukunft noch mehrere Revisionen erfahren. Um eine Abwärtskompatibilität zukünftiger CIDAS-Server und Clients zu ermöglichen, beginnt jede Nachricht mit der Versionsnummer des Protokolls, nach dem diese zu interpretieren ist. Sowohl die Version als auch die Unterversion sind als ganze 8-Bit Zahlen zu interpretieren. Bei Nachrichten, die den in diesem Dokument gemachten Spezifikationen entsprechen, ist PVer auf 01h und PSubVer auf 00h zu setzen.

Der Message-Type

Der Message-Type **MType** spezifiziert die Art der vorliegenden Nachricht. Die möglichen Belegungen werden an späterer Stelle aufgeführt und erläutert. **MType** ist als ganze, positive 8-Bit-Zahl, wie am Anfang dieses Abschnitts dargestellt, zu interpretieren.

Flags

Zur Spezifikation des Inhalts der jeweiligen Nachricht wird primär der Nachrichtentyp **MType** verwendet, einige besondere Eigenschaften der Nachricht können jedoch durch das Feld **Flags** ausgedrückt werden. Das Feld hat eine Länge von einem Oktett. Jedem Bit ist eine gesonderte Funktion zugeordnet. Der Aufbau des Feldes und die Bedeutung der Bits sind Abbildung 5.2 und Tabelle 5.6 zu entnehmen. In einer CIDAS-Referenzimplementierung sollen hierfür Bezeichner der Art `CIDAS_FLAG_<Flag-Bezeichner>` verwendet werden.

0	1	2	3	4	5	6	7
R	C	r	r	r	r	T	T

Abbildung 5.2: Aufbau des **Flags**-Feldes.

Bit	Bezeichner und Beschreibung
0	RADIX : Der Payload der Nachricht enthält Radix-64-kodierte Daten, falls das Bit gesetzt ist.
1	COMPRESSION : Der Payload der Nachricht enthält komprimierte Daten, falls das Bit gesetzt ist.
2 - 5	RESERVED : Reserviert, derzeit ohne Bedeutung.
6, 7	TESTING : Reserviert zu Testzwecken.

Tabelle 5.6: Bedeutung der Flags in Abbildung 5.2.

Der Sequence-Counter

Der Sequence-Counter **SEQC** ist als vorzeichenlose, ganze, zwei Bytes lange Zahl zu interpretieren. Er muß bei der ersten Anfrage vom Client auf Null (0000h) gesetzt werden. Nimmt der Server die Anfrage an, initialisiert er den **SEQC** mit einer Pseudo-Zufallszahl. **SEQC** muß in jeder Nachricht enthalten sein und wird vor jeder Übertragung sowohl seitens des Clients als auch seitens des Servers um 1 inkrementiert. Erreicht **SEQC** bei einer Inkrementierung den Wert FFFFh, wird er bei einer darauf folgenden erneuten Inkrementierung auf Null gesetzt.

Der Session-Identifizier

Der Session-Identifizier **SID** ist eine numerische, dem Client vom Server zugewiesene Identifikationsgröße. Er ist als ganze, positive 64-Bit-Zahl wie am Anfang dieses Abschnitts dargestellt, zu interpretieren. Aus Sicherheitsgründen sollte er serverseitig nicht oder nur in Verbindung mit anderen Verbindungsmerkmalen wie der Adresse und dem verwendeten Port auf dem Client-System zur Identifikation bestehender Verbindungen eingesetzt werden. Der **SID** muß in allen Nachrichten enthalten sein und ist sowohl seitens des Clients als auch seitens des Servers auf Korrektheit zu prüfen. Für Nachrichten vom Client zum Server vor Zuweisung eines **SID** hat dieser Null (00h) zu sein. Nimmt der Server die Anfrage an, initialisiert er den **SID** mit einer von Dritten nicht vorhersehbaren Zahl ungleich Null. Die Verwendung einer Pseudo-Zufallszahl bietet sich hier an. Im folgenden Verlauf der Verbindung ist Null kein valider **SID** mehr. Session-Identifizier werden vom CIDAS-Server eindeutig vergeben, so daß zu einem bestimmten Zeitpunkt keine zwei Verbindungen mit gleichem **SID** existieren. Die Maximalzahl von akzeptierten Verbindungen pro Server ist damit beschränkt auf $2^{64} - 1$.

Der Payload und Payload-Size

Der Payload enthält die eigentlichen Nutzdaten und zugehörige Metadaten, der Payload darf Null Bytes lang sein. Er wird nur soweit eingelesen, wie es **PSize** vorgibt. **PSize** ist als eine vorzeichenlose, ganze Zahl mit einer Län-

ge von zwei Bytes zu interpretieren. Die maximale Länge für die Nutzdaten beträgt aufgrund der Größe von `PSize` 65534 (FFFFh) Bytes. Um eventuell größere Datenmengen transportieren zu können, beispielsweise aufwendige Authentifikationsdaten bei Nutzung biometrischer Authentifikationsverfahren oder Protokolldaten der CIDAS Special-Feature-Module, darf `PSize` auf FFFFh gesetzt werden. Dies hat zur Folge, daß die 65534 Datenbytes aus dem Payload dieser Nachricht konkateniert mit maximal 65534 Datenbytes des Payloads der nächsten Nachricht zusammen das eigentliche Datenpaket ergeben. Dieses Verfahren kann beliebig häufig wiederholt werden, Serverimplementierungen überprüfen den Datenstrom bzw. jede Einzelnachricht auf Validität. Das maximal transportierbare Volumen von Nutzdaten ist also prinzipiell unbegrenzt, Serverimplementationen definieren unter Umständen harte oder konfigurierbare Limits.

Datenorganisation im Payload

Der Payload einer Nachricht enthält, sofern er nicht leer ist, immer mittels XML strukturiert gespeicherte Daten.

Datenkomprimierung im Payload

Aus Performancegründen sind komprimierte Daten im Payload einer Nachricht zulässig, jedoch nur von Nachrichten, die vom Server zu einem Client übertragen werden. Ein komprimierter Payload wird durch Setzen des Bits 1 im `Flags`-Feld angezeigt. Als Komprimierungsverfahren werden die in RFC 1950: „ZLIB Compressed Data Format Specification“ (siehe [49]) beschriebenen bzw. referenzierten Verfahren verwendet.

Radix-64-Konversion

Insbesondere bei Nutzung der gebotenen Möglichkeiten zur Komprimierung des Payloads einer Nachricht basiert die innerhalb von CIDAS verwendete Datenrepräsentation auf willkürlichen 8 Bit langen Datenbytes. Da verschiedene Systeme lediglich die Verwendung der ersten 7 Bits, also der druckbaren ASCII-Zeichen zulassen, ist eine Konvertierung unter Umständen nötig.

Bei der Radix-64-Kodierung werden acht Bit lange Datenbytes beliebigen Inhalts auf ASCII-Zeichen abgebildet. Das Verfahren ist identisch mit dem in RFC 2440, [43, S. 7, 41-48], vorgestellten. Die Radix-64-Konversion darf für CIDAS-Nachrichten beliebigen Inhalts verwendet werden. Sowohl Server als auch Client müssen das Verfahren unterstützen.

Die Verwendung von Radix-64 ist durch das Setzen des Bits 0 im **Flags**-Feld der Nachricht anzuzeigen. Das Bit ist ausschließlich zu setzen, wenn der gesamte Payload Radix-64 kodiert ist.

Sind in einer Nachricht sowohl Komprimierung als auch Radix-64-Konversion eingeschaltet, so wurde die betroffene Nachricht zuerst komprimiert und anschließend, vor dem Versenden, mittels Radix-64 in 7 Bit-Zeichen konvertiert. Zur Auswertung der Nachricht ist das umgekehrte Vorgehen anzuwenden.

5.2.9 Definition der Nachrichtentypen, Protokollablauf

Im Folgenden seien die einzelnen möglichen und während des Prozesses der Identifikation, der Authentifikation bzw. zur Aufrechterhaltung einer Verbindung zwischen Client und Server und zur weiteren Nutzung der von CIDAS gebotenen Funktionalitäten verwendeten Nachrichtentypen dargestellt und erläutert. Die Auflistung erfolgt größtenteils in der Reihenfolge, in der die einzelnen Nachrichten im Verlauf einer CIDAS-Sitzung übertragen werden.

Jeder Nachricht ist ein Nachrichtenname mit einer Kurzform und einem numerischen Wert in hexadezimaler Zahlendarstellung zugewiesen. Beispielsweise ist **CONREQ** die Kurzform der Nachricht mit dem Namen „Connection Request“. Dieser Nachricht ist der numerische Wert 01h zugeordnet. Im Verlauf des CIDAS-Protokolls werden Nachrichten ausschließlich nach diesem numerischen Wert identifiziert, er stellt den **MType** einer jeden Nachricht dar.

Eine Referenzimplementierung sollte für diese Nachrichtentypen eindeutige Bezeichner der Art `CIDAS_MTYPE_<Kurzname>` verwenden. In C nutzt man dabei sehr häufig Präprozessor-Anweisungen⁸. Aus Gründen der Übersichtlichkeit verzichten wir hier darauf, alle Nachrichtentypen nach dieser Regel zu benennen, bei späteren Verweisen auf einzelne Nachrichtentypen wird die an dieser Stelle vorgeschlagene Notation dennoch genutzt werden.

Die jeweiligen Nachrichten enthalten, falls notwendig, in ihrem Payload XML-Strukturen entsprechend Abschnitt 5.2.10. Diese entsprechen der in Anhang B zu findenden Data-Type-Definition für CIDAS. Sie enthalten also jeweils mindestens einen `cidas`-Root Tag. Die in diesem Abschnitt als Payload der jeweiligen Nachricht angegebenen XML-Tags bezeichnen die jeweils direkt in diesen Root-Tag eingeschlossenen Tags.

Eröffnung der Verbindung

CONREQ (01h) Connection Request.

Verbindungsanfrage: Der Client eröffnet eine Verbindung zum Server. Nach erfolgreichem SSL/TLS-Handshake wird zuerst diese Nachricht übertragen. Sie wird genutzt, um die im Folgenden zwischen Client und Server verwendete Version des CIDAS-Protokolls auszuhandeln. Der Client setzt hierzu über die Nachrichten-Felder `PVer` und `PSubVer` die höchste von ihm unterstützte Protokollversion. Diese Nachricht muß in allen, auch zukünftigen Versionen des Protokolls gleich kodiert sein.

Der Payload einer Nachricht dieses Typs ist leer.

CONACC (02h) Connection Accept.

Akzeptiert der Server die Verbindung, sendet er diese Nachricht zurück an den Client. Anderfalls wird er die Verbindung schließen. Die Nachricht dient primär zum weiteren Aushandeln der verwendeten Protokollversion: Der Server muß in dieser Nachricht die von ihm unterstützte Protokollversion setzen, die der vom Client verwendeten am nächsten ist. Darüber hinaus werden die

⁸siehe Seite 169: „cidas_1-0.h“

Felder `SeqC` und `SID` initialisiert.

Nach dem Erhalt einer `CONACC`-Nachricht kann der Client die vorgeschlagene Protokollversion akzeptieren, indem er die Nachricht mit inkrementiertem `SeqC` zurückschickt, oder die Verbindung trennen. Wird die Protokollversion akzeptiert, ist sie für die gesamte folgende CIDAS-Sitzung bindend.

Aus Kompatibilitätsgründen muß auch diese Nachricht in zukünftigen Versionen des Protokolls gleich kodiert sein.

Der Payload einer Nachricht dieses Typs ist leer.

Austausch von Identifikationsdaten

`IDDATAREQ` (03h) Identification Data Request.

Um den Identifizierungsprozeß einzuleiten, überträgt der Server mit dieser Nachricht die zulässigen Identifizierungsmethoden an den Client.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `idmethods`-Tag.

`IDDATA` (04h) Identification Data.

Antwort auf `IDDATAREQ`. Der Client übermittelt Daten bezüglich der Identität des bedienenden Benutzers.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `iddata`-Tag.

Die übergebenen Identifizierungsdaten werden nach dem Empfang durch den CIDAS-Server auf formale Korrektheit überprüft. Entsprechen die übermittelten Identifizierungsdaten nicht dem in der `IDDATAREQ`-Nachricht spezifizierten Schema, kann der Authentifizierungsserver die `IDDATAREQ`-Nachricht erneut senden oder die Verbindung abbrechen.

Wurde ein eindeutiges Identifizierungsmerkmal, beispielsweise die E-Mail-Adresse oder der CIDAS-Benutzername zur Benutzerauthentifizierung verwendet, wird das Protokoll ohne Auswertung der Identifizierungsdaten fortgesetzt.

Reichen bei der Benutzeridentifizierung über personenbezogene Daten die übermittelten Identifizierungsdaten nicht aus, um den Benutzer eindeutig zu identifizieren, kann die `IDDATAREQ`-Nachricht ebenfalls erneut gesendet werden. Die verwendete XML-Struktur ist in diesem Fall derart anzupassen, daß die Identifizierung nach dem nächsten Versuch eindeutig ist. Auch wenn die Identifizierungsdaten eindeutig keinem Benutzer zugeordnet werden können, wird das Protokoll fortgesetzt.

Nachrichten zur Authentifizierung

AUTHREQ (05h) Authentication Request.

Nach dem Senden der Identifikationsdaten kann der Client eine Authentifizierungsanfrage an den Server senden, in der spezifiziert wird, welche Authentifizierungsverfahren im Folgenden verwendet werden sollen.

Der Client kann diese Anfrage, nachdem ein Authentifizierungsversuch erfolgreich abgeschlossen wurde, erneut senden, um sich mittels einer als qualitativ nieder- oder höherwertig klassifizierten Methode erneut zu authentifizieren und damit seine Vertrauensstufe zu ändern.

Auch der Server kann diese Anfrage zum Client senden, falls dieser auf einen Dienst zugreifen möchte, für den seine Authentifizierungsstufe nicht genügt. Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `authmethods`-Tag.

AUTHDATAREQ (06h) Authentication Data Request.

Als Reaktion auf eine `AUTHREQ`-Nachricht vom Client fragt der Server mit dieser Nachricht Authentifizierungsdaten wie beispielsweise Passworte ab. Es wird für jedes abgefragte Datum, also in der Regel für jede Authentifizierungsmethode, eine eigene Nachricht verwendet. Authentifizierungsverfahren, die mehrere aufeinander aufbauende Nachrichten verwenden, sind möglich. Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `authdatareq`-Tag.

AUTHDATA (07h) Authentication Data.

Eine Nachricht dieses Typs enthält eine Antwort auf eine `AUTHDATAREQ`-

Nachricht.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `authdata`-Tag.

Auch hierbei kann eine Nachricht des Typs `AUTHDATAREQ` erneut gesendet werden, falls die darauf gegebene Antwort formal inkorrekt war. Alternativ steht es dem Authentifizierungsserver wiederum frei, die Verbindung abzubrechen.

Erst wenn alle benötigten Identifizierungs- und Authentifizierungsdaten abgefragt wurden und formal korrekt sind, beginnt der Authentifizierungsserver mit der Auswertung der Daten. Die einzige Ausnahme von dieser Regel stellen in mehreren aufeinander aufbauenden Schritten ablaufende kryptographische Authentifizierungsprotokolle dar.

`AUTHSUCC` (08h) Authentication Success.

Die erfolgreicher Abfrage und Auswertung der Identifikations- und Authentifikationsdaten wird dem Client mittels `AUTHSUCC` signalisiert. Die Verbindung bleibt offen, der Client wird üblicherweise einen `TOKREQ` schicken, um ein Token zu erhalten.

Der Payload einer Nachricht dieses Typs ist leer.

`AUTHFAIL` (09h) Authentication Failure.

Fehler bei der Anmeldung werden mit einer `AUTHFAIL`-Nachricht quittiert. Nach dem Senden dieser Nachricht kann ein erneuter Identifizierungs- und Authentifizierungsversuch unternommen werden, der durch ein erneutes Senden einer `IDDATAREQ`-Nachricht an den Client initiiert wird. Alternativ kann der Server die Verbindung auch schließen.

Im Payload der Nachricht können Informationen über den Grund des Fehlschlagens übermittelt werden. Dies bezieht sich jedoch nur auf formale Aspekte des Authentifizierungsprozesses, wie beispielsweise das Nichteinhalten des Protokolls oder Zeitüberschreitungen. Grundsätzlich wird kein direkter Hinweis auf ein fehlerhaftes Passwort, einen fehlerhaften Schlüssel oder Vergleich-

bares gegeben, was einem Angreifer einen Ansatz zum Durchprobieren möglicher Eingabedaten geben könnte.

Der Payload einer Nachricht dieses Typs kann eine XML-Struktur mit einem *infomsg*-Tag enthalten.

Behandlung von Tokens

TOKREQ (0Ah) Token Request.

Mit diesem Request fragt ein Client beim Server ein Token an. Token haben eine zeitlich begrenzte Gültigkeit. Ist die Gültigkeitsdauer zur Hälfte abgelaufen, kann der Client ein neues Token anfordern, wozu ebenfalls diese Nachricht verwendet wird.

Der Payload einer Nachricht dieses Typs ist leer.

TOKDATA (0Bh) Token Data.

Antwort auf TOKREQ: Tokenzuweisung.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem *token*-Tag.

TOKFAIL (0Ch) Token Failure.

Antwort auf TOKREQ: Die Tokenzuweisung schlug fehl. Der Payload der Nachricht soll Informationen über den Grund des Fehlschlagens enthalten.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem *infomsg*-Tag.

Nachdem der Client ein Token erhalten hat, kann er es entsprechend Abschnitt 5.2.6 an Applikationen als Autorisierungszeichen weitergeben. Diese Applikationen müssen ebenfalls bei ein und demselben CIDAS-Server authentifizierte Clients sein und können das Token vom Authentifizierungsserver auf seine Validität überprüfen lassen.

TOKVERREQ (0Dh) Token Verification Request.

Ein Client hat auf externem Wege den Token eines anderen, evtl. authentifizierten Benutzers erhalten und erwartet vom Server eine Bestätigung der

Validität dieses Tokens. Der Server hat hierbei sowohl die Validität des Tokens an sich als auch die minimalen Schutzbedürfnisse beider Parteien zu überprüfen, bevor er die Anfrage beantwortet.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `token_ver_record`-Tag und kann in dieser eine `infomsg` enthalten, in der dem Benutzer der Grund für die Überprüfung mitgeteilt wird.

TOKCONFREQ (0Eh) Token Confirmation Request.

Der Server gibt den im Rahmen eines `TOKVERREQ` erhaltenes `token` zur Bestätigung an den Client zurück. Dem Token werden hierzu Informationen über den die `TOKVERREQ`-Anfrage stellenden Dienst oder Benutzer beigefügt.

Obiges Verhalten ist optional und kann seitens des Benutzers frei konfiguriert werden.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `token_peer_record`-Tag.

TOKCONFACC (0Fh) Token Confirmation Accept.

Der Client akzeptiert eine `TOKCONFREQ`-Anfrage und erkennt damit möglicherweise implizit die über den `infomsg`-Tag übergebene Nachricht an. Dieser Mechanismus ist gedacht, um dem Benutzer eventuelle Konsequenzen aus der Benutzung der Applikation deutlich zu machen und sicherzustellen, dass er diese kennt.

Der Payload einer Nachricht dieses Typs ist identisch mit dem der korrespondierenden `TOKCONFREQ`-Nachricht.

TOKCONFDEC (10h) Token Confirmation Decline.

Der Client lehnt die `TOKCONFREQ`-Anfrage ab, die Überprüfung ist damit gescheitert, der Server muß dem die `TOKVERREQ`-Anfrage stellenden Dienst oder Benutzer eine entsprechende `TOKVERFAIL`-Nachricht zustellen.

Der Payload einer Nachricht dieses Typs ist identisch mit dem der korrespondierenden `TOKCONFREQ`-Nachricht.

TOKVERSUC (11h) Token Verification Success.

Die Verifizierung eines Tokens war erfolgreich. Diese Nachricht wird demjenigen Client zugestellt, der eine Tokenverifizierung angefragt hat, sie enthält nochmals das für valid befundene Token.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `token_ver_succ`-Tag.

TOKVERFAIL (12h) Token Verification Failure.

Die Verifizierung eines Tokens schlug fehl. Diese Nachricht wird demjenigen Client zugestellt, der eine Tokenverifizierung angefragt hat. Der Client erhält keine weiteren Informationen über den Grund des Fehlschlagens. Die Nachricht enthält das als ungültig eingestufte Token, sie kann auch noch zugestellt werden, nachdem ein Token bereits bei einer vorhergehenden Überprüfung für valid befunden wurde, beispielsweise, wenn der betroffene Benutzer sich ausloggt.

Der Inhalt des Payloads dieser Nachricht ist identisch mit dem der korrespondierenden **TOKVERREQ**-Nachricht.

TOKPERMDEN (13h) Token Permission Denied.

Die Verifizierung des Tokens des Clients, an den diese Nachricht zugestellt wird, schlug fehl. Der Client hatte sein Token auf externem Wege einem anderen Client zur Verfügung gestellt, dieser ließ das Token verifizieren und der Authentifizierungsserver befand es für ungültig oder erlaubte aus anderen Gründen keinen Zugriff auf die Applikation. Dieser Fall tritt üblicherweise dann ein, wenn die minimalen Sicherheitsanforderungen eines Kommunikationspartners unterschritten wurden. Der Benutzer kann dieses Problem lösen, indem er sich erneut authentifiziert und dabei eine höhere Authentifikationsmethode verwendet oder sein minimalen Sicherheitslevel für Kommunikationspartner reduziert. Ist eine erneute Authentifizierung erforderlich, kann die minimal notwendige Methode in einer auf diese Nachricht folgende **AUTHREQ**-Nachricht spezifiziert sein.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `token_perm_den`-Tag.

Abfrage zusätzlicher Fähigkeiten des Servers, Nutzung dieser Funktionalität

CAPREQ (14h) Capability Request.

Mittels dieser Anfrage kann ein authentifizierter Client Informationen über besondere Fähigkeiten eines CIDAS-Servers, beispielsweise installierte Special Feature Module anfragen. Serverimplementierungen schränken die Häufigkeit der Nutzung dieser Anfrage unter Umständen ein.

Der Payload einer Nachricht dieses Typs ist leer, wenn eine Übersicht über die verfügbare Funktionalität angefragt wird. Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `capability`-Tag, wenn detaillierte Informationen zu einer Funktion abgefragt werden.

CAPDATA (15h) Capability Data.

Die Antwort des Servers auf einen **CAPREQ**. Bei einer allgemeinen Anfrage werden nur Namen, Hersteller und Versionsinformationen der verfügbaren Module/Fähigkeiten ausgegeben. Spezielle Anfragen werden mit erweiterten Informationen beantwortet.

Es werden nur Informationen über Funktionen des Servers zurückgegeben, auf die der Benutzer mit seiner derzeitigen Authentifizierungsstufe Zugriff hat.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `capdata`-Tag.

SFMREQ (16h) Special Feature Module Request

Zusätzliche Funktionalität des CIDAS-Servers kann über eine **SFMREQ**-Nachricht verwendet werden. Im Payload dieser Nachricht werden das jeweilige Special-Feature-Modul spezifiziert und mögliche Parameter übergeben.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `sfmdata`-Tag.

SFMRESP (17h) Special Feature Module Response

Die Antwort auf eine **SFMREQ**-Nachricht wird in dieser Nachricht übertragen.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `sfmdata`-Tag.

Abfrage des CIDAS Server Records

`CSRREQ` (18h) CIDAS Server Record Request.

Ein Client hat die Möglichkeit, Informationen über den Server, den sog. CIDAS Server Record, abzufragen. Serverimplementationen beschränken unter Umständen die maximale Häufigkeit für das Stellen dieser Anfrage.

Der Payload einer Nachricht dieses Typs ist leer.

`CSRDATA` (19h) CIDAS Server Record Data.

Der Server gibt als Antwort auf eine Anfrage des Typs `CSRREQ` einen Server Record mit verschiedenen Informationen über sich selbst zurück.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `serverrecord`-Tag.

Austausch und Änderung von Benutzer- und Applikationsdaten

`DATAREADREQ` (1Ah) Data Read Request.

Ein authentifizierter Client kann mittels dieser Nachricht Daten eines beim Server authentifizierten Benutzers, dessen Token er kennt, anfordern.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `datareadrequest`-Tag.

`DATAWRITEREQ` (1Bh) Data Write Request.

Ein authentifizierter Client kann mittels dieser Nachricht Daten eines beim Server authentifizierten Benutzers, dessen Token er kennt, ändern.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `datawriterequest`-Tag.

Im Rahmen einer `DATAREADREQ`-Nachricht können sowohl personenbezogene Daten bezüglich des durch das Token identifizierten Benutzers, als auch

benutzerspezifische Applikationsdaten (vgl. Schäfer, [93, S. 69 f.]) abgefragt werden. Die Datenabfrage kann nur erfolgen, wenn der Benutzer eingeloggt ist – andernfalls würde auch kein valides Token existieren. Eine Applikation kann immer ihr zugeordnete Applikationsdaten auslesen oder ändern. Für das Auslesen oder Ändern der personenbezogenen Daten des Benutzers ist eine Bestätigung durch den jeweiligen Benutzer erforderlich. Über entsprechende Attribute im Bereich der personenbezogenen Daten eines Benutzers kann hierfür eine generell gültige Bestätigung vorhanden sein, die vom CIDAS-Server ausgewertet wird. Auch hat das Sicherheitslevel der die Daten anfragenden Applikation Auswirkungen auf das Verhalten des Servers. Ganz allgemein sollten bei einem Sicherheitslevel kleiner 6 (vgl. hierzu Tabelle 5.4 auf Seite 96) keine sicherheitskritischen Daten wie beispielsweise Telephonnummern, E-Mail-Adressen oder gar Kreditkarteninformationen herausgegeben werden. Auch dieses Verhalten ist server- sowie benutzerseitig konfigurierbar.

USERDACCREQ (1Ch) User Data Access Request.

Als Reaktion auf eine Nachricht des Typs **DATAREADREQ** oder **DATAWRITEREQ**, in dessen Payload Lese- oder Schreiboperationen auf die personenbezogenen Daten eines Benutzers spezifiziert sind, sendet der Server diese Nachricht an den betroffenen Benutzer. Sollte die die Anforderung absendende Applikation generelles Lese- oder Schreibrecht haben, wird anstelle dieser Nachricht eine **USERDACCINFO**-Nachricht versendet.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem **userdaccrequest**-Tag.

USERDACCINFO (1Dh) User Data Access Info.

Als Reaktion auf eine Nachricht des Typs **DATAREADREQ** oder **DATAWRITEREQ**, wird vom CIDAS-Server, falls eine generelle Zugriffsfreigabe für die angeforderten Benutzerdaten vorliegt, diese Nachricht an den betroffenen Benutzer versendet.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit ei-

nem `userdaccinfo`-Tag.

USERDACCPerm (1Eh) User Data Access Permission.

Der Benutzer kann mittels einer **USERDACCPerm**-Nachricht der in einer **USERDACCReq**-Nachricht zugestellten Datenzugriffsanfrage zustimmen oder dem CIDAS-Server seine Ablehnung mitteilen.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `userdaccperm`-Tag.

Entsprechend Schäfer [93, S. 70] müssen personenbezogene Daten nicht zwangsläufig im Klartext auf dem CIDAS-Server gespeichert sein. Der CIDAS-Server muß daher unter bestimmten Umständen nachträglich symmetrisches Schlüsselmaterial oder die benötigten Daten selbst erfragen. Die zur Verschlüsselung der Benutzerdaten verwendeten kryptographischen Verfahren sind implementierungsabhängig.

USERDATAREQ (1Fh) User Data Request.

Müssen personenbezogene Daten erst vom Benutzer erfragt werden, wird diese Anfrage nach dem Erhalt der Zustimmung zur Datenweitergabe gesendet. Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `userdataacq`-Tag.

USERDATA (20h) User Data Request.

Der Client kann obige Anfrage mit einer **USERDATAREQ**-Nachricht beantworten.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem `userdataacq`-Tag.

Die übergebenen Daten werden anschließend vom Server validiert. Schlägt die Validierung fehl, kann die **USERDATAREQ**-Nachricht erneut gesendet werden. Nach einer konfigurierbaren Anzahl von fehlgeschlagenen Versuchen wird davon ausgegangen, daß der Benutzer die angeforderten Daten möglicherweise doch nicht weitergeben wollte.

DATARESP (21h) User Data Response.

Als Antwort auf eine Nachfrage der Typen **DATAREADREQ** oder **DATAWRITEREQ** wird diese Nachricht zurückgegeben.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem **dataresponse**-Tag.

Aufrechterhalten einer Verbindung, Sonstiges

KAREQ (22h) Keep Alive Request.

Nach Abschluß der Tokenzuweisung können Client und Server den Zustand der Verbindung prüfen, indem sie sich Lebenszeichen zusenden und beantworten. Ein Verbindungsteilnehmer darf maximal einen **KAREQ** alle 30 Sekunden versenden. Eine Verbindung gilt als gestört oder abgebrochen, falls ein Kommunikationspartner nicht innerhalb einer in der Serverkonfiguration einstellbaren Zeitspanne auf diese Anfrage antwortet.

Der Payload einer Nachricht dieses Typs ist leer.

KAANS (23h) Keep Alive Answer.

Antwort auf eine **KAREQ**-Nachricht.

Der Payload einer Nachricht dieses Typs ist leer.

INFOMSG (24h) Info Message.

Übertragung einer (informativen) Nachricht vom Server zum Client. Die Nachricht wird dem Benutzer vom Client angezeigt.

Der Payload einer Nachricht dieses Typs enthält eine XML-Struktur mit einem **infomsg**-Tag.

Verbindungsende

LOGOUT (25h) Logout.

Diese Nachricht wird vom Client an den Server gesendet. Sie führt zum Abbruch der Verbindung und zum Ungültigwerden aller Token des betreffenden Benutzers.

Der Payload einer Nachricht dieses Typs ist leer.

5.2.10 Beschreibung der verwendbaren XML-Tags

Nachrichten verschiedener Typen nutzen zur strukturierten internen Datenrepräsentation die Extensible Markup Language, kurz XML. XML ist ein offener und frei erweiterbarer Standard, wurde vom *World Wide Web Consortium* in [14] spezifiziert. In Zusammenhang mit diesem Dokument werden einige, speziell für den Datenaustausch zwischen CIDAS-Server und -Client konzipierte XML-Strukturen verwendet. Eine Data-Type-Definition ist am Ende dieser Arbeit in Anhang B verfügbar. An dieser Stelle seien lediglich die Grundzüge der Verwendung der dort definierten Strukturen an wenigen Beispielen beschrieben.

Der Root-Tag einer in einem Payload befindlichen XML-Struktur ist immer `<cidas>`. Dementsprechend endet jede Struktur mit `</cidas>`. Innerhalb des Root-Tags sind je nach Nachrichtentyp verschiedene XML-Elemente zulässig. Diese werden im Folgenden an Beispielen erläutert.

Darüber hinaus muß jeder Nachricht die verwendete XML-Version, die verwendete Zeichendarstellung und die der Nachricht zugrundeliegende Data-Type-Definition beigefügt sein. Dies geschieht in nachstehenden Code-Beispielen jeweils durch die ersten zwei Zeilen gemäß den diesbezüglichen Ausführungen in Bradley [41, S. 32 f., 36-100].

In den folgenden Beispielen werden drei Punkte, „...“, als Auslassungszeichen für bereits vorgestellte XML-Tags verwendet. Die Zeichenkette „...“ ist keinesfalls als Beispiel für den Inhalt des jeweiligen Tags zu verstehen.

infomsg Das `infomsg`-Element dient zur Übertragung von „menschenslesbaren“ Informationen vom Server an den Client bzw. dessen Benutzer. Die entsprechenden Nachrichten können multilingual verfaßt sein. `infomsg`-Tags werden an verschiedenen Stellen auch eingebettet in andere XML-Elemente verwendet. Ein Beispiel:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <infomsg>
    <infomsg_message xml:lang="en">
      Willkommen auf dem CIDAS-Server der Firma XYZ!
    </infomsg_message>
    <infomsg_message xml:lang="de">
      Welcome to the CIDAS server of company XYZ!
    </infomsg_message>
  </infomsg>
</cidas>
```

`idmethods` Größtmögliche Freiheit hinsichtlich der Konfiguration der auf seinem Server zulässigen Identifikationsmethoden hat ein CIDAS-Administrator dadurch, daß er beliebige Identifikationsmerkmale aus Abschnitt 5.2.3 durch boolesche Ausdrücke miteinander kombinieren und verschachteln kann. Grundsätzlich vorgesehen sind in der XML-Struktur die Verwendung von Benutzernamen, E-Mail-Adresse und Postanschrift. Es können darüber hinaus beliebige andere Daten durch Nutzung hierfür vorgesehener spezieller XML-Tags eingebunden werden. Folgendes Beispiel verwendet nur Benutzernamen, E-Mail-Adresse und Postanschrift.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <idmethods>
    <idmethods_xor>
      <idmethods_and>
        <idmethods_city></idmethods_city>
        <idmethods_zip></idmethods_zip>
        <idmethods_street></idmethods_street>
        <idmethods_realname></idmethods_realname>
      </idmethods_and>
      <idmethods_username></idmethods_username>
      <idmethods_email></idmethods_email>
    </idmethods_xor>
  </idmethods>
</cidas>
```

Die Struktur ist wie folgt zu interpretieren: Ein Benutzer kann sich identifizieren, indem er entweder seinen CIDAS-Benutzernamen oder seine bei CIDAS registrierte E-Mail-Adresse oder seinen Namen und seinen Wohnort angibt.

iddata Dieses Element wird im Payload einer Nachricht des Typs CIDAS_MTYPE_IDDATA verwendet und enthält eine Kombination von Identifikationsdaten, die obigem Beispiel für eine **idmethods**-Struktur entsprechend zulässig ist. Ein valider Payload wäre beispielsweise:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <iddata>
    <iddata_city>Brandenburg</iddata_city>
    <iddata_zip>14770</iddata_zip>
    <iddata_street>Magdeburger Strasse</iddata_street>
    <iddata_realname>Karl Mustermann</iddata_realname>
  </iddata>
</cidas>
```

Hierbei identifiziert sich Herr Karl Mustermann durch Angabe seiner nahezu vollständigen Postanschrift.

authmethod und **authmethods** Eine konkret zu verwendende Authentifizierungsmethode wird über den XML-Tag **authmethod** angegeben. Eine einfache Passwortabfrage kann wie im folgenden Beispiel dargestellt, spezifiziert werden.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <authmethod>
    <authmethod_class>TEXT</authmethod_class>
    <authmethod_name>PASSWORD</authmethod_name>
  </authmethod>
</cidas>
```

Sollen mehrere Methoden verwendet werden, können diese mittels des XML-Tags `authmethods` kombiniert werden. Die einzelnen hierbei angegebenen Authentifizierungsmethoden sind als und-verknüpft zu betrachten:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <authmethods>
    <authmethod>
      <authmethod_class>TEXT</authmethod_class>
      <authmethod_name>PASSWORD</authmethod_name>
    </authmethod>
    <authmethod>
      <authmethod_class>STORAGE_PASSIVE</authmethod_class>
      <authmethod_name>OPGP</authmethod_name>
    </authmethod>
  </authmethods>
</cidas>
```

`authdatareq` Die folgende Struktur spezifiziert vom Benutzer einzugebende oder zu erzeugende Daten näher und stellt damit die Authentifizierungsdaten-anfrage dar. Beispielsweise für kryptographische oder biometrische Authentifizierungsverfahren kann die XML-Struktur Radix-64-kodierte Binärdaten aufnehmen. Hierfür ist der `authdatareq_bytestream`-Tag zu verwenden. In einem `authdatareq` enthaltene `infomsg`-Tags sind als direkte Frage oder Handlungsaufforderung an den Benutzer zu verstehen. In einem `authdatareq`-Tag werden immer nur die im Rahmen einer Authentifizierungsmethode benötigten Daten abgefragt. Authentifizierungsmethoden, die das mehrmalige Übersenden von Anfragen und Antworten erfordern, werden über mehrere Nachrichten mit jeweils einem `authdatareq`-Tag realisiert.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <authdatareq>
    <authmethod>
      <authmethod_class>TEXT</authmethod_class>
      <authmethod_method>RANDDATA</authmethod_method>
    </authmethod>
  </authdatareq>
</cidas>
```

```

<infomsg>
  <infomsg_message xml:lang="en">
    What is your mother's maiden-name?
  </infomsg_message>
  <infomsg_message xml:lang="de">
    Wie lautet der M&auml;dchenname ihrer Mutter?
  </infomsg_message>
</infomsg>
</authdatareq>
</cidas>

```

authdata Als Antwort auf einen **authdatareq** wird eine Nachricht mit einem **authdata**-Tag erzeugt. In dieser Nachricht werden textuelle Antworten (monolingual) getrennt von Radix-64 kodierten Binärdaten gespeichert. Siehe hierzu Anhang B.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <authdata>
    <authmethod>
      <authmethod_class>TEXT</authmethod_class>
      <authmethod_name>RANDDATA</authmethod_name>
    </authmethod>
    <authdata_text xml:lang="de">
      G&auml;rtner
    </authdata_text>
  </authdata>
</cidas>

```

token und **csid** Die Verwendung von Token ebenso wie die Bedeutung der in der folgenden XML-Struktur enthaltenen Daten sind primär in Abschnitt 5.2.6 spezifiziert. Folgende Zeiten (Zeitangabe entsprechend RFC 822 [47, S. 26 f.]) sind im folgenden Beispiel gesetzt:

```

token_time-issue : Tue, 3 Feb 2004 19:24:13 +0100
token_time-valid : Tue, 3 Feb 2004 19:32:56 +0100

```

Das Token hat also eine Gültigkeit von insgesamt 8 min und 43 s. Bei dem Inhalt des **token_tokendata** handelt es sich um 128 Bytes Radix-64-kodierte

Daten. Das Verfahren ist in RFC 2440 [43, S. 7, 41-48] beschrieben.

Im Beispiel wird gleichzeitig die `csid`-Struktur als Bestandteil des Tokens eingeführt. Dieser „CIDAS Server Identifier“ enthält den Hostname, die unterstützten Protokollversionen und optional die verwendeten Ports des CIDAS-Servers. Der `csid`-Eintrag ermöglicht damit die Zuordnung eines Tokens zu einem CIDAS-Server.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <token>
    <token_time-issue>1075832653</token_time-issue>
    <token_time-valid>1075833176</token_time-valid>
    <token_tokendata>
      S1wXbFfdGieGFaw0+R3XC1/HASgBJ0SGxW0Ads/Ikc9/milrfAV1EZfVxmiy+AQH
      +v4RLeUt0TEDkhFNuF9bGckeikBkvYwP9kq0rwyvsBSMiNhVyo41138GyrwfnAHat
      S/CjeBChq6XREZqJwXx+IfA2TprYYPbE9HAylE+x4Lg=
      +=aMo
    </token_tokendata>
    <csid>
      <csid_hostname>cidas.cidas.org</csid_hostname>
      <csid_pversions>
        <csid_pversion>1.0</csid_pversion>
      </csid_pversions>
    </csid>
  </token>
</cidas>
```

`token_ver_record` Der „Token Verification Record“ enthält neben einem Token auch noch einen `infomsg`-Tag. Dieser dient als Möglichkeit, den Grund einer Token-Überprüfung durch eine Applikation zu spezifizieren.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <token_ver_record>
    <token>
      ...
    </token>
```

```

<infomsg>
  <infomsg_message xml:lang="en">
    You signed in for the Cryptography-Course! We need to
    check whether you are authenticated.
  </infomsg_message>
</infomsg>
</token_ver_record>
</cidas>

```

`token_peer_record` Bei der Überprüfung eines Tokens wird der `token_peer_record`-Tag verwendet:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <token_peer_record>
    <token>
      ...
    </token>
    <service>
      <service_name>
        University Information System
      </service_name>
      <service_provider>
        University of Applied Sciences, Brandenburg
      </service_provider>
      <service_url>
        https://uis.fh-brandenburg.de
      </service_url>
    </service>
  </token_peer_record>
</cidas>

```

Die weiteren XML-Strukturen zum Token-Austausch und zur Tokenbehandlung sind analog zu den vorgestellten zu bedienen. Der innerhalb des `token_ver_succ`-Tags verwendete `guid`-Tag wird bei erfolgreicher Verifikation eines Tokens an die Applikation gesendet. Über den `guid`-Tag kann eine Applikation einen Benutzer wiedererkennen: Jeder Benutzer hat entsprechend

Schäfer [93, S. 78-84] für jede Applikation einen eindeutigen, von CIDAS gepflegten „Global Unique User Identifier“.

capability, capdata und sfmdata Mittels des **capability**-Tags können zusätzliche Funktionen eines CIDAS-Servers, beispielsweise die Funktionalität installierter Special Feature Module, inklusive deren Schnittstellen beschrieben werden. Für die Schnittstellenspezifikation wird SOAP entsprechend dem *World Wide Web Consortium* [28] verwendet. Da derzeit noch keine weiterführende konzeptionelle Dokumentation zur Einbindung von Special Feature Modulen in CIDAS bzw. zu deren Schnittstellen existiert, kann an dieser Stelle kein Beispiel für eine vollständige Funktionalitätsbeschreibung gegeben werden.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <capability>
    <capability_name>
      Notenübersicht
    </capability_name>
    <capability_vendor>
      University Information System,
      University of Applied Sciences, Brandenburg
    </capability_vendor>
    <capability_version>1.5</capability_version>
  </capability>
</cidas>
```

Ein **capdata**-Tag enthält eine beliebige Anzahl von **capability**-Tags mit entsprechenden Inhalten.

Über Schnittstellenbeschreibungen hinausgehende Kommunikation zwischen Special Feature Modulen und dem CIDAS Client, also die tatsächliche Benutzung eines solchen Moduls, wird über XML-Strukturen mit einem **sfmdata**-Tag realisiert. In diesen ist das jeweils verwendete Special Feature Modul benannt. Darüber hinaus enthält der Tag eine SOAP-Struktur, die Eingabeparameter oder Rückgaben des Moduls aufnehmen kann.

serverrecord Der **serverrecord**-Tag wird verwendet, um weitergehende Informationen über einen CIDAS-Server und diesen verwendende Applikationen zu erhalten. Die Detailliertheit der weitergegebenen Informationen soll konfigurierbar sein. Insbesondere die Auflistung der den Server verwendenden Applikationen sollte im Ermessen der Applikationsbetreiber liegen.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <serverrecord>
    <csid>
      ...
    </csid>
    <serverrecord_pubkey>
      ...
    </serverrecord_pubkey>
    <serverrecord_product>
      CIDAS Server Reference Implementation
    </serverrecord_product>
    <serverrecord_vendor>
      University of Applied Sciences, Brandenburg
    </serverrecord_vendor>
    <serverrecord_version>0.2</serverrecord_version>
    <serverrecord_provider>
      University of Applied Sciences, Brandenburg
    </serverrecord_provider>
    <serverrecord_services>
      <service>
        ...
      </service>
    </serverrecord_services>
  </serverrecord>
</cidas>
```

datareadrequest, **datawriterequest** und **dataresponse** Diese drei XML-Tags haben alle einen sehr ähnlichen Aufbau, untenstehendes Beispiel zeigt lediglich die einfachste der drei Strukturen. Alle spezifizieren eine Menge von Attributen aus dem personen- oder applikationsbezogenen Datenbestand eines CIDAS-Benutzers, im Folgenden sind das beispielsweise Teile der Post-

anschrift sowie der letzte Anmelde-Zeitpunkt eines durch ein Token bezeichneten Benutzers. Der `infomsg`-Tag kann hierbei Hinweise für den jeweiligen Benutzer aufnehmen.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE cidas SYSTEM "http://cidas.org/xml/cidas_1-0.dtd">
<cidas>
  <datareadrequest>
    <token>
      ...
    </token>
    <userdata>
      <data_attribute>
        <data_attribute_name>city</data_attribute_name>
      </data_attribute>
      <data_attribute>
        <data_attribute_name>state</data_attribute_name>
      </data_attribute>
    </userdata>
    <appdata>
      <data_attribute>
        <data_attribute_name>last_login</data_attribute_name>
      </data_attribute>
    </appdata>
    <infomsg>
      ...
    </infomsg>
  </datareadrequest>
</cidas>
```

Die Antwort des CIDAS-Servers auf Datenanfragen wird in einer `dataresponse`-Struktur übertragen. Dabei enthalten die `data_attribute`-Tags jeweils einen `data_attribute_value`-Tag mit dem entsprechenden Wert des Attributs. Gleiches gilt für `datawriterequest`-Strukturen, der enthaltene Wert ist hierbei der in den Datenbestand des CIDAS-Servers einzutragende. Mit der Rückgabe wird in diesem Fall nochmal bestätigt, welche Attribute geschrieben wurden.

`userdaccrequest` **und** `userdaccinfo` In Abhängigkeit davon, ob die jeweilige Applikation Benutzerdaten grundsätzlich schreiben oder lesen darf,

werden Nachrichten mit den XML-Elementen `userdaccrequest` oder `userdaccinfo` an den Client gesendet. Es werden jeweils ein die Applikation identifizierender `server-` oder `user-`Tag, die jeweils angeforderten Daten sowie die von der Applikation übertragene `infomsg`.

`userdaccperm` Der Inhalt dieses Tags ist dem des `userdaccrequest`-Tags sehr ähnlich. Es wird spezifiziert, welchen Bestandteilen einer Lese- oder Schreibanfrage auf die personenbezogenen Daten eines Benutzers dieser zustimmt. Als Schlüsselworte im `data_attribute_value`-Tag können hierbei „ALWAYS“, „NEVER“ und „YES“ verwendet werden. „YES“ hat einmalig, für die aktuelle Transaktion, Gültigkeit. „ALWAYS“ und „NEVER“ haben ab dem aktuellen Zeitpunkt bis auf Widerruf durch den Benutzer immer Gültigkeit, die Einstellungen werden im Server vermerkt. Die Ablehnung ist implizit gegeben, wenn ein angefragtes Datum nicht aufgeführt wird.

`userdataacq` Falls Daten an eine Applikation übergeben werden sollen, die im Datenbestand des CIDAS-Servers nur symmetrisch verschlüsselt vorhanden sind, wobei der Schlüssel lediglich dem Benutzer bekannt ist oder, falls der Server lediglich über Metadaten⁹ über den zu angefragten Daten verfügt, müssen Schlüsselmaterial oder die Daten selbst angefordert werden. Anforderung und Antwort werden in einer `userdataacq`-Struktur übergeben, die einzelnen Tags enthalten für die Anfrage keinen Inhalt. Falls ein Schlüssel zu übergeben ist, wird dieser in Form von Text, quasi als Passwort, angegeben.

⁹Es werden hierfür HASH-Funktionen, vgl. Schneier, [96, S. 30 f.], verwendet.

Kapitel 6

Prototypische Umsetzung des Authentifizierungsverfahrens und des Kommunikations- protokolls

Allein der Rückgriff auf einen Computer zur Lösung eines Problems verringert schon die Chance, andere Lösungen zu erkennen. Wenn das einzige Werkzeug, das Sie kennen, ein Hammer ist, sieht alles wie ein Nagel aus.
— Clifford Stoll, *Die Wüste Internet* ([102, S. 76])

Prototypische Umsetzung des Authentifizierungsverfahrens und des Kommunikationsprotokolls

In diesem Kapitel erläutert der Autor sein Vorgehen bei der prototypischen Umsetzung eines CIDAS-Clients. Eingegangen wird insbesondere auf das Grundgerüst des Clients und auf in diesem Zusammenhang gefällte Designentscheidungen. Auch wird die ansatzweise Implementierung des CIDAS-Protokolls erläutert. Zur Umsetzung des in Kapitel 4 spezifizierten Authentifizierungsprotokolls werden verschiedene, dem Autor als wichtig erscheinende Hinweise gegeben. Eine tatsächliche Implementierung des Verfahrens wird nicht durchgeführt. Hierzu wäre ein funktionstüchtiger CIDAS-Server nötig gewesen.

Insbesondere das Fehlen des CIDAS-Servers – dieser war Gegenstand einer im August 2003 an der Fachhochschule Brandenburg eingereichten Diplomarbeit ([93]) und sollte dem Autor laut Aussage der Projektleitung zur Bearbeitung dieses Kapitels zur Verfügung stehen – ist ein primärer Grund für die Unvollständigkeit der prototypischen Umsetzung der in der vorliegenden Arbeit konzipierten Verfahren.

Bezüglich des im Rahmen der Bearbeitung dieses Kapitels geschriebenen Quellcodes sei auf die der vorliegenden Arbeit beiliegende CD-ROM verwiesen. Sie enthält einen vollständigen Auszug des insgesamt zu CIDAS existierenden Quellcodes, sowohl L^AT_EX-Dokumentation als auch Programmcode, und aller digital verfügbaren literarischen Quellen zum Zeitpunkt der Abgabe der Arbeit. Der Quellcode der CIDAS-Client-Implementierung befindet sich im Verzeichnis `/src/cidas/src/client/`.

6.1 Designentscheidungen

Zur Bearbeitung der prototypischen Realisierung eines CIDAS-Clients wurden insbesondere die im Folgenden dargestellten Designentscheidungen getroffen.

Betriebssystem Um von Anfang an eine möglichst große Palette von aktuellen Betriebssystemen abdecken zu können entschied sich der Autor primär für die Unterstützung von Betriebssystemen, die den POSIX-Standard entsprechend IEEE 1003-1 [4] unterstützen. Es wird partiell die in den Erweiterungen zu IEEE 1003-1 ([7], [9] und [11]) spezifizierte Funktionalität genutzt. Der CIDAS-Client sollte damit unter allen modernen UNIX¹-ähnlichen Betriebssystemen lauffähig sein. Insbesondere die Verfügbarkeit der Standards und der Quellen zu den o.g. Betriebssystemen trugen zu dieser Entscheidung bei. Konkret wurde die Entwicklung vorrangig unter SunOS 5.8² und Linux 2.4.x³ durchgeführt. Als Hardware-Plattformen wurden Sun-SPARC⁴-, MIPS⁵- und Intel⁶-kompatible Systeme verwendet.

Programmiersprache Um Zugriffe auf die Systemhardware einfach zu gestalten und direkte Authentifizierungsleistungen, beispielsweise Workstation-Logins, für die o.g. Betriebssysteme erbringen zu können, wurde die native Programmierschnittstelle von SunOS 5.8 bzw. Linux 2.4.x verwendet. Diese ist entsprechend dem POSIX-Standard für die Programmiersprache C nach Kernighan et al. [69, S. 183-238] spezifiziert. Ein weiterer Vorteil der Verwendung von C resultiert daraus, daß viele Netzwerkdienste, für die CIDAS Authentifizierungsleistungen erbringen könnte, in C entwickelt wurden. Hier-

¹Der Begriff UNIX bezeichnet eine weit verbreitete Betriebssystemfamilie mit derzeit unklaren Eigentumsrechten, entsprechende Spezifikationen finden sich im Internet unter <http://www.unix.org>.

²Siehe hierzu <http://www.sun.com>.

³Informationen zu Linux finden sich unter <http://www.kernel.org>.

⁴Dokumentation zu dieser Prozessorfamilie ist unter <http://www.sun.com> zu finden.

⁵Siehe <http://www.mips.com>.

⁶Siehe <http://www.intel.com>.

zu gehört beispielsweise der derzeit erfolgreichste Webserver *Apache*⁷ mit seinen vielen Erweiterungen (vgl. Wheeler, [108]). Um diese Dienste an CIDAS anzubieten, müssen sie häufig um einige Funktionen erweitert werden, was in der Regel nur in der Sprache geschehen kann, in der die Software entwickelt wurde. Auch erlaubt die Sprache C die Nutzung verschiedener Betriebssystem-Funktionen zur Speicherverwaltung, die Voraussetzung für die Implementierung sicherer kryptographischer Verfahren sind und aus anderen Sprachen häufig nicht zur Verfügung stehen.

Konkret wurden zur prototypischen Umsetzung des CIDAS-Clients neben der Standard-C-Bibliothek der verwendeten Betriebssysteme die *GNU Compiler Collection*⁸ in der Version 2.95.4, die *glib*⁹ in der Version 2.0.1 sowie zur Erzeugung und Verarbeitung der XML-Strukturen die *libxml*¹⁰ in der Version 2.4.19 verwendet.

Entwicklungsmethodik Als grundsätzliche Entwicklungsmethodik wird vom Autor das *Extreme Programming* entsprechend Beck (vgl. Beck, [37]) bevorzugt. Der Autor hat mit dieser Entwicklungsmethode bereits in früheren Projekten sehr gute Erfahrungen gemacht. Laut Beck ([37, S. 2-9]) steht beim *Extreme Programming* die Programmierung von minimal vollständigem Quellcode während des gesamten Entwicklungsprozesses im Vordergrund. Die Programmierung wird dabei in der Regel paarweise, also jeweils zwei Programmierer an einem Computer, durchgeführt und ist testorientiert: Die Tests werden vor der Entwicklung des Anwendungscodes geschrieben, die Anwendung wird soweit verbessert, bis sie alle Tests erfüllt. Das Design der Software wird beim *Extreme Programming* ebenfalls in die Entwicklungsphase gelegt, die Projektdokumentation entsteht mit dem bzw. im Quellcode und ist damit für nachfolgende Entwickler und auch zur externen Analyse des Co-

⁷Informationen zu diesem Open-Source-Projekt sind unter <http://www.apache.org> erhältlich.

⁸Die Software ist im Internet unter <http://www.gnu.org> erhältlich.

⁹Die *glib* ist eine portable Funktionsbibliothek, Teil des *GTK*-Projektes und verfügbar unter <http://www.gtk.org>.

¹⁰Diese Bibliothek ist Teil des *GNOME*-Projekts und unter <http://xmlsoft.org/> verfügbar.

des jederzeit verfügbar. Resultierend daraus existiert jedoch keine separate Quellcode-Dokumentation in Form von Struktogrammen und Programmablaufplänen.

Die Methode des *Extreme Programming* bietet sich bei der Entwicklung von CIDAS insbesondere deshalb an, weil das Grundkonzept des CIDAS-Servers von einem anderen Mitarbeiter erstellt wurde, als das in dieser Arbeit spezifizierte Authentifizierungsverfahren und das Kommunikationsprotokoll, wodurch eine Zusammenarbeit bei der Implementierung erforderlich war. Auch enthalten die client- und die serverseitige Implementierung der Verfahren mehrere ähnliche oder gleiche Komponenten. Beim der paarweisen Entwicklung dieser Module können daher Redundanzen und Programmierfehler vermieden werden.

Durch vom Autor nicht zu vertretende Umstände, auf die hier auch nicht näher eingegangen wird, kam es nie zum paarweisen von Client- und Server-Software. Zum Zeitpunkt der Abgabe dieser Arbeit sind beide noch überaus unvollständig.

Programmaufbau Der CIDAS-Client wurde vom Autor als eine sehr modular aufgebaute Software konzipiert, deren Einzelteile möglichst unkompliziert erweiterbar und wiederverwendbar sein sollten. Neben einem kompakten Grundprogramm besteht der CIDAS-Client daher aus einer Reihe von dynamisch, zur Laufzeit des Grundprogramms ladbaren Teilkomponenten, in denen die eigentliche Funktionalität implementiert ist. Es existieren verschiedene Module zur Behandlung der Identifizierung, der Authentifizierung und der Kommunikation. Diese werden jeweils von einem Handler-Modul angesprochen. Auch die Benutzungsoberfläche ist in ein eigenständiges Modul ausgegliedert und kann dementsprechend jederzeit ersetzt werden. Zur Dokumentation der einzelnen Module sei, getreu den Grundmotiven des *Extreme Programming*, auf den Quellcode auf der beiliegenden CD-ROM verwiesen.

Die Benutzungsschnittstelle Die vom Autor im Rahmen der prototypischen Umsetzung implementierte Benutzungsschnittstelle des CIDAS-Clients ist befehlsorientiert und dementsprechend verhältnismäßig spartanisch. Sie verfügt über eine rudimentäre Hilfefunktion, die über das Kommando `help`¹¹ erreichbar ist. Jedoch genügt sie aus Sicht des Autors völlig, um die Grundfunktionalität der Software zu benutzen. Komfortablere Benutzungsoberflächen können durch den modularen Gesamtaufbau der Client-Software jederzeit entwickelt und nachgeladen werden.

¹¹Die Kommandoingabe muß durch einen Druck auf die Eingabetaste abgeschlossen werden.

6.2 Prototypische Umsetzung des Kommunikationsprotokolls

Im Rahmen der prototypischen Umsetzung des in Kapitel 5 spezifizierten Kommunikationsprotokolls wurde lediglich die auf der Übertragung unverschlüsselter TCP-Pakete (vgl. RFC 793, [2]) basierende und als „CIDAS-Plain“ bezeichnete Variante des Protokolls implementiert. Dies bringt insbesondere während der Entwicklung den Vorteil der einfachen Überprüfbarkeit des Datenstroms mit sich, was bei einer verschlüsselten Datenübertragung über „CIDAS-Encrypted“ nicht so einfach möglich ist. Auch läßt sich eine Implementierung von „CIDAS-Plain“ nachträglich verhältnismäßig einfach, beispielsweise durch die Benutzung der *OpenSSL*-Bibliothek¹², um die Verwendung kryptographischer Verfahren zur Sicherung der Datenübertragung erweitern.

Die eigentliche Funktionalität des Protokolls wurde vom Autor in Form eines dynamisch ladbaren Moduls entwickelt, das eine vom CIDAS-Client nutzbare, prozedurale Schnittstelle bereitstellt. Dieses Vorgehen erlaubt es prinzipiell, mehrere Module, die eine ähnliche Funktionalität bieten – beispielsweise verschiedene (zukünftige) Versionen des CIDAS-Protokolls gleichzeitig zu verwenden. Die Funktionsweise der internen Schnittstellen ist im Quellcode dokumentiert.

Die derzeitige Implementierung unterstützt lediglich die Kommunikation mit einem einzelnen CIDAS-Server. Auch ist das Kommunikationsprotokoll nur unvollständig implementiert. Die prototypische Umsetzung sollte jedoch, so denn ein CIDAS-Server vorhanden wäre, in der Lage sein, den Verbindungsaufbau, die Identifizierung und die Authentifizierung eines Benutzers unter Verwendung textueller Authentifizierungsmethoden abzuhandeln.

¹²*OpenSSL* ist eine freie Implementierung von SSL und TLS entsprechend Netscape Inc. ([55]) und RFC 2246 ([50]). Das Projekt ist unter <http://www.openssl.org> verfügbar.

Die Protokollimplementierung wurde mittels mehrerer manuell erstellter Test-Nachrichten während der Entwicklungsperiode überprüft, der Client verfügt hierfür über eine umfassende Logging-Funktionalität. Mitschnitte verschiedener Testläufe sind auf der beiliegenden CD-ROM im Verzeichnis `/src/cidas/src/client/common/tests` enthalten. Umfassende Integrationstests konnten aufgrund des Fehlens eines CIDAS-Servers nicht durchgeführt werden.

6.3 Implementierungsansatz für das Authentifizierungsprotokoll

Nachdem in Kapitel 4 das Verfahren zur Verifikation der Identität eines Benutzers im Allgemeinen und für verschiedene Formen identifizierbarer und nichtidentifizierbarer Datenträger erläutert wurde, sollen im Folgenden Einzelheiten bezüglich einer Implementierung des Authentifizierungssystems vorgestellt werden. Dies betrifft insbesondere die Art der Speicherung von Daten auf Authentifizierungsmedien, Details zum Umgang mit diesen Daten sowie die Länge von Zufallsdaten und Schlüsselmaterial.

6.3.1 Datenspeicherung auf Authentifizierungsmedien

Von zu unterschätzender Wichtigkeit für das Funktionieren der Authentifizierung ist die Anordnung der Daten auf den jeweils verwendeten Authentifizierungsmedien. Damit CIDAS-Clients diese finden und korrekt interpretieren können, müssen diese an einer spezifizierten Stelle und in einem spezifizierten Format vorliegen. Die Festlegung geschieht hierbei in Abhängigkeit von der Art des Datenträgers. In jedem Fall handelt es sich hierbei größtenteils um individuelle Entscheidungen des Autors, die potentiell auch anders hätten ausfallen können.

Datenträger mit Dateisystem Auf verschiedenen, möglicherweise im Zusammenhang mit CIDAS und dem in dieser Arbeit vorgestellten Authentifizierungsverfahren einsetzbaren Datenträgern, beispielsweise auf CD-ROMs, USB-Memory-Sticks oder auch Disketten und Festplatten existiert üblicherweise ein Dateisystem, über das eine CIDAS-Client-Implementierung auf benötigte Daten zugreifen kann. Die Daten sind hierbei in Form von Verzeichnissen und Dateien organisiert. Häufig können diese Datenträger auch nochmals in verschiedene logische Bereiche, sogenannte Partitionen unterteilt werden.

Der Einfachheit halber wird im Folgenden eine UNIX-typische Notation verwendet. Dabei entspricht / dem Wurzelverzeichnis der Partition des

Authentifizierungsmediums, in dem sich die für CIDAS relevanten Daten befinden. Weitere /-Zeichen in einer Pfadangabe bezeichnen einen Wechsel in ein Unterverzeichnis. Beispielsweise bezeichnet eine Pfadangabe der Art `/daten/schluessel/geheimnis` eine Datei namens `geheimnis`, die sich im Unterverzeichnis `schluessel` befindet, welches wiederum ein Unterverzeichnis von `daten` ist. Aus dem / am Anfang des Pfades läßt sich ableiten, daß sich `daten` im Wurzelverzeichnis einer Partition befindet.

Für Datenträger mit einem Dateisystem sollten sich die für die Authentifizierung relevanten Daten auf der ersten Partition des jeweiligen Mediums befinden. Sollte ein CIDAS-Client dort keine verwendbaren Daten vorfinden oder die Auswahl zwischen mehreren Datenträgern mit Authentifizierungsdaten haben, ist eine Rückfrage beim Benutzer erforderlich.

Auf der verwendeten Partition des Mediums werden die Authentifizierungsdaten in einem Verzeichnis namens `.cidas` direkt unterhalb des Wurzelverzeichnisses gespeichert. Der Verzeichnisname läßt sich auf eine für Applikationen unter UNIX-ähnlichen Betriebssystemen geltende „Tradition“ zur Speicherung applikationsbezogener Daten zurückführen. Dateien oder Verzeichnisse, die mit einem `.` beginnen, gelten hier auch als „versteckt“.

Unterhalb dieses Verzeichnisses können nun Konfigurationsinformationen für den CIDAS-Client ebenso wie Schlüsselmaterial und personenbezogene Daten abgelegt werden. Konkret sei folgende Struktur vorgeschlagen, Aufbau und Bedeutung der einzelnen Dateien werden im folgenden nochmals näher erläutert:

```
/.cidas                -- das Verzeichnis mit allen
                        benoetigten Authentifizierungsdaten
/.cidas/cidas.conf     -- die primaere Konfigurationsdatei fuer
                        CIDAS-Clients, u. U. verschluesselt
/.cidas/medium.id     -- Identifizierungsdaten des Mediums
/.cidas/openpgp        -- ein Verzeichnis fuer OpenPGP-Schlues-
                        selmaterial
/.cidas/openpgp/pubring.asc -- oeffentliches Schluesselmaterial
```

```
(OpenPGP)
/.cidas/openpgp/secring.asc -- privates Schluesselmaterial (OpenPGP)
/.cidas/user                -- ein Verzeichnis fuer die personen-
                             bezogenen Daten des Benutzers
/.cidas/user/*              -- moeglicherweise mehrere ver-
                             schluesselte Dateien
/.cidas/x.509               -- ein Verzeichnis fuer X.509-
                             Zertifikate
/.cidas/x.509/*             -- moeglicherweise mehrere Dateien
                             mit Schluesselmaterial (X.509)
```

`cidas.conf` Die Datei `cidas.conf` ist als allgemeine Konfigurationsdatei des CIDAS-Clients vorgesehen und wird möglicherweise sicherheitsrelevante Daten wie die Informationen über die verwendeten CIDAS-Server, persönliche Einstellungen oder gar Identifizierungsdaten des Benutzers enthalten. Die Datei sollte aus diesem Grund nach Möglichkeit verschlüsselt sein. Die Art des zu verwendenden Schlüsselmaterials zum Entschlüsseln der Datei muß eine Client-Implementierung aus dem jeweiligen Aufbau der Datei erkennen. Dabei bestehen die auf Seite 138 in Tabelle 6.1 aufgelisteten Möglichkeiten. Ist die Datei verschlüsselt gespeichert, muß vom Benutzer üblicherweise ein Passwort, entweder zur Entschlüsselung asymmetrischen Schlüsselmaterials oder als direkt anzuwendender symmetrischer Schlüssel für die Datei, abgefragt werden. Wird ein asymmetrisches Verfahren verwendet, muß das notwendige Schlüsselmaterial ebenfalls bereitgestellt werden. CIDAS-Clients sollten hiernach selbstständig in `./cidas/openpgp` bzw. `./cidas/x.509` suchen. Die vollständige Formulierung einer `cidas.conf` ist nicht Gegenstand dieser Arbeit.

`medium.id` Diese Datei enthält die das Authentifizierungsmedium identifizierende Signatur über nichtmanipulierbaren Daten des Mediums. Ähnlich wie bei der Datei `cidas.conf` müssen Authentifizierungsclients die Art der Signatur identifizieren und entsprechend auswerten. Sollte sich die Signatur als ungültig herausstellen, darf der Authentifizierungsvorgang nicht fortgesetzt werden. Auf identifizierbaren Medien muß die Datei existieren, andernfalls darf sie nicht vorhanden sein.

Dateiaufbau	Dateityp
Datei liegt im Klartext vor	unverschlüsselt
Datei entspricht RFC 2633 ([87])	verschlüsselt, es muß ein X.509-Schlüssel verwendet werden
Datei entspricht RFC 2440 ([43, S. 39])	verschlüsselt, es muß ein direkt einzugebender symmetrischer Schlüssel verwendet werden
Datei entspricht RFC 2440 ([43, S. 37-39])	verschlüsselt, es muß ein asymmetrischer OpenPGP-Schlüssel verwendet werden

Tabelle 6.1: Möglichkeiten der Speicherung der Datei `cidas.conf`

`openpgp/pubring.asc` In dieser Datei wird, falls OpenPGP-Schlüssel verwendet werden, das zur Authentifizierung benötigte öffentliche Schlüsselmaterial wie in RFC 2440 [43, S. 34, 35-37] beschrieben, gespeichert. Die Datei kann mehrere öffentliche Schlüssel und falls nötig zugehörige Signaturen enthalten. Alle Schlüssel und Signaturen in dieser Datei sind in ASCII-Armored, Radix-64-kodierter Form entsprechend RFC 2440 [43, S. 7, 41-48] abzuspeichern.

`openpgp/secring.asc` Die Datei `openpgp/secring.asc` ist zur Speicherung des privaten Schlüsselmaterials bei Verwendung von OpenPGP-Schlüsseln vorgesehen. Der Aufbau der Schlüsselpakete wird in RFC 2440 [43, S. 34-35, 37-38], beschrieben. Auch bei dieser Datei ist die Radix-64-Kodierung erforderlich.

`user/*` In `./cidas/user/` können personenbezogene Daten des Benutzers in einer oder mehreren Dateien abgelegt werden. Die Daten in diesem Verzeichnis sollten grundsätzlich verschlüsselt sein. Die Spezifikation weiterer Dateinamen oder konkreter Dateiinhalte in dieser Unterhierarchie von `./cidas` ist nicht Gegenstand dieser Arbeit.

`x.509/*` Das Verzeichnis `./cidas/x.509/` dient zur Aufnahme von X.509-konformem Schlüsselmaterial zur Authentifizierung. Der Aufbau von

X.509 Zertifikaten wird in [44]¹³ bzw. [65], Abschnitt 4: „Certificate and Certificate Extensions Profile“ oder [68], Abschnitt 3: „Architecture“ beschrieben. Das Schlüsselmaterial kann hier in mehreren Dateien abgelegt werden. Authentifizierungsclients müssen diese im Bedarfsfall nach dem benötigten Zertifikat durchsuchen.

Verschlüsselte Dateisysteme Obiger Ansatz zum Schutz relevanter bzw. personenbezogener Daten basiert auf der Anwendung kryptographischer Verfahren auf einzelne Dateien. Die Verwendung ganzer verschlüsselter Dateisysteme, bei denen Ver- und Entschlüsselungsfunktionen bereits in der Steuerungssoftware des Gerätes oder im Betriebssystem implementiert sind, wird in [96, S. 220-223] unter dem Begriff „Driver-Level Encryption“ der „File-Level Encryption“ gegenüber gestellt. Die Verschlüsselung des Dateisystems oder zumindest des `/.cidas`-Verzeichnisses würde der Implementierung eines CIDAS-Clients die Möglichkeit geben, auch temporäre Arbeitsdateien sehr unkompliziert, aber dennoch sicher zu speichern. Darüber hinaus könnten das geheime Schlüsselmaterial sowie die personenbezogenen Daten ohne explizite Verschlüsselung gespeichert werden, da diese auf einer anderen Ebene außerhalb des CIDAS-Clients realisiert werden würde.

Leider gibt es derzeit keinen dem Autor bekannten und allgemein verbreiteten Standard, der Driver-Level Encryption plattformübergreifend realisiert. Der Einsatz solcher Technologien würde den Benutzer zu sehr an eine bestimmte Betriebssystemfamilie binden und ihm dementsprechend vieles von der Flexibilität wieder nehmen, die CIDAS bieten kann. Aus diesem Grund wird vom Einsatz verschlüsselter Dateisysteme derzeit abgesehen.

Datenträger ohne Dateisystem Für Datenträger, auf denen aus technischen Gründen oder durch Einschränkungen in der zur Verfügung stehenden Speicherkapazität kein Dateisystem verwendet wird, müssen das zur Authen-

¹³X.509 ([44]) lag dem Autor aus finanziellen Gründen nicht während der gesamten Bearbeitungszeit vor, die RFCs 1422 ([68]) und 2459 ([65]) entsprechen jedoch den nach X.509 geltenden Normen.

tifizierung verwendete Schlüsselmaterial von den zur Identifizierung des Datenträgers verwendeten Daten auf geeignete Weise voneinander unterscheidbar abgelegt sein. Als Datenträger ohne Dateisystem werden derzeit lediglich RFID-Geräte unterstützt.

Wie bereits in Abschnitt 4.2.4 angeführt, preferiert der Autor bei der Verwendung von Authentifizierungsmedien mit sehr geringem Speicherplatz die Nutzung des, verglichen mit X.509 Zertifikaten, geringfügig kompakteren OpenPGP-Schlüsselmaterials. Auch ist zu berücksichtigen, daß durch die Möglichkeit, einen öffentlichen Schlüssel mehrfach zu unterschreiben, eine bereits mehrfach angedachte¹⁴ Erweiterung von CIDAS umgesetzt werden könnte: Die Implementierung von Sicherungsmechanismen für Verarbeitungsketten von Produkten oder auch Abfertigungskontrollen für Fahr- und Fluggäste.

In welcher Reihenfolge die zur Datenträgeridentifizierung verwendete Signatur und das Schlüsselmaterial auf dem Datenträger gespeichert werden ist unerheblich, weil sich die entsprechenden Daten wie in X.509 bzw. RFC 2440 spezifiziert voneinander unterscheiden lassen.

6.3.2 Erzeugung und Größe von Zufallsdaten und Zeitstempeln

Das in Kapitel 4 spezifizierte Authentifizierungsverfahren verwendet mehrfach Zeitstempel und Zufallsdaten. Auf deren Erzeugung und Größe wird in diesem Abschnitt eingegangen.

Zufallsdaten Für die Sicherheit der verwendeten kryptographischen Verfahren, insbesondere bei der Schlüsselerzeugung, für das in Kapitel 4 erläuterte Authentifizierungsverfahren, aber auch im Rahmen des CIDAS-Protokolls sind mehrfach qualitativ hochwertige, von Dritten nicht nachvollziehbare oder vorhersehbare Zufallsdaten nötig. Zur Erzeugung dieser Daten stellen UNIX-

¹⁴Der Autor bezieht sich hierbei auf Gespräche mit der Projektleitung von CIDAS.

ähnliche Betriebssysteme entsprechend Version 3 der *Single UNIX Specification* ([24]) eine Gerätedate `/dev/random` bereit, über die auf Daten echt-zufälliger Systemereignisse wie beispielsweise externe Interrupts zugegriffen werden kann. Für eine erste prototypische Umsetzung des Authentifizierungsverfahrens wird das Auslesen dieses Gerätes sicherlich ausreichend sein. Es sei darauf hingewiesen, daß `/dev/random` nur solange ausgelesen werden kann, wie tatsächlich entsprechende nichtvorhersehbare Systemereignisse auftreten. Für einen produktiven CIDAS-Server, der sehr viele Logins verarbeiten muß, stehen möglicherweise nicht immer ausreichend Zufallsdaten zur Verfügung. Das System wird aus diesem Grund die Werte aus `/dev/random` als Initialisierungswerte für einen Pseudo-Zufallsgenerator, der entsprechend größere Mengen pseudo-zufälliger Daten erzeugt, benutzen müssen. Hinweise zur Erzeugung pseudo-zufälliger aus echt-zufälligen Daten finden sich beispielsweise bei Schneier (vgl. Schneier, [96, 420-428]).

Für die in Kapitel 4 verwendeten Zufallsdaten M_P und M_V werden jeweils 64 Bytes aus `/dev/random` ausgelesen und weiterverarbeitet. Resultierend daraus hat auch ΔM eine Länge von 64 Bytes.

Zeitstempel Um von CIDAS-Client und -Server verstanden zu werden, müssen Zeitstempel ein einheitliches Format haben. Zeitstempel bezeichnen den aktuellen Zeitpunkt der Erzeugung einer Nachricht durch die Anzahl der seit 0:00 Uhr des 1. Januars 1970 UTC abgelaufenen Sekunden und werden als 4 Byte lange, vorzeichenlose Zahlen im big-endian Format gespeichert.

6.3.3 Umsetzung des Authentifizierungsverfahrens mit OpenPGP

Zur Umsetzung des Kapitel 4 konzipierten Authentifizierungsverfahrens mit OpenPGP-Schlüsselmaterial entsprechend RFC 2440 ([43]) empfiehlt der Autor die Verwendung der bereits erwähnten Software *GnuPG*. *GnuPG* steht für *GNU Privacy Guard* und stellt eine freie Implementierung des in RFC 2440 beschriebenen Standards dar. *GnuPG* ist eine *Open-Source-Software*, die von

einer offenen Entwicklergemeinde gepflegt wird. Die Entwicklung wurde maßgeblich vom Bundesministerium für Wirtschaft und Technologie gefördert.

Insbesondere für die clientseitige Implementierung des Protokolls kann *GnuPG* von einem CIDAS-Client auf die selbe Weise benutzt werden, wie beispielsweise von E-Mail-Clients, nämlich durch einen direkten Aufruf. Die zu ver- oder entschlüsselnden Daten werden hierfür vom CIDAS-Client fertig zusammengestellt und dem *GnuPG* unter Verwendung entsprechender Kommandozeilen-Parameter, nachzulesen in Ashley [32], über eine sogenannte Pipe übergeben und die Rückgaben entgegengenommen (vgl. Kernighan et al., [69, 145-147]).

Für die serverseitige Implementierung des Authentifizierungsprotokolls wird diese Methode maximal im Rahmen einer prototypischen Umsetzung ausreichend performant sein. Für eine produktive Version des CIDAS-Servers ist eine in den Server eingebettete Implementierung von OpenPGP anzustreben.

Kapitel 7

Zusammenfassung

Security is a process, not a product.
— Bruce Schneier, *Secrets and Lies* ([98, S. 276])

Zusammenfassung

Wie in den Kapiteln 1 und 2 dargestellt wurde, gibt es in heutigen Netzwerkinfrastrukturen, unabhängig davon, ob diese institutionsintern oder global ausgerichtet sind, häufig das Problem, Benutzer zu identifizieren, die angegebene Identität zu überprüfen und den Benutzer anhand des Ergebnisses für den Zugriff auf bestimmte Ressourcen zu autorisieren. Der Ansatzpunkt des in Kapitel 3 erläuterten Identitymanagement-Systems CIDAS, wie auch der in Abschnitt 2.4 dargestellten Systeme, besteht hierbei darin, einen Benutzer einmalig auf einem zentralen Server zu authentifizieren und anschließend seine Autorisation bei Bedarf über das Netzwerk zu verbreiten. Kritisch an dieser Vorgehensweise ist insbesondere die Tatsache, daß ein Angreifer, sollte es ihm gelingen, den Authentifizierungsserver davon zu überzeugen, ein legitimer Benutzer zu sein, Zugang zu allen Ressourcen hat, für die der betreffende Benutzer zugangsberechtigt ist. Wie in Abschnitt 2.4 erläutert, wird dieses Problem durch die Verwendung schwacher Authentifizierungsverfahren und die Art der Verbreitung von Autorisierungszeichen bei den Lösungen *Passport* und *Liberty Alliance Project* noch verschärft. Auch ergeben sich bei *Passport* Probleme durch die Speicherung großer Mengen personenbezogener Daten auf einem zentralen Server.

Mit dem von Schäfer in [93] konzipierten und in Kapitel 3 dieser Arbeit nochmals vorgestellten System CIDAS wird versucht, diese Probleme durch starke Authentifizierungsverfahren und verschiedene weitere Sicherheitsmaßnahmen zu reduzieren. Hierfür wurde in Kapitel 4 dieser Arbeit ein auf asymmetrischer Kryptographie und mobilen Speichermedien basierendes Authentifizierungsverfahren konzipiert. In Kapitel 5 wurde darüber hinaus das zwischen CIDAS-Server und den Clients zu verwendende Kommunikationsprotokoll spezifiziert. Eine prototypische Umsetzung der dargestellten Konzepte wurde in Kapitel 6 beschrieben.

Bei der Konzeption des Authentifizierungsverfahren konnte gezeigt werden, daß sich auch einfache passive Datenträger durchaus als Authentifizie-

rungsmedien eignen. Hierbei ergeben sich jedoch neue Probleme, insbesondere kann das verwendete Schlüsselmaterial nur sehr schwer von unberechtigten Zugriffen geschützt werden. Insgesamt wird auf diese Weise die von Chipkarten oder vergleichbaren aktiven Authentifizierungsgeräten gebotene Sicherheit nicht erreicht.

Das vom Autor entworfene Kommunikationsprotokoll ist, wie bereits am Anfang von Kapitel 5 dargestellt, partiell unvollständig. Aufgrund der Unspezifiziertheit verschiedener Authentifizierungsverfahren und Komponenten von CIDAS war dem Autor eine vollständige Ausarbeitung des Protokolls unmöglich. Auch birgt die manuelle Art der Protokollentwicklung das Risiko des Übersehens von Details, die sich in einer Implementierung möglicherweise als Schwachstellen oder als nicht ausreichend spezifiziert herausstellen. Auf formale Analysen des Protokolls wurde verzichtet, um den Umfang dieser Arbeit nicht noch weiter auszuweiten.

Die Unvollständigkeit der prototypischen Umsetzung des Kommunikationsprotokolls und des Authentifizierungsverfahrens resultieren aus dem Nichtvorhandensein einer CIDAS-Server-Implementierung. Diese war Gegenstand einer im August 2003 an der Fachhochschule Brandenburg vorgelegten Diplomarbeit ([93]), die weiterführenden Arbeiten an dieser wurden dem Autor von der Projektleitung des Projektes CIDAS zur Bearbeitung der vorliegenden Arbeit zugesichert, liegen jedoch zum gegenwärtigen Zeitpunkt (3. März 2004) noch nicht vor. In Kapitel 6 bzw. auf der beiliegenden CD wird daher neben verschiedenen Hinweisen zur Umsetzung des Authentifizierungsverfahrens nur das Grundgerüst eines CIDAS-Clients und die ansatzweise Implementierung des Protokolls in Codebeispielen belegt.

7.1 Ausblick und weiterführende Arbeiten

Ziel weiterführender Arbeiten am Projekt CIDAS sollten vor allem die Implementierung eines voll funktionsfähigen Authentifizierungsservers und Clients sein. Im Rahmen dieser Tätigkeiten müssen dementsprechend weite-

re Authentifizierungsverfahren spezifiziert werden. Auch wird eine partielle Überarbeitung des Kommunikationsprotokolles, möglicherweise verbunden mit der Verwendung formaler Verifikationsmethoden, erforderlich sein. Ebenfalls weitgehend unspezifiziert ist auch die Übergabe des Autorisierungstokens zwischen dem angemeldeten Benutzer, repräsentiert durch den CIDAS-Client und die jeweiligen Applikationen. Hierfür wird eine Spezifikation von Schnittstellen zu existierenden Authentifizierungsframeworks wie beispielsweise PAM (vgl. OSF-RFC 86.0 [92]) notwendig sein.

Darüber hinaus bietet sich insbesondere in Bezug auf die Verwendung von RFID-Geräten in Verbindung mit CIDAS die Möglichkeit zur Verwendung von CIDAS zur Identifikation von Gegenständen in Fertigungs- oder Verarbeitungsketten.

Als bereits laufende Projekte, die sich mit der Weiterführung der Arbeiten an CIDAS bzw. mit konkreten Einsatzmöglichkeiten für das Identity-Management-System befassen, sei an erster Stelle auf eine Diplomarbeit von C. Thon und M. Kasten verwiesen, deren Ziel es ist, ein Hochschulinformationssystem mit weitreichender Funktionalität auf Basis von CIDAS zu entwickeln. Die Arbeit wird in den nächsten Monaten an der Fachhochschule Brandenburg vorgestellt werden. Auch soll CIDAS im Rahmen des Hochschulprojektes „Lernlandschaft Mobile Multimediale Systeme“ zur Benutzerauthentifizierung eingesetzt werden. Die Konzeption und Umsetzung der CIDAS-Komponente zum Aufnehmen von Benutzern und zur Pflege des Datenbestandes wird in absehbarer Zeit im Rahmen einer weiteren Diplomarbeit an der Fachhochschule Brandenburg erarbeitet und vorgestellt.

Abbildungsverzeichnis

3.1	Modularisierung von CIDAS	37
4.1	Nachrichtenaustausch im vorgestellten Protokoll.	59
4.2	Üblicher Speicheraufbau in RFIDs	72
5.1	Aufbau einer CIDAS-Nachricht.	99
5.2	Aufbau des Flags -Feldes.	100

Tabellenverzeichnis

2.1	Formen der Unsicherheit bei digitalen Transaktionen	13
4.1	Der Geräte-Deskriptor eines USB-Gerätes nach Axelson [34, S. 105 f.]	67
5.1	Identifizierungsmethoden für CIDAS entsprechend Schäfer [93, S. 66 f.]	91
5.2	Spezifikation der Klassen von Authentifizierungsverfahren. . .	92
5.3	Spezifikation der Authentifizierungsmethoden in Abhängigkeit von den möglichen Verfahrensklassen.	93
5.3	Spezifikation der Authentifizierungsmethoden in Abhängigkeit von den möglichen Verfahrensklassen.	94
5.4	Sicherheitslevel in CIDAS	96
5.5	Bedeutung der Bezeichner aus Abbildung 5.1.	99
5.6	Bedeutung der Flags in Abbildung 5.2.	100
6.1	Möglichkeiten der Speicherung der Datei <code>cidas.conf</code>	138

Literaturverzeichnis

- [1] Internet Protocol – DARPA Internet Program Protocol Specification. RFC 791, Information Sciences Institute, University of Southern California, 1981. <http://www.faqs.org/rfcs/rfc791.html>; visited on February 1st 2004.
- [2] Transmission Control Protocol – DARPA Internet Program Protocol Specification. RFC 793, Information Sciences Institute, University of Southern California, 1981. <http://www.faqs.org/rfcs/rfc793.html>; visited on February 1st 2004.
- [3] Volume and File Structure of CDROM for Information Interchange. ECMA 119, Ecma International – Standardizing Information and Communication Systems, 1987. <http://www.ecma-international.org/publications/standards/Ecma-119.htm>; visited on November 19th 2003.
- [4] IEEE Standard Portable Operating System Interface for Computer Environments. IEEE Std. 1003-1, Institute of Electrical and Electronics Engineers, Inc., 1988.
- [5] Volume and File Structure of CD-ROM for Information Interchange. ISO 9660, International Organization for Standardization, 1988.
- [6] *Die Bibel in heutigem Deutsch*. Deutsche Bibelgesellschaft Stuttgart, Stuttgart, 2nd edition, 1990.
- [7] IEEE Standard Portable Operating System Interface for Computer Environments (POSIX.1-1988 update). IEEE Std. 1003-1, Institute of Electrical and Electronics Engineers, Inc., 1990.
- [8] GNU General Public License. Website, June 1991. <http://www.gnu.org/>; visited on January 20th 2004.

- [9] IEEE Standard Portable Operating System Interface for Computer Environments (Realtime Extensions). IEEE Std. 1003-1, Institute of Electrical and Electronics Engineers, Inc., 1993.
- [10] Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. ISO/IEC iso7498-1, International Organization for Standardization, 1994.
- [11] IEEE Standard Portable Operating System Interface for Computer Environments (Threads Extensions). IEEE Std. 1003-1, Institute of Electrical and Electronics Engineers, Inc., 1996.
- [12] Planning and Deploying a Single Sign-On Solution. Website, 1997. <http://developer.netscape.com/docs/manuals/security/SSO/sso.htm>; visited on January 20th 2004.
- [13] Universal Serial Bus Specification – Revision 1.1. Specification, Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, NEC Corporation, September 1998. <http://www.usb.org/>; visited on January 20th 2004.
- [14] Extensible Markup Language (XML) 1.0. Website, 2000. <http://www.w3.org/TR/REC-xml>; visited on February 3rd 2004.
- [15] Identification cards – Contactless integrated circuit(s) cards – Proximity cards. ISO/IEC 14443, International Organization for Standardization, 2000.
- [16] Identification cards – Contactless integrated circuit(s) cards – Vicinity cards. ISO/IEC 15693, International Organization for Standardization, 2000.
- [17] Information technology – Universal Multiple-Octet Coded Character Set (UCS). ISO 10646, International Organization for Standardization, 2000.
- [18] Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics. IEC 60027-2, International Electrotechnical Commission, 2000.
- [19] Universal Serial Bus Revision 2.0 specification. Specification, Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V., December 2000. <http://www.usb.org/>; visited on January 20th 2004.

- [20] FIPS197: Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication*, (197), 2001. <http://csrc.nist.gov/CryptoToolkit/aes/>; visited January 25th 2004.
- [21] Microsoft passport technical overview. White paper, Microsoft Corporation, 2001. download.microsoft.com/download/1/0/0/10088d15-5a65-4ad8-a4bf-bf466012c%407/wp_engl_net_passport.doc; last visited February 16th 2004.
- [22] Short Product Information: SPI SRF55V10P. Product information, Infineon Technologies AG, July 2002. available at <http://www.infineon.com/>; visited on January 19th 2004.
- [23] Short Product Information: SPI SRF55V10S. Product information, Infineon Technologies AG, July 2002. available at <http://www.infineon.com/>; visited on January 19th 2004.
- [24] The Single UNIX Specification, Version 3. Specification, The Open Group, 2002. <http://www.unix-systems.org/version3/>; visited on February 9th 2004.
- [25] Bundesdatenschutzgesetz. gem. Bekanntmachung vom 14. Januar 2003, 2003. Bundesdatenschutzgesetz vom 20. Dezember 1990, BGBl. I S. 2954, 2954, zuletzt geändert durch Art. 3 des Gesetzes vom 16.12.1997, BGBl. I S. 3094, und durch Art. 2 Abs. 5 des Gesetztes vom 17.12.1997, BGBl. I S. 3108, durch Art 1 des Gesetzes vom 18.05.2001, BGBl. I S. 904, den Artikel 3 Abs. 3 des Gesetzes vom 26. Juni 2001 (BGBl. I S. 1254), den Artikel 21 des Gesetzes vom 3. Dezember 2001 (BGBl. I S. 3306), den Artikel 3 Nr. 2 des Gesetzes vom 20. Dezember 2001 (BGBl. I S. 3926), den Artikel 9 Abs. 1 des Gesetzes vom 19. Juli 2002 (BGBl. I S. 2674) sowie den Artikel 12 des Gesetzes vom 21. August 2002 (BGBl. I S. 3322).
- [26] Liberty Architecture Overview – Version 1.1. White paper, Liberty Alliance Project, 2003. available at www.projectliberty.org; last visited February 16th 2004.
- [27] Microsoft .NET Passport for Businesses. Website, 2003. <http://www.microsoft.com/net/services/passport/business.asp> visited Februar 16th 2004.
- [28] SOAP: Simple Object Access Protocol – Version 1.2. Website, 2003. <http://www.w3.org/>; visited on February 3rd 2004.

- [29] Kompressionsbomben gefährden Antivirens Scanner. Website, 2004. <http://www.heise.de/newsticker/meldung/43562>; visited February 16th 2004.
- [30] What is RFID? Website, 2004. <http://www.ems-rfid.com/whatisrfid.html>; visited on January 19th 2004.
- [31] Martín Abadi and Mark R. Tuttle. A Semantics for a Logic of Authentication. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, number 10th, Montreal, Canada, 1991.
- [32] Mike Ashley. The GNU Privacy Handbook. Website, 1999. <http://www.gnupg.org/gph/en/manual.html>; visited February 14th 2004.
- [33] St. Augustinus. *DE DOCTRINA CHRISTINA*. Kein bekannter Verlag, 397.
- [34] Jan Axelson. *USB – Handbuch für Entwickler*. MITP-Verlag, Bonn, 2001.
- [35] J. W. Backus. The syntax and semantics of the proposed international algebraic language of the Zurich ACMG-GAMM conference. In *Proceedings of the International Conference on Information Processing*, pages 125–132, 1959.
- [36] D. Balenson. Privacy enhancement for internet electronic mail: Part iii: Algorithms, modes, and identifiers. RFC 1423, Internet Engineering Task Force, 1993. <http://www.faqs.org/rfcs/rfc1423.html>; visited on October 1st 2003.
- [37] Kent Beck. *Extreme Programming*. Addison-Wesley, München, 2001.
- [38] Michael Behrens and Richard Roth. *Biometrische Identifikation*. Friedrich Vieweg und Sohn Verlagsgesellschaft mbH, Braunschweig, 2001.
- [39] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396, 1998. <http://www.faqs.org/rfcs/rfc2396.html>; visited on February 1st 2004.
- [40] Adam D. Bradley, Azer Bestavros, and Assaf J. Kfoury. Validating Arbitrarily Large Network Protocol Compositions with Finite Computation. citeseer.nj.nec.com/article/bradley02validating.html; last visited February 20th 2004.

- [41] Neil Bradley. *The XML Companion*. Addison-Wesley, Harlow, 1999.
- [42] Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. In *Proceedings of the Royal Society*, number 426 in A, pages 233 – 271, 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in *ACM transactions on Computer Systems* 8, 1 (February 1990), 18-36.
- [43] J. Callas, L. Donnerhacker, H. Finney, and R. Thayer. OpenPGP message format. RFC 2440, 1998. <http://www.faqs.org/rfcs/rfc2440.html>; visited on October 1st 2003.
- [44] CCITT. The Directory-Authentication Framework. Recommendation X.509, Consultation Committee, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989. can be purchased at <http://www.itu.int>; visited on January 24th 2004.
- [45] Volker Claus and Andreas Schwill, editors. *Duden Informatik*. Dudenverlag, Mannheim, 3rd edition, 2001.
- [46] The BioAPI Consortium. BioAPI Specification Version 1.1. Specification, 2001. <http://www.bioapi.org>; visited on January 15th 2004.
- [47] David H. Crocker. Standard for the Format of ARPA Internet Text Messages. RFC 822, 1982. <http://www.faqs.org/rfcs/rfc822.html>; visited on February 1st 2004.
- [48] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag, Berlin, 2002.
- [49] P. Deutsch and J-L. Gailly. ZLIB Compressed Data Format Specification version 3.3. RFC 1950, 1996. <http://www.faqs.org/rfcs/rfc1950.html>; visited on February 1st 2004.
- [50] T. Dierks and C. Allen. The TLS Protocol. RFC 2246, Internet Engineering Task Force, 1999. <http://www.faqs.org/rfcs/rfc2246.html>; visited on December 21st 2003.
- [51] W. Diffie and M. E. Hellmann. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22, 1976.
- [52] Claudia Eckert. *IT-Sicherheit – Konzepte, Verfahren, Protokolle*. Oldenbourg Verlag, Munich, 2nd edition, 2003.

- [53] Taher ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology – Proceedings of Crypto 1984*, pages 10 – 18, 1984. available within [78].
- [54] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, 1999. <http://www.faqs.org/rfcs/rfc2616.html>; visited on February 1st 2004.
- [55] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol – version 3. Internet draft, Netscape Inc., Transport Layer Security Working Group, 1996. <http://wp.netscape.com/eng/ssl3/>; visited on December 22st 2003.
- [56] Kai Fuhrberg. *Internet Sicherheit*. Hanser Verlag, Fulda, 2nd edition, 2000.
- [57] Sandra Gerbich. Authentifizierung. *InformationWeek*, (24):18–19, 11 2003.
- [58] N. Haller and R. Atkinson. On Internet Authentication. RFC 1704, Network Working Group, 1994. <http://www.faqs.org/rfcs/rfc1704.html>; visited on December 28th 2003.
- [59] N. Haller, C. Metz, P. Nesser, and M. Straw. A One-Time Password System. RFC 2289, Network Working Group, 1998. <http://www.faqs.org/rfcs/rfc2289.html>; visited on December 28th 2003.
- [60] D. Harkis and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, Network Working Group, 1998. <http://www.faqs.org/rfcs/rfc2409.html>; visited on December 28th 2003.
- [61] Friedrich-L. Holl and Jan Tobias Mühlberg. Überblick über Aufbau und Funktionsweise von CIDAS. Draft, 2004. <http://zeus.fh-brandenburg.de/cidas/>; visited on January 3rd 2004.
- [62] G. J. Holzmann. Tutorial: Design and Validation of Protocols. In B. Pehrson, B. Jonsson, and J. Parrow, editors, *Proc. 11th Int. Conf on Protocol Specification, Testing, and Verification, INWG/IFIP*, Stockholm, Sweden, 1991.
- [63] G. J. Holzmann. Designing bug-free protocols with Spin. *Computer Communications Journal*, 20(2):97–105, 1997.

- [64] R. Housley. Cryptographic Message Syntax. RFC 2630, Internet Engineering Task Force, 1999. <http://www.faqs.org/rfcs/rfc2630.html>; visited on October 1st 2003.
- [65] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure – Certificate and CRL Profile. RFC 2459, Internet Engineering Task Force, 1999. <http://www.faqs.org/rfcs/rfc2459.html>; visited on January 24th 2004.
- [66] Entrust Inc. Secure identity management. White paper, Entrust Inc., 2003.
- [67] B. Kaliski. Privacy enhancement for internet electronic mail: Part iv: Key certification and related services. RFC 1424, Internet Engineering Task Force, 1993. <http://www.faqs.org/rfcs/rfc1424.html>; visited on October 1st 2003.
- [68] S. Kent. Privacy enhancement for internet electronic mail: Part ii: Certificate-based key management. RFC 1422, Internet Engineering Task Force, 1993. <http://www.faqs.org/rfcs/rfc1422.html>; visited on October 1st 2003.
- [69] Brian W. Kernighan and Dennis M. Ritchie. *Programmieren in C – ANSI C*. Carl Hanser Verlag, Munich, 2nd edition, 1990.
- [70] H. Kersten. *Sicherheit in der Informationstechnik*. R. Oldenbourg Verlag, München, 2nd edition, 1995.
- [71] J Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC 1510, Network Working Group, 1993. <http://www.faqs.org/rfcs/rfc1510.html>; visited on December 28th 2003.
- [72] Wolfgang P. Kowalk. *Korrekte Software: Semantik, Spezifikation, Verifikation und Testen von Programmen*. BI-Wissenschafts-Verlag, Mannheim, 1993.
- [73] J. Linn. Privacy enhancement for internet electronic mail: Part i: Message encryption and authentication procedures. RFC 1421, Internet Engineering Task Force, 1993. <http://www.faqs.org/rfcs/rfc1421.html>; visited on October 1st 2003.
- [74] Gavin Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.

- [75] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.
- [76] Charles C. Mann. Homeland Insecurity. Website, 2002. available at The Atlantic Online; <http://www.theatlantic.com/issues/2002/09/mann.htm>; last visited February 9th 2004.
- [77] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408, Network Working Group, 1998. <http://www.faqs.org/rfcs/rfc2408.html>; visited on December 28th 2003.
- [78] Kevin S. McCurley and Claus Dieter Ziegler, editors. *Advances in Cryptology 1981-1997 – Electronic Proceedings of the Crypto and Eurocrypt Conferences 1981-1997*, Heidelberg, 1998. Springer Verlag.
- [79] Stefan Meretz. Freie Software – 20 Thesen für eine andere Gesellschaft. *spw – Zeitschrift für Sozialistische Politik und Wirtschaft*, 120(4/01), 2001. As addition to discussions about OSS an ethics.
- [80] Hans-Peter Messmer. *PC-Hardwarebuch – Aufbau, Funktionsweise, Programmierung*. Addison-Wesley, Munich, Germany, 2000.
- [81] Roger Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):pp. 993–999, 1978.
- [82] Otto Petrovic, Markus Fallenböck, Christian Kittl, and Thomas Wolkingner. Vertrauen in digitale Transaktionen. *Wirtschaftsinformatik*, 45(1):53–66, 01 2003.
- [83] Norbert Pohlmann. Die Zukunft von Public-Key-Infrastrukturen. *IT-Sicherheit*, 8(6/2002):38–43, June 2002.
- [84] Obye Polybius. *The Histories*. Harvard University Press, Harvard, 1978.
- [85] J. Postel and J. Reynolds. File Transfer Protocol (FTP). RFC 959, 1985. <http://www.faqs.org/rfcs/rfc959.html>; visited on February 1st 2004.

- [86] Olga Pötzsch, Birgit Korth, and Susanne Schnorr-Bäcker. *Informationstechnologie in Haushalten - Ergebnisse einer Pilotstudie für das Jahr 2002*. Statistisches Bundesamt, Frankfurt am Main, 2003.
- [87] B. Ramsdell. S/MIME Version 3 Message Specification. RFC 2633, Internet Engineering Task Force, 1999. <http://www.faqs.org/rfcs/rfc2633.html>; visited on October 1st 2003.
- [88] Wolfgang Rankl and Wolfgang Effing. *Handbuch der Chipkarten*. Carl Hanser Verlag, München and Wien, 3rd edition, 1999.
- [89] Eric Raymond and Bruce Perens. *Open Sources*. O'Reilly Verlag, London, 1999.
- [90] Claudia Remus. Telefonische Notenabfrage. Circular email/Spam, 2004. Published by circular email to the students mailing lists on January 27th 2004, Message-Id:<5.2.0.9.0.20040127101808.00ae7d50@pop3.fh-brandenburg.de>.
- [91] Arto Salomaa. *Public-Key Cryptography*. Springer, Berlin, 2nd edition, 1996.
- [92] V. Samar and R. Schemers. Unified login with pluggable authentication modules (pam). DCE/OSF-RFC 86.0, The Open Group, 1996. <http://www.kernel.org/pub/linux/libs/pam/>; last visited 23. February 2004.
- [93] Ingo Schäfer. Konzeption und prototypische Umsetzung eines universellen Benutzermanagement-Systems mit Transaktionskomponente. Diploma thesis, University of Applied Sciences in Brandenburg, Germany, August 2003.
- [94] Wolfgang Schmidetzki and pg. Personalisierung macht das Internet zeitsparend nutzbar. *Computerzeitung*, (47):24, 11 2003.
- [95] Bruce Schneier. Description of a New Variable-Length Key, 64-Bit Block Cypher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings*, Berlin, 1994. Springer-Verlag.
- [96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., New York, 2nd edition, 1996.
- [97] Bruce Schneier. Crypto-gram newsletter. Website, September 1999. <http://www.schneier.com/crypto-gram-9909.html>; visited on January 26th 2004.

- [98] Bruce Schneier. *Secrets & Lies*. Wiley Computer Publishing, New York, 2000.
- [99] Bruce Schneier. *Beyond Fear*. Copernicus Books, an imprint of Springer-Verlag, New York, 2003.
- [100] Ian Sommerville. *Software Engineering*. Pearson Studium, Munich, 6th edition, 2001.
- [101] Sorin G. Stan. *The CD-ROM Drive – A Brief System Description*. Kluwer Academic Publishers, Boston, 1998.
- [102] Clifford Stoll. *Die Wüste Internet*. S. Fischer Verlags GmbH, Frankfurt am Main, 2002.
- [103] Andrew S. Tannenbaum. *Computernetzwerke*. Pearson Studium, Munich, 3rd edition, 2000.
- [104] Andrew S. Tannenbaum and Steen Maarten van. *Verteilte Systeme – Grundlagen und Paradigmen*. Pearson Studium, Munich, 2003.
- [105] Virginia U.S. Department of Commerce/ National Bureau of Standards/ National Technical Information Service, Springfield. FIPS46: Data Encryption Standard. *Federal Information Processing Standards Publication*, (46), 1977. <http://www.itl.nist.gov/fipspubs/fip46.htm>; visited January 25th 2004.
- [106] Virginia U.S. Department of Commerce/ National Bureau of Standards/ National Technical Information Service, Springfield. FIPS186: Digital Signature Standard (DSS). *Federal Information Processing Standards Publication*, (186), 1994. <http://www.itl.nist.gov/fipspubs/fip186.htm>; visited January 25th 2004.
- [107] David A. Wheeler. *Secure Programming for Linux and Unix HOW-TO*. Website, 2003. <http://www.dwheeler.com/secure-programs/>; visited on January 20th 2004.
- [108] David A. Wheeler. *Why Open Source Software / Free Software? Look at the Numbers!* Website, 2004. http://www.dwheeler.com/oss_fs_why.html; visited on June 17th 2004.
- [109] Alma Whitten and J. D. Tygar. *Usability of Security: A Case Study*. Website, 1998. <http://reports-archive.adm.cs.cmu.edu/anon/1998/abstracts/98-155.html>; visited February 14th 2004.

- [110] Mark Wilcox. *Implementing LDAP*. Wrox Press Ltd., Acocks Green, 1999.
- [111] Rutrell Yasin. What is Identity Management? Website, 2002. http://infosecuritymag.techtarget.com/2002/apr/cover_casestudy.shtml; visited on November 30th 2003.
- [112] Alec Yasinsac and William A. Wulf. Evaluating cryptographic protocols. Technical Report CS-93-66, 22, 1993.
- [113] W. Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. RFC 1777, 1995. <http://www.faqs.org/rfcs/rfc1777.html>; visited on December 22nd 2003.
- [114] F. Yergeau. UTF-8, a transformation format of ISO 10646. RFC 2279, 1998. <http://www.faqs.org/rfcs/rfc2279.html>; visited on February 17th 2004. ISO 10646 is specified in [17].

Index

A

- Access-Management** 16
- AES** 47
- Angriff** 54, 75–77, 83
 - aktiver-, 22
 - gegen Einweg-Funktionen, 24
 - Wiederholungs-, 21, 28
- ATIP** 70
- Authentifizierung** . 6, 9, 19, 22–24,
27–29, 31, 38, 39, 41, 69, 75,
77–79, 135, 144
 - sdaten, 21
 - smedium, 80, 144
 - sprotokoll, 27
 - sverfahren, 3, 6, 19–21, 40, 42,
54, 80
 - Bewertung von -sverfahren, 28,
59
 - biometrische -sverfahren, 20, 22
 - mittels asymmetrischer Krypto-
graphie, IV, 3, 6, 54, 56, 80,
92
 - mittels Kryptographie, 24
 - something he has-, 20
 - something he knows-, 20
 - textuelle -sverfahren, 20, 92
- Authentifizierungsmethode** ... 91
- Autorisierung** 5, 9, 19, 39, 144

- szeichen, 43, 95
- Cookie, 29–32, 43

B

- BAN-Logik** 28, 57, 59
- Benutzer** .. 9, 14, 19, 22, 41, 55, 76,
79, 83, 90
 - Verwaltung von-, 9, 16, 29, 31, 35,
36, 38–41, 146
- Benutzungsschnittstelle** 131
- Betriebssystem** 47, 129
- Bezeichnerwahl** 86
- Big-Endian-Format** 88
- Biometrie** 22, 92
 - Fingerabdruck, 23
 - Gesichtserkennung, 23
 - Sprechererkennung, 23
 - Zuverlässigkeit, 23
- Blowfish** 47

C

- C** 129, 181
- CD-ROM** 45, 46, 69, 70

- Certificate-Authority** 79
- Chipkarte** 20, 45, 46, 92, 144
- CIDAS** IV, 3, 5, 35, 61, 75, 144
 -Client, IV, 52, 75, 77–79, 131, 135, 145, 181
 -Protokoll, 3, 6, 78, 79, 86, 88–90, 145, 169, 181
 -Server, IV, 52, 55, 145, 181
 -Token, 95
 Authentifizierung, 6, 38, 39, 41, 42, 91, 144
 Autorisierung, 39
 Benutzerverwaltung, 35, 146
 Client, 36, 61, 76, 80
 Datenbank, 42
 Einsatzmöglichkeiten, 35, 47, 146
 Identifizierung, 39, 41
 Identity-Management-System, 3, 35, 36
 Kommunikation, 38
 Kompatibilität, 38
 Komponentenmodell, 36
 Open-Source, 43
 personenbezogene Daten, 36, 40, 41
 Schnittstellen, 36
 Server, 36
 Sicherheit, 42, 77
 Single-Sign-On, 38, 42
 Special-Feature-Module, 42
 unterstützte Anwendungen, 38
- CIDAS-Client** 83
- cidas_1-0.dtd** 169
- cidas_1-0.h** 181
- Client** 9
- Cookie** 43
- D**
- Dateisystem** 135
 Datenträger mit-, 135
 Datenträger ohne-, 139
 verschlüsseltes-, 139
- Daten**
 personenbezogene-, 36, 40, 81
- Datenbank** 42
- Datensatz**
 biometrischer, 23
- Datenschutz** 15
- Datenspeicherung** 135
- Datentäger** 62
- Datenträger** 6, 21, 51, 63
 identifizierbarer-, 6, 45, 46, 48, 51, 71
 indifizierbarer, 65
 passiver, 144
- DES** 47
- Dienst** 9–11
 Anbieter, 9
 Zugriff auf-, 11
- Diskette** 45, 46
- DSA** 63
- DVD** 70

E

E-Mail 2, 90
Einmal-Passwort 21
Einweg-Funktion 24
ElGamal 63
Entschlüsselung 24–26
Extreme Programming 4, 130

F

Festplatte 45, 46
Fingerabdruck 23

G

Gesichtserkennung 23
glib 130

GNU

Compiler Collection, 130
 General Public License, 35
 Privacy Guard, 63

GPL 35

H

Home-Banking 81

I

Identifizierung 9, 19, 21, 39, 41, 77, 83, 90, 144
 -eines Mediums, 63, 70
 -von CIDAS-Clients, 78, 79
 Ausspähen von Daten, 41
 Benutzername, 41, 90
 biometrische -sverfahren, 22
 E-Mail-Adresse, 41

Identifizierungsdaten 75

Identität

-sraub, 15, 23
 von Benutzern, 41

Identity Theft 15

Identity-Management 3, 5, 16
 -System, IV, 3, 16, 29, 31, 35, 36, 146

Internet 2, 9–11, 14
 -Terminal, 77

ISO/OSI-Referenzmodell 10

J

Java 42

K**Kommunikation**

-smedium, 10
 -sprotokoll, 10
 -sverbindung, 21, 24, 29, 31

Kommunikationsprotokoll 6, 9, 85

-
- Aufgaben von-, 9
 - FTP, 11
 - HTTP, 11
 - Protokollschichten, 10
 - Komponentenmodell** 36
 - Kompromittierung** 69, 80, 82
 - Konfiguration** 76
 - Kryptographie** 24
 - asymmetrische, IV, 25–28, 82
 - asymmetrische-, 47
 - Schlüsselaustausch, 24, 26, 27
 - Schlüsselmaterial, 25, 28
 - symmetrische, 24, 25, 27, 28, 79
 - symmetrische-, 47
 - Ziele der-, 24
 - zur Authentifizierung, 24, 27
 - L**
 - LDAP** 38
 - Legitimierung** 20
 - Liberty Alliance** 31, 38
 - Datenspeicherung, 32
 - Offener Standard, 32
 - Probleme, 32
 - Liberty Alliance Project** 144
 - libxml** 130
 - M**
 - Merkmal**
 - biometrisches, 22, 23
 - Microsoft**
 - Passport, 29
 - Mithören von Tastatureingaben**
 - 77
 - Modularisierung** 36
 - N**
 - Nachrichtenaufbau** 87, 98, 103, 181
 - Nachrichtentyp** 98, 103, 181
 - Needham-Schroeder** 28, 54
 - Network-Byte-Order** 88
 - Netzwerk** 2, 5, 9, 10, 14
 - O**
 - Online-Banking** 77, 81
 - Open-Source** 13, 35, 50, 80
 - OpenPGP** 39, 48, 52, 56, 63, 69, 75
 - OpenSSL** 63
 - P**
 - Partnerunsicherheit** 12, 14
 - Passport** 29, 144
 - Identifizierung und Authentifizierung, 29
 - Probleme, 30
 - Single-Sign-On, 29

-
- Passwort** 19–21, 77, 80
 -Reset, 17
 -Synchronisierung, 17
- Payload** 98
- PEM** 48
- Postanschrift** 90
- Private-Key** 25
- Protokollentwurf** 6
- prototypische Umsetzung** ... 3, 6,
 128
- Pseudo-Nachricht** 83
- Public-Key** 25
- Public-Key Cryptography** 25
- Public-Key-Infrastructure** . 48, 57
- Q**
- Quellcode** 7, 14, 80
- R**
- RFID** 45, 46, 71, 73, 92, 146
- RSA Security Inc.** 80
- S**
- Schlüssel** 24
 geheimer-, 25, 28, 58
 Länge, 63
 öffentlicher-, 47, 58, 63
 privater-, 25, 28, 47
- Schlüssel, privater-** 75
- Schlüsselaustausch** 24, 26, 27
- Schlüsselmaterial** ... 25, 28, 56, 63,
 64, 77, 79, 81, 83
- Schriftwerkennung** 23
- Sekundärschlüsseln** 81
- Server** 9
- Sicherheit** . 2, 12, 13, 42, 43, 47, 76,
 77
 der Authentifizierung, 144
 Evaluierung, 43
 Single-Sign-On, 18, 42
 von Schlüsselmaterial, 24, 26, 28,
 144
- Sicherheitslevel** 94
- Signatur**
 digitale, 51, 64, 69, 73
- Single-Sign-On** 17, 38, 49
 Probleme, 18
 Sicherheit, 42
- Speicherkapazität** 64, 70
- Speichermedium** IV, 45, 46, 51, 62,
 63, 65, 70, 135
- Sprechererkennung** 23
- SSL** 78, 79, 89

Systemunsicherheit 12, 13

T

TAN 81

TLS 78, 79, 89

Token 95

Transaktion

digitale, 2, 9

Vertrauen in-, 2

Transponder 46, 71

Trust-Center 3

U

UNIX 129, 135

USB-Memory-Stick 45, 46, 65, 66,
69

V

Verbindungsaufbau 89

Verschlüsselung 24–26, 79, 139

Vertrauen 2, 3, 12, 16, 75, 77
Aufbau von-, 3, 12–14

Vertraulichkeit 24

X

X.509 . 39, 48, 52, 54–56, 69, 75, 78,
79

XML 87, 116, 169

Z

Zeitstempel 57, 58, 141

Zertifikat 48, 78–80

Zufallsdaten ... 57, 58, 79, 140, 141

Anhang A

Eine Beispielsitzung des CIDAS-Protokolls

Die folgende Tabelle stellt eine Beispielsitzung aus Sicht des CIDAS-Servers dar. In dieser Beispielsitzung wird lediglich eine einfache Identifizierung, Authentifizierung, ein nachfolgender Tokenaustausch behandelt und die Beendigung der Sitzung behandelt. Die Spalte Richtung gibt hierbei an, ob eine Nachricht vom Server empfangen (\rightarrow) oder von ihm gesendet (\leftarrow) wird. Die mittlere Spalte stellt die eigentliche Nachricht in hexadezimaler Codierung, jeweils vier Oktets pro Zeile, dar. Diese Darstellung ist unter Umständen verkürzt, zumindest soweit es den Payload der Nachrichten betrifft – eine vollständige hexadezimale Darstellung würde teilweise mehrere Seiten pro Nachricht einnehmen. Der Anfang des Payloads wird durch einen wagerechten Strich in der mittleren Tabellenspalte symbolisiert. Nebenstehend werden Nachricht und Payload erläutert. Die verwendeten Payload-Inhalte entsprechen den in Abschnitt 5.2.10 verwendeten.

Richtung	Paketlayout (Hex.)				Bedeutung
\rightarrow	01	00	01	00	Die Session beginnt mit einem CONREQ.
	00	00	00	00	Die verwendete Versionsnummer ist die
	00	00	00	00	in diesem Dokument beschriebene 1.0,
	00	00	00	00	der Payload ist leer und SeqC sowie SID
					sind vorschriftsmäßig mit Null initialisiert.

Rich- tung	Paketlayout (Hex.)				Bedeutung
←	01	00	02	00	Der Authentifikationsserver nimmt die Verbindung an und sendet einen CO-NACK zurück. Die fortan genutzte Protokollversion ist 1.0. Der SeqC wurde initialisiert und ein SID wurde vergeben.
	2F	A1	00	00	
	03	3E	11	4C	
	03	3E	11	4C	
←	01	00	03	00	Im Folgenden sendet der Authentifikationsserver eine IDDATAREQ-Nachricht an den Client. Payload der Nachricht werden zulässige Identifikationsverfahren spezifiziert, die hierfür verwendete XML-Struktur ist identisch mit dem auf Seite 117 gegebenen Beispiel.
	2F	A1	01	EB	
	03	3E	02	04	
	03	3E	11	4C	
	3C	3F	78	6D	
	6C	20	76	65	
	72	73	69	6F	
	6E	3D	22	31	
	2E	30	22	20	
			...		
→	01	00	04	00	Der Client übermittelt dem Server seine Identifikationsdaten in einer IDDATA-Nachricht. Der SeqC wurde um 1 inkrementiert. Der Payload enthält wiederum eine XML-Struktur, vgl. iddata, Seite 118.
	2F	A2	01	46	
	03	3E	01	4E	
	03	3E	11	4C	
	3C	3F	78	6D	
			...		
→	01	00	05	00	Anschließend sendet er einen AUTHREQ um dem Server seine gewünschte Authentifikationsmethode mitzuteilen, in diesem Fall möchte sich der Benutzer mittels seines Passwortes authentifizieren. Dies geht aus der authmethods-Struktur im Payload hervor, vgl. Seite 118.
	2F	A2	01	02	
	03	3E	11	4C	
	03	3E	11	4C	
	3C	3F	78	6D	
			...		
←	01	00	06	00	In einem AUTHDATAREQ fragt der Server nun die benötigten Authentifikationsdaten ab. In diesem Fall wird nach einem Passwort gefragt, die Anfrage ist in einer XML-Struktur des Typs authdatareq, ähnlich Seite 119, kodiert.
	2F	A3	02	29	
	03	3E	11	4C	
	03	3E	11	4C	
	3C	3F	78	6D	
			...		

Richtung	Paketlayout (Hex.)				Bedeutung
→	01	00	07	00	Der Client überträgt die angefragten Authentifizierungsdaten in einer AUTH-DATA-Nachricht. Der Payload enthält eine <code>authdata</code> -Struktur ähnlich dem auf Seite 120 aufgeführten Beispiel.
	2F	A4	01	01	
	03	3E	11	4C	
	03	3E	11	4C	
	3C	3F	78	6D	
	...				
←	01	00	08	00	Die Authentifizierung war erfolgreich, der Server signalisiert dies durch das Übersenden einer AUTHSUCC-Nachricht signalisiert.
	2F	A5	00	00	
	03	3E	11	4C	
	03	3E	11	4C	
	...				
→	01	00	0A	00	Mit einem nachfolgenden TOKREQ fordert der authentifizierte Client ein Autorisierungszeichen an.
	2F	A6	00	00	
	03	3E	11	4C	
	03	3E	11	4C	
←	01	00	0B	00	Der Server erzeugt das angeforderte Token und übergibt es dem Client in einer TOKDATA-Nachricht. Die Nachricht enthält im Payload eine XML-Struktur mit einem <code>token</code> -Tag entsprechend Seite 120.
	2F	A7	02	B5	
	03	3E	11	4C	
	03	3E	11	4C	
	3C	3F	78	6D	
	...				
→	01	00	25	00	Der Client sendet anschließend eine LOGOUT-Nachricht. Diese führt zur Beendigung der Verbindung.
	2F	A8	00	00	
	03	3E	11	4C	
	03	3E	11	4C	

Anhang B

cidas_1-0.dtd

Die Datei `cidas_1-0.dtd` enthält die Data Type Definition für die im CIDAS-Protokoll verwendeten XML-Tags:

```
<!--
  This file specifies the usable XML-Tags within the
  CIDAS-Protocol. For additional explanations see the
  specification of this protocol.
  VERSION: 1.0
-->

<!DOCTYPE cidas [

<!ELEMENT cidas (idmethods | iddata |
  authdatareq | authdata |
  token | token_peer_record |
  token_ver_record | token_ver_succ |
  token_perm_den |
  capability | capdata | sfmdata |
  serverrecord |
  datareadrequest | datawriterequest |
  userdaccrequest | userdaccinfo | userdaccperm |
  userdataacq | dataresponse |
  infomsg)>

<!--
  The structure 'infomsg' is designed to store a multilingual
  message for the user. The client has to decide what
  language to use and prints out the message appropriate.
```

```

-->
<!ELEMENT infomsg (infomsg_message*)>
  <!ELEMENT infomsg_message (#PCDATA)>

<!--
  Since clients may keep connections to several servers
  and therefore need to know which server could verify
  which token, the token contains a Cidas Server
  Identifier. CSIDs can be explicitly requested from a
  server a client is authenticated at by using the
  CIDAS_MTYPE_CSRREQ.
-->
<!ELEMENT csid (csid_hostname, csid_port_ssl?, csid_port_tcp?,
  csid_pversions)>
  <!-- Hostname to query. -->
  <!ELEMENT csid_hostname (#PCDATA)>

  <!-- Port Numbers for CIDAS-Unsecure und CIDAS-Secure.
  If they are not specified, the defaults are assumed
  to be valid. -->
  <!ELEMENT csid_port_ssl (#PCDATA)>
  <!ELEMENT csid_port_tcp (#PCDATA)>

  <!-- What protocol versions are suported? -->
  <!ELEMENT csid_pversions (csid_pversion+)>
    <!ELEMENT csid_pversion (#PCDATA)>

<!--
  Generic User IDs are used to identify a user within
  a single application. GUIDs are just strings, generated
  by the server and provided to the application. Each
  user has a unique GUID for each application he uses.
  GUIDs consists of not more than 64 characters.
-->
<!ELEMENT guid (#PCDATA)>

<!--
  A user is a legal or physical person. Its attributes are

```

```

        self explaining.
-->
<!ELEMENT user (user_name?, user_address?, user_comment*)>
    <!ELEMENT user_name    (#PCDATA)>
    <!ELEMENT user_address (#PCDATA)>
    <!ELEMENT user_comment (#PCDATA)>

<!--
    A service within CIDAS might be determined by its name,
    perhaps uri and (several, multilingual) descriptions.
    Beside this a service is provided by a physical or legal
    person who should provide his/her details within the
    service_provider record.
-->
<!ELEMENT service (service_name, service_provider,
                    service_uri, service_description*)>
    <!ELEMENT service_name    (#PCDATA)>
    <!ELEMENT service_provider (user)>
    <!ELEMENT service_uri    (#PCDATA)>
    <!ELEMENT service_description (#PCDATA)>

<!--
    A quite essential thing within the CIDAS protocol are
    tokens. They are used to verify the identity of
    authenticated users.
-->
<!ELEMENT token (token_time-issued, token_time-valid,
                 token_tokendata, csid)>
    <!-- Tokens are valid from the time they are issued
         until the time given by 'token_time-valid'
         (both contain the number of seconds elapsed
         since midnight, 01. January 1970, UTC, in decimal
         encoding) is reached.
-->
    <!ELEMENT token_time-issue (#PCDATA)>
    <!ELEMENT token_time-valid (#PCDATA)>

    <!-- The token itself contains of 128 bytes of random
         data. Radix-64 encoding is used.
-->

```

```

<!ELEMENT token_tokendata    (#PCDATA)>

<!--
    The Token Verification Record is used to transfer a
    given token to the authentication server in order to
    get it validated.
    Its infomsg-tag is used in a similar manner as the
    'reason for transaction'-field on a money transfer
    form.
-->
<!ELEMENT token_ver_record (token, infomsg?)>

<!--
    For token confirmation this record might be returned to
    its owner over the secure CIDAS-connection.
    The (user | service)-part identifies the person who
    got the token from its owner.
    Its infomsg-tag is used in a similar manner as the
    'reason for transaction'-field on a money transfer
    form.
-->
<!ELEMENT token_peer_record (token, (user | service), infomsg?)>

<!--
    The following record is returned to the user or service
    after successful validation of a token. Its subtags
    have the same meanings as within the token_ver_record,
    the guid might be used by the user or service to store
    data belonging to the user whose token was validated.
-->
<!ELEMENT token_ver_succ (token, infomsg?, guid)>

<!--
    In case the verification of a token failed and the owner
    of the invalid token could be determined, he gets this
    message explaining why the verification failed.
-->
<!ELEMENT token_perm_den    (token, (user | service), infomsg)>

```

```

<!--
    The next group of tags is used for specifying
    identification methods for CIDAS_MTYPE_IDDATAREQ -
    messages.
    An CIDAS_MTYPE_IDDATAREQ always contains an element
    of the type 'idmethods', containing one or several
    generic methods combined by boolean operators.
-->
<!ELEMENT idmethods ((idmethods_or? | idmethods_and? |
    idmethods_xor?)? |
    (idmethods_username? |
    idmethods_email? |
    idmethods_city? | idmethods_zip? |
    idmethods_street? |
    idmethods_realname? |
    idmethods_newattr?)?
)>

<!-- The usable boolean operators: -->
<!-- idmethods_or: At least one of the mentioned
    attributes must be set. -->
<!ELEMENT idmethods_or ((idmethods_or* |
    idmethods_xor* |
    idmethods_and* |
    idmethods_not*)*,
    idmethods_username?,
    idmethods_email?,
    idmethods_city?,
    idmethods_zip?,
    idmethods_street?,
    idmethods_realname?,
    idmethods_newattr*
)>

<!-- idmethods_xor: Exactly one of the mentioned
    attributes must be set. -->
<!ELEMENT idmethods_xor ((idmethods_or* |
    idmethods_xor* |
    idmethods_and* |
    idmethods_not*)*,
    idmethods_username?,
    idmethods_email?,

```

```

        idmethods_city?,
        idmethods_zip?,
        idmethods_street?,
        idmethods_realname?,
        idmethods_newattr*
    )>

<!-- idmethods_and: All of the mentioned attributes
      must be set. -->
<!ELEMENT idmethods_and ((idmethods_or* |
        idmethods_xor* |
        idmethods_and* |
        idmethods_not*)*,
        idmethods_username?,
        idmethods_email?,
        idmethods_city?,
        idmethods_zip?,
        idmethods_street?,
        idmethods_realname?,
        idmethods_newattr*
    )>

<!-- idmethods_not: Negation of another expression. -->
<!ELEMENT idmethods_not ((idmethods_or? |
        idmethods_xor? |
        idmethods_and?)?,
        idmethods_username?,
        idmethods_email?,
        idmethods_city?,
        idmethods_zip?,
        idmethods_street?,
        idmethods_realname?,
        idmethods_newattr*
    )>

<!-- A couple of tags specifying the what the server
      could accept as identification data: -->
<!ELEMENT idmethods_username EMPTY>
<!ELEMENT idmethods_email    EMPTY>
<!ELEMENT idmethods_city     EMPTY>
<!ELEMENT idmethods_zip      EMPTY>
<!ELEMENT idmethods_street   EMPTY>
<!ELEMENT idmethods_realname EMPTY>

```

```

<!-- Administrators should be able to define new
      attributes.
      They contain descriptions in several languages
      within the infomsg-tag. -->
<!ELEMENT idmethods_newattr (
      idmethods_newattr_name,
      infomsg)>

<!-- Attributes should always have a name. -->
<!ELEMENT idmethods_newattr_name (#PCDATA)>

<!--
The following structure is designed to keep the final
identification data in a CIDAS_MTYPE_IDDATA - message.
The payload of this type of message always contains
an 'iddata'-structure containing one or more
datafields. The combination of datafields must be one
allowed by the CIDAS_MTYPE_IDDATAREQ received before
sending the CIDAS_MTYPE_IDDATA message.
-->
<!ELEMENT iddata (iddata_username?,
      iddata_email?,
      iddata_city?, iddata_zip?, iddata_street?,
      iddata_realname?, iddata_newattr*)>

<!-- These are the tags to be used for transmitting
      identification data: -->
<!ELEMENT iddata_username (#PCDATA)>
<!ELEMENT iddata_email    (#PCDATA)>
<!ELEMENT iddata_city     (#PCDATA)>
<!ELEMENT iddata_zip      (#PCDATA)>
<!ELEMENT iddata_street   (#PCDATA)>
<!ELEMENT iddata_realname (#PCDATA)>

<!-- Administrators should be able to define new
      attributes: -->
<!ELEMENT iddata_newattr (iddata_newattr_name,
      iddata_newattr_value)>
<!ELEMENT iddata_newattr_name (#PCDATA)>
<!ELEMENT iddata_newattr_value (#PCDATA)>

```

```

<!--
    CIDAS allows the use of different authentication methods.
    Each method has a name and belongs to a class of methods.
    A list of valid classes and method names is given in the
    formal protocol specification.
-->
<!ELEMENT authmethod (authmethod_class, authmethod_name)>
    <!ELEMENT authmethod_class (#PCDATA)>
    <!ELEMENT authmethod_name (#PCDATA)>

<!--
    In a CIDAS_MTYPE_AUTHREQ-message the client may
    specifies a list of authentication methods to use.
-->
<!ELEMENT authmethods (authmethod+)>

<!--
    Within the CIDAS_MTYPE_AUTHDATAREQ the the server requests
    authentication data for a single authentication method
    from the client. The server may pass additional
    information, such as a question (i.e. for the
    TEXT:RANDDATA method) to answer or information
    related to the data requested, to the client.
-->
<!ELEMENT authdatareq (authmethod, authdatareq_bytestream?,
    infomsg?)>
    <!-- Authentication data requests may contain additional
    data such as random data or messages for zero-
    knowledge proofs. This data is always Radix-64
    encoded. -->
    <!ELEMENT authdatareq_bytestream (#PCDATA)>

<!--
    Authentication data is stored in the followind srtucture.

```

```

    The authentication method must be specified.
-->
<!ELEMENT authdata (authmethod, authdata_text? | authdata_bytestream?)>
    <!-- The authentication data itself contains always
         textual information. Binary data, for example
         for the biometric or cryptographic methods, is
         Radix-64 encoded. However, we are using different
tags for textual information and binary data.
To provide additional security
         authentication data might be encrypted with the
         servers public key. -->
    <ELEMENT authdata_text (#PCDATA)>
    <ELEMENT authdata_bytestream (#PCDATA)>

<!-- A capability (i.e. a special feature module) has
         at least a name, a version number and a vendor.
         It may contain some more verbose information. -->
<ELEMENT capability (capability_name,
                    capability_vendor,
                    capability_version,
                    capability_verbose?)>

    <ELEMENT capability_name    (#PCDATA)>
    <ELEMENT capability_vendor  (#PCDATA)>
    <ELEMENT capability_version (#PCDATA)>

    <!-- The verbose description of a capability contains
         a multilingual description of this capability as
         well as it's options and returns explained using SOAP
         (c.f. Simple Object Access Protocol 1.2). -->
    <ELEMENT capability_verbose (
                            infomsg,
                            soapdata?)>

<!--
    The following tag is used to store SOAP-structures
    for the communication between CIDAS clients and SFMs
    running within the servers process space.
-->
<ELEMENT soapdata    (SOAP-ENV:Envelope+)>

```

```

<!--
    The message types CIDAS_MTYPE_CAPREQ and
    CIDAS_MTYPE_CAPDATA are using a structure called
    'capdata' for transferring necessary
    information. This structure is defined as follows:
-->
<!ELEMENT capdata (capability*)>

<!--
    Within CIDAS_MTYPE_SFMREQ and CIDAS_MTYPE_SFMRESP
    we use:
-->
<!ELEMENT sfmdata    (capability, soapdata)>

<!--
    Beside from giving information about installed
    capabilities, the server is able to deliver some data
    about itself.
-->
<!ELEMENT serverrecord (csid,
                        serverrecord_pubkey,
                        serverrecord_product,
                        serverrecord_vendor,
                        serverrecord_version,
                        serverrecord_provider,
                        serverrecord_services?)>

<!-- Each server has a OpenPGP (RFC 2440) keypair. It
    might be used for encryption of sensitive data
    exchanged using possibly unsecure links. -->
<!ELEMENT serverrecord_pubkey (#PCDATA)>

<!-- Who wrote the server? -->
<!ELEMENT serverrecord_product (#PCDATA)>
<!ELEMENT serverrecord_vendor (#PCDATA)>
<!ELEMENT serverrecord_version (#PCDATA)>

<!-- Who runs this server? -->
<!ELEMENT serverrecord_provider (#PCDATA)>

```

```

    <!-- What services/applications do use this server
           as their identity provider -->
    <!ELEMENT serverrecord_services (service+)>

<!--
    The following structures are used within the exchange of
    user and application data.
-->

<!--
    Requests for reading and writing data.
-->
<!ELEMENT datareadrequest      (token, userdata* appdata* infomsg?)>
<!ELEMENT datawriterequest     (token, userdata* appdata* infomsg?)>

<!--
    Structures related to granting and declining access
    to the user's personal data.
-->
<!ELEMENT userdaccrequest      ((service | user), (datareadrequest |
                                datawriterequest), infomsg?)>
<!ELEMENT userdaccinfo         ((service | user), (datareadrequest |
                                datawriterequest), infomsg?)>
<!ELEMENT userdaccperm         ((service | user), userdata*)>

<!--
    Data acquisition.
-->
<!ELEMENT userdataacq          ((service | user), (userdata* |
                                data_key))>

<!--
    The final response to the application who requested
    data.
-->
<!ELEMENT dataresponse         (token, userdata* appdata*)>

<!--
    A passphrase might be used to encrypt data on the

```

```

        CIDAS-Server.
-->
<!ELEMENT data_key          (#PCDATA)>

<!--
    User and application data look pretty much the same,
    just the place where they are stored on the server is
    different.
-->
<!ELEMENT userdata          (data_attribute+)>
<!ELEMENT appdata           (data_attribute+)>

<!--
    Attributes usually have a name and a value:
-->
<!ELEMENT data_attribute    (data_attribute_name,
                             data_attribute_value?)
    <!ELEMENT data_attribute_name (#PCDATA)>
    <!ELEMENT data_attribute_value (#PCDATA)>

]>

```

Anhang C

cidas_1-0.h

Die Datei `cidas_1-0.h` ist eine C Header-Datei für CIDAS. In ihr sind alle Nachrichtentypen, Nachrichtenaufbauten, etc. für die in diesem Dokument beschriebene Version 1.0 des CIDAS-Protokolls definiert. Die Datei kann mittels der Anweisung

```
#include "cidas_1-0.h"
```

in beliebige Client- oder Serverimplementierungen von CIDAS, sofern diese in C geschrieben sind, aufgenommen werden.

```
#ifndef _CIDAS_1_0_H
#define _CIDAS_1_0_H
```

```
/*
 * This file defines several constants used within
 * version 1.0 of the CIDAS-protocol for communication
 * between a CIDAS-server and its clients.
 */
```

```
/*
 * Protocol version:
 */
#define CIDAS_PROTOCOL_MAJORVERSION 1
#define CIDAS_PROTOCOL_MINORVERSION 0
```

```
/*
 * The message types and their hex-values:
 */
```

```

#define CIDAS_MTYPE_CONREQ                0x01
#define CIDAS_MTYPE_CONACC                0x02
#define CIDAS_MTYPE_IDDATAREQ            0x03
#define CIDAS_MTYPE_IDDATA                0x04
#define CIDAS_MTYPE_AUTHREQ              0x05
#define CIDAS_MTYPE_AUTHDATAREQ          0x06
#define CIDAS_MTYPE_AUTHDATA              0x07
#define CIDAS_MTYPE_AUTHSUCC              0x08
#define CIDAS_MTYPE_AUTHFAIL              0x09
#define CIDAS_MTYPE_TOKREQ                0x0a
#define CIDAS_MTYPE_TOKDATA                0x0b
#define CIDAS_MTYPE_TOKFAIL                0x0c
#define CIDAS_MTYPE_TOKVERREQ              0x0d
#define CIDAS_MTYPE_TOKCONFREQ            0x0e
#define CIDAS_MTYPE_TOKCONFACC            0x0f
#define CIDAS_MTYPE_TOKCONFDEC            0x10
#define CIDAS_MTYPE_TOKVERSUC              0x11
#define CIDAS_MTYPE_TOKVERFAIL            0x12
#define CIDAS_MTYPE_TOKPERMDEN            0x13
#define CIDAS_MTYPE_CAPREQ                0x14
#define CIDAS_MTYPE_CAPDATA                0x15
#define CIDAS_MTYPE_SFMREQ                0x16
#define CIDAS_MTYPE_SFMRESP                0x17
#define CIDAS_MTYPE_CSRREQ                0x18
#define CIDAS_MTYPE_CSRDATA                0x19
#define CIDAS_MTYPE_DATAREADREQ            0x1a
#define CIDAS_MTYPE_DATAWRITEREQ          0x1b
#define CIDAS_MTYPE_USERDACCREQ            0x1c
#define CIDAS_MTYPE_USERDACCINFO          0x1d
#define CIDAS_MTYPE_USERDACCPerm          0x1e
#define CIDAS_MTYPE_USERDATAREQ            0x1f
#define CIDAS_MTYPE_USERDATA                0x20
#define CIDAS_MTYPE_DATARESP                0x21
#define CIDAS_MTYPE_KAREQ                  0x22
#define CIDAS_MTYPE_KAANS                  0x23
#define CIDAS_MTYPE_INFOMSG                0x24
#define CIDAS_MTYPE_LOGOUT                0x25

/*
 * Flags:
 */
#define CIDAS_FLAGS_COMPRESSION            0x80
#define CIDAS_FLAGS_RADIX64                0x40

```

```
#define CIDAS_FLAGS_TESTING1          0x01
#define CIDAS_FLAGS_TESTING2          0x02

#endif /* _CIDAS_1_0_H */
```

Anhang D

GNU Free Documentation License

GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document „free“ in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of „copyleft“, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published

as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The „**Document**“, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as „**you**“. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A „**Modified Version**“ of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A „**Secondary Section**“ is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The „**Invariant Sections**“ are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The „**Cover Texts**“ are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A „**Transparent**“ copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable

for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not „Transparent“ is called „**Opaque**“.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The „**Title Page**“ means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, „Title Page“ means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section „**Entitled XYZ**“ means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as „**Acknowledgements**“, „**Dedications**“, „**Endorsements**“, or „**History**“.) To „**Preserve the Title**“ of such a section when you modify the Document means that it remains a section „Entitled XYZ“ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or

control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling

the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled „History“, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled „History“ in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on.

These may be placed in the „History“ section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled „Acknowledgements“ or „Dedications“, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled „Endorsements“. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled „Endorsements“ or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled „Endorsements“, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled „History“ in the various original documents, forming one section Entitled „History“; likewise combine any sections Entitled „Acknowledgements“, and any sections Entitled „Dedications“. You must delete all sections Entitled „Endorsements“.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an „aggregate“ if the copyright resulting from

the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled „Acknowledgements“, „Dedications“, or „History“, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will

be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License „or any later version“ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Anhang E

Ehrenwörtliche Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbstständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Brandenburg an der Havel, 3. März 2004

Jan Tobias Mühlberg