

Technische Universität Darmstadt

**Fachbereich Informatik
Fachgebiet Theoretische Informatik**

Prof. Dr. J. Buchmann

**Entwicklung einer sicheren
Web-Schnittstelle
für eine bestehende Enterprise-Anwendung
basierend auf
PKI-Technologien am Beispiel der
ERP-Software ProAd**

Diplomarbeit von Daniel Kwiotek

In Zusammenarbeit mit der

**J+D Software AG
in Dietzenbach**

Betreuer TUD: Marcus Lippert

Betreuer J+D: Marcus Bichowicz



Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dietzenbach, den 12.09.2002

Daniel Kwiotek
daniel@kwiotek-online.de

Die in diesem Dokument erwähnten Soft- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen.

Vorwort

Die hier vorliegende Arbeit untersucht die Absicherungsmöglichkeiten der Internet-Kommunikation unter Verwendung von Public Key Infrastrukturen. Als konkretes Problem wird der Betrieb des ERP-Systems ProAd über das Internet behandelt. Dabei wird eine Webschnittstelle für das System entworfen und ein Prototyp implementiert.

Die Arbeit wurde in Kooperation zwischen dem Fachgebiet Theoretische Informatik an der Technischen Universität Darmstadt und der J+D Software AG in Dietzenbach angefertigt.

Besonderer Dank gilt Marcus Lippert, der in zahlreichen Fachgesprächen keine Frage unbeantwortet ließ, und Marcus Bichowicz, der bei der alltäglichen Arbeit am Prototyp und bei der schriftlichen Ausarbeitung in den Räumen der J+D Software AG für Hilfestellung immer offen war.

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis.....	III
Abkürzungsverzeichnis.....	IV
1. Einleitung.....	1
2. Sichere digitale Kommunikation mit PKI.....	3
2.1 Sichere digitale Kommunikation	3
2.2 Kryptographische Grundlagen.....	3
2.2.1 Symmetrische Verschlüsselung	4
2.2.2 Asymmetrische Verschlüsselung.....	5
2.2.3 Digitale Signatur und digitales Zertifikat	6
2.2.4 Hashing	8
2.3 PKI im Überblick.....	8
2.3.1 Aufbau.....	9
2.3.2 Dienste	10
2.3.3 Sicherheitsniveau	11
2.3.4 Rechtliche Grundlagen.....	12
3. Überblick über ProAd	14
3.1 Technische Plattform	14
3.2 Funktionalität	14
3.3 Nächste Generation.....	16
4. Sichere Webintegration von ProAd mit PKI	18
4.1 Vorgaben und Anforderungen	18
4.2 Architektur der Webanwendung.....	18
4.2.1 Präsentationsebene	19
4.2.2 Anwendungsebene	20
4.3 Sicherheitsniveau der eingesetzten PKI.....	24
5. Entwicklung eines Prototyps.....	26
5.1 Umgesetzte Konzepte und eingesetzte Technologien.....	26
5.2 Authentisierung.....	28
5.2.1 Konzept.....	28
5.2.2 Angewandte Technologie	29
5.2.3 Implementierung	31
5.3 Autorisierung	32
5.3.1 Konzept von ProAd.....	32
5.3.2 Konzept von Java	33
5.3.3 Implementierung	34
6. Fazit.....	36
Quellennachweis	38
Literatur.....	38
Request for Comments.....	39

Internet	39
Anhang	42
Installation und Konfiguration	42
Programmcode	44

Abbildungsverzeichnis

Abbildung 1: Symmetrische Verschlüsselung.....	4
Abbildung 2: Asymmetrische Verschlüsselung.....	5
Abbildung 3: Digitale Signatur.....	6
Abbildung 4: X.509 Format für digitale Zertifikate Version 3.....	7
Abbildung 5: Sichern der Integrität durch Hashing.....	8
Abbildung 6: Der Aufbau einer PKI.....	9
Abbildung 7: Flussdiagramm zum Ablauf eines Jobs.....	15
Abbildung 8: Das Hierarchiemodell von ProAd.....	16
Abbildung 9: Architektur der Webanwendung im Überblick.....	19

Abkürzungsverzeichnis

AES -	Advanced Encryption Standard
AJP -	Apache Jserv Protocol
CP -	Certificate Policy
CPS -	Certification Practices Statement
DES -	Data Encryption Standard
EDV -	Elektronische Datenverarbeitung
ERP -	Enterprise Resource Planning
HTML -	Hypertext Markup Language
HTTP -	Hypertext Transport Protocol
IDEA -	International Data Encryption Algorithm
JAAS -	Java Authentication and Authorization Service
JDBC -	Java Database Connectivity
JDK -	Java Development Kit
JSP -	Java Server Pages
LAN -	Local Area Network
LDAP -	Lightweight Directory Access Protocol
OID -	Object Identifier
PKI -	Public Key Infrastructure
RAD -	Rapid Application Development
RFC -	Request For Comments
SQL -	Standard Query Language
SSL -	Secure Socket Layer
TLS -	Transport Layer Security
WAN -	Wide Area Network
XML -	Extensible Markup Language

1. Einleitung

Auf den Einsatz der elektronischen Datenverarbeitung (EDV) kann im Geschäftsalltag nicht mehr verzichtet werden. Durch die Anwendung von Buchführungs- und Fakturierungsprogrammen sowie Planungs- und Controllingtools konnte die Effizienz und Transparenz der in einem Unternehmen ablaufenden Geschäftsprozesse gesteigert werden. Dabei hat sich gezeigt, dass der Einsatz von heterogenen Lösungen den Nachteil hat, dass die einzelnen Programme auf unterschiedlichen Datenbeständen arbeiten und nicht ohne weiteres miteinander kommunizieren können. Dieser Nachteil konnte durch die Entwicklung von integrierten Systemen zur Unterstützung der Planung, Steuerung und Kontrolle der Geschäftsprozesse einer Unternehmung beseitigt werden. Solche Systeme werden oft mit dem Begriff *Enterprise Resource Planning System* (ERP-System) bezeichnet.¹ Das in Deutschland wohl bekannteste ERP-System ‚R3‘ stammt von der Firma SAP.²

ERP-Systeme werden oft in lokalen firmeninternen Netzwerken (*Local Area Network*, LAN) betrieben. Sobald ein Unternehmen aus mehreren Niederlassungen besteht, stellt sich die Frage nach der Vernetzung der lokalen Netzwerke zu einem *Wide Area Network* (WAN), so kann das ERP-System unternehmensweit, standortübergreifend eingesetzt werden. Bis vor kurzem war das Mieten von teuren Standleitungen, die sich nur größere Unternehmen und Konzerne leisten konnten, die einzige Möglichkeit ein WAN zu betreiben. Mit dem Internet entstand eine Alternative, die unabhängig von Ort, Zeit und eingesetztem Betriebssystem für jedermann zu erschwinglichen Kosten verfügbar ist. Der Erfolg des Internets führte dazu, dass Internet-Technologien inzwischen beim Aufbau von LANs - den sogenannten Intranets - und WANs - den sogenannten Extranets - Einsatz finden. Damit ist es möglich geworden, auf eine in einem LAN installierte Software, die auf Internet-Technologien basiert, sowohl lokal als auch aus einem fernen Netzwerk zuzugreifen. Der Zugriff erfolgt für den Anwender transparent, es ist also nicht erkennbar, ob das Programm lokal oder vom Standort entfernt installiert ist.³

Damit scheint das Internet eine geeignete Plattform für den Betrieb eines ERP-Systems zu sein. Allerdings hat die Kommunikation über das Internet folgendes Merkmal, das bisher unerwähnt blieb: Falls keine besonderen Maßnahmen ergriffen werden, um eine solche Kommunikation abzusichern, ist jeder im Internet in der Lage den gesamten ausgetauschten Datenstrom mitzulesen, zu manipulieren oder unter einer falschen Identität an der Kommunikation teilzunehmen. Dies stellt ein großes Hindernis dar, um das Internet mit allen seinen Vorteilen als eine Plattform für ERP-Systeme zu beanspruchen, die über einen hoch sensiblen Datenbestand verfügen, der selbst von allen regulären Anwendern nicht vollständig eingesehen werden darf. Da das Hindernis alle Applikationen betrifft, die eine sichere Kommunikation über das Internet erfordern, bestehen inzwischen in der Internet-Branche viele Bemühungen, Technologien zu entwickeln, die es ermöglichen bestimmte Bereiche des Internets in ein sicheres Netzwerk umzuwandeln. Einen verbreiteten Einsatz finden dabei Technologien, die Konzepte der sogenannten *Public Key Infrastructure* (PKI) anwenden. Eine wesentliche Idee von PKI ist die Existenz einer vertrauenswürdigen

¹ Vgl. Günther / Tempelmeier, 2000, S. 313

² Vgl. SAP Web, 2002

³ Vgl. Edwards / Harkey / Orfali, 1997, S. 11

dritten Partei (*Trusted Third Party*), die von allen Teilnehmern der Infrastruktur anerkannt ist. Diese vertrauenswürdige dritte Partei stellt die Identität aller Teilnehmer sicher und vergibt Zertifikate, in der Art eines digitalen Ausweises. Die Daten der Zertifikate werden in einer PKI eingesetzt, um mittels kryptographischen Verfahren eine sichere Kommunikation in einem unsicheren Netzwerk zu gewährleisten.

Das Ziel dieser Diplomarbeit ist es, zu untersuchen, inwieweit PKI dazu geeignet ist ein bestehendes ERP-System abzusichern, welches um die Kommunikationsfähigkeit über Internetprotokolle erweitert werden soll. Als Beispiel-ERP-System soll dabei ProAd Professional, kurz ProAd, der J+D Software AG aus Dietzenbach dienen. Für dieses System soll eine Architektur einer sicheren Webintegration mit PKI entworfen und darauf basierend ein Prototyp implementiert werden.

In dieser Ausarbeitung wird anfangs die sichere digitale Kommunikation mit PKI erörtert. Dabei wird die Sicherheit der digitalen Kommunikation definiert sowie kryptographische Grundlagen und das Konzept von PKI erläutert. Anschließend werden die Konzepte von ProAd und die einer möglichen nächsten Generation der Software beschrieben. Anschließend wird mit der Architektur der sicheren Webintegration von ProAd mittels PKI ein zentraler Punkt der Diplomarbeit beschrieben. Dabei werden zuerst die Anforderungen definiert, gefolgt von einer Beschreibung der Architektur. Des Weiteren werden hier Aspekte beschrieben, deren Erfüllung die Gewährleistung des geforderten Sicherheitsniveaus der eingesetzten PKI sicher stellt. Einen weiteren zentralen Punkt bildet die Implementierung eines Prototyps, der auf der entworfenen Architektur basiert und mit der Zeit- und Materialerfassung einen abgeschlossenen Teil der Geschäftslogik implementiert. Neben den eingesetzten technischen Komponenten steht dabei insbesondere die Authentisierung und Autorisierung im Mittelpunkt.

2. Sichere digitale Kommunikation mit PKI

Dieses Kapitel stellt die theoretische Grundlage für den Entwurf einer mit PKI abgesicherten Anwendung dar. Im ersten Abschnitt werden Eigenschaften beschrieben, die eine sichere digitale Kommunikation charakterisieren. Danach werden die Grundlagen kryptographischer Verfahren erläutert, welche eine sichere digitale Kommunikation ermöglichen. Anschließend wird mit PKI ein Konzept beschrieben, das die erläuterten kryptographischen Verfahren auf eine eigene Art und Weise einsetzt und für die Absicherung der Webintegration von ProAd eingesetzt wird.

2.1 Sichere digitale Kommunikation

Die Sicherheit in der Informationstechnologie (IT) kann, ähnlich wie die Sicherheit in anderen Disziplinen, niemals absolut garantiert werden. Dies resultiert daraus, dass jede Handlung neben den erhofften Chancen potentielle Risiken birgt. Eine absolut sichere Umgebung erlaubt keine Handlung und eliminiert damit alle Möglichkeiten potentielle Chancen zu nutzen. Die Herausforderung der IT-Sicherheit besteht also darin, ein funktionsfähiges System mit einem hohen Sicherheitsgrad zu gewährleisten. Der Sicherheitsgrad eines IT-Systems kann durch die Überprüfung von folgenden Schlüsselmerkmalen festgestellt werden: Authentisierung und Autorisierung, Vertraulichkeit, Integrität und Nicht-Abstreitbarkeit.⁴

Die Authentisierung stellt sicher, dass nur ein ausgewählter Anwenderkreis Zugang zum System erhält. Der Anwender muss sich identifizieren und einen Beweis erbringen, dass er derjenige ist für den er sich ausgibt. Das bekannteste Verfahren der Authentisierung fordert den Anwender auf, dem System seinen Namen und ein oder mehrere dazugehörige Passwörter mitzuteilen. Weitere Verfahren können auf digitaler Signatur und digitalem Zertifikat oder auf biometrischen Merkmalen aufbauen.⁵ Ein authentisierter Anwender kann autorisiert werden, d.h. auf den Besitz von Rechten überprüft werden, Zugang zu allen oder bestimmten Bereichen eines Systems erhalten zu dürfen. Vertraulichkeit meint die Gewährleistung einer Kommunikation zwischen dem System und einem authentisierten Anwender, die von einer weiteren Partei nicht belauscht werden kann. Dies geschieht vor allem durch Verschlüsselung. Eine integrale Kommunikation ist gegen die Modifikation der getauschten Informationen geschützt. Die Integrität wird durch kryptographische Funktionen wie *Message Authentication Code* (MAC) und die digitale Signatur sichergestellt. Nicht-Abstreitbarkeit erlaubt im Nachhinein den Beweis erbringen zu können, dass eine bestimmte Nachricht von einem bestimmten Kommunikationsteilnehmer stammt. Nicht-Abstreitbarkeit kann mittels digitaler Signatur sichergestellt werden.⁶

2.2 Kryptographische Grundlagen

Ein Verschlüsselungsverfahren erzeugt aus einem Klartext mittels eines speziellen Algorithmuses Geheimtext (auch „Schlüsseltext“ genannt). Dies geschieht in Abhängigkeit von einem Schlüssel, der als Parameter für den Algorithmus dient.⁷ Ein wichtiges Prinzip in der Kryptographie ist, dass die Sicherheit der Verschlüsselung

⁴ Vgl. Austin, 2001, S. 24

⁵ Vgl. Austin, 2001, S. 8-12

⁶ Vgl. Stallings, 1995, S. 18

⁷ Vgl. Seberry / Pieprzyk, 1989, S. 2-3

nur von der Geheimhaltung des Schlüssels abhängt und nicht von der Geheimhaltung des Verfahrens (auch als „Kerckhoffs Prinzip“ bekannt). Dadurch entfällt bei auftretenden Problemen eine - gegenüber der Änderung des Schlüssels - deutlich aufwendigere Änderung des Algorithmuses.⁸

Allgemein wird zwischen symmetrischen und asymmetrischen Verschlüsselungsverfahren unterschieden. Durch gezielten Einsatz von Verschlüsselungsverfahren in Verbund mit Hashing wird die digitale Unterschrift und digitale Signatur ermöglicht. Dies wird im folgenden genauer erläutert.

2.2.1 Symmetrische Verschlüsselung

Bei den symmetrischen Verschlüsselungsverfahren dient ein einziger geheimer Schlüssel zum Ver- und Entschlüsseln. Die Haupteinsatzgebiete sind das verschlüsselte Aufbewahren lokaler Daten sowie die Verschlüsselung von Daten bei der Übertragung über eine unsichere Kommunikationsleitung.⁹

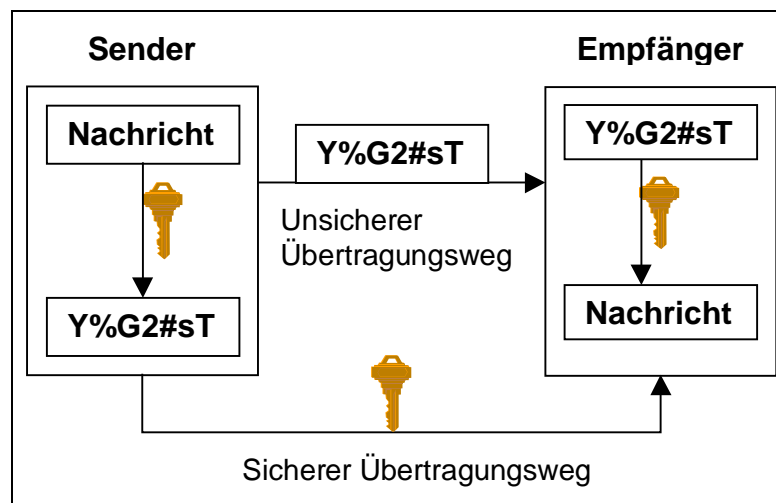


Abbildung 1: Symmetrische Verschlüsselung

Die symmetrischen Verfahren sind im Allgemeinen sehr effizient in Software oder Hardware zu implementieren, da für den Computer einfache Operationen verwendet werden können. Deshalb ist der Vorteil dieser Verfahren, dass der Vorgang der Ver- und Entschlüsselung sehr wenig Zeit in Anspruch nimmt. Problematisch ist dagegen der Austausch des geheimen Schlüssels. Dazu muss ein sicherer Übertragungskanal existieren, über den, vor der eigentlichen, verschlüsselten Kommunikation, der geheime Schlüssel übertragen wird, ohne dass ein Dritter mithören kann. Solch einen Übertragungskanal zu finden, kann unter Umständen sehr schwierig sein.¹⁰

Ein weiteres Problem ist die mangelnde Skalierbarkeit. Für 1000 Kommunikationspartner sind 1000 verschiedene geheime Schlüssel notwendig, die verwaltet werden müssen. Für eine geschlossene Gruppe von 1000 Personen, in der jede mit jeder kommunizieren möchte, muss also eine Gesamtzahl in der

⁸ Vgl. Seberry / Pieprzyk, 1989, S. 5

⁹ Vgl. Rihaczek, 1984, S. 12

¹⁰ Vgl. Adams / Lloyd, 1999, S. 15

Größenordnung von ungefähr $n^2/2 = 500000$ Schlüsseln (inkl. des eigenen) gespeichert werden.¹⁰

Eine Angriffsmöglichkeit besteht darin, durch Ausprobieren aller möglichen Schlüssel, den richtigen zu erraten. Je länger ein Schlüssel ist (d.h. je mehr Bits er umfasst), desto länger dauert das Ausprobieren. Für einen Schlüssel mit einer Länge von 64 Bit müsste man maximal $2^{64} = 1,8 \cdot 10^{19}$ Bitkombinationen testen. Durch den Einsatz einer großen Anzahl von Rechnern, die parallel daran arbeiten und speziell dafür konstruierten Computern, lässt sich der Zeitaufwand allerdings drastisch reduzieren. Deshalb gelten bei den symmetrischen Verfahren erst 128 Bit als sicher.

Die bekanntesten Verfahren sind DES (*Data Encryption Standard*) bzw. Triple DES (3DES)¹¹, AES (*Advanced Encryption Standard*)¹², die Rivest Cipher Verfahren (RC2, RC4, RC5, RC6)¹³, welche im SSL-Protokoll (SSL – *Secure Socket Layer*)¹⁴ eingesetzt werden, sowie IDEA (*International Data Encryption Algorithm*)¹³.

2.2.2 Asymmetrische Verschlüsselung

Die asymmetrischen Verfahren unterscheiden sich von den symmetrischen dadurch, dass zwei unterschiedliche Schlüssel zum Ver- und Entschlüsseln verwendet werden: Ein öffentlicher Schlüssel („Public Key“), der allgemein bekannt gemacht werden kann und ein privater, geheimer Schlüssel („Private Key“). Wenn eine vertrauliche Nachricht übermittelt werden soll, codiert der Sender diese mit dem Public Key des Empfängers, den er zuvor auf sichere Weise geholt hat und verschickt sie an den Empfänger. Ausschließlich dieser kann die Nachricht mit seinem Private Key decodieren. Das bei der symmetrischen Kryptographie auftauchende Problem der Schlüsselübermittlung ist damit beseitigt.¹⁵

Asymmetrische Kryptographie ermöglicht, außer Verschlüsselung – und damit Wahrung der Vertraulichkeit – zusätzlich die digitale Signierung von Dokumenten.¹⁶

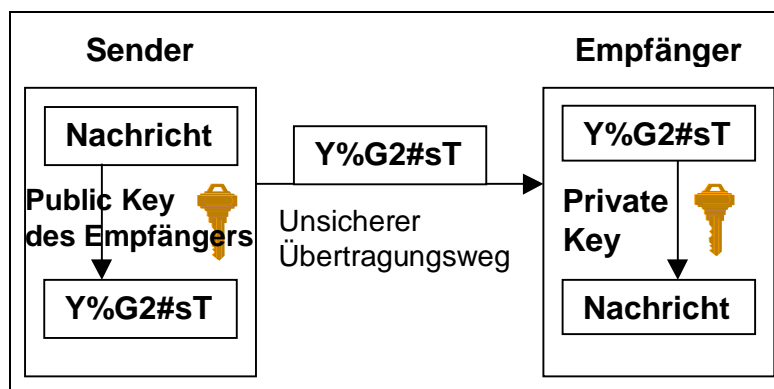


Abbildung 2: Asymmetrische Verschlüsselung

¹¹ Vgl. National Bureau of Standards and Technology, 1999

¹² Vgl. AES Web, 2002

¹³ Vgl. RSA Security, 2002a

¹⁴ Spezifikation in Dierks / Allen, 1999, RFC 2246

¹⁵ Vgl. Adams / Lloyd, 1999, S. 16-17

¹⁶ Vgl. Kapitel 2.2.3

Asymmetrische Kryptographie basiert auf einer Funktion, die nur sehr schwer umzukehren ist (z.B. Produkt zweier Primzahlen). „Sehr schwer“ bedeutet in diesem Zusammenhang, dass es zum jetzigen Zeitpunkt und voraussichtlich auch in den nächsten Jahren aufgrund technischer Beschränkungen nicht möglich ist, solche Berechnungen innerhalb einer praktikablen Zeitspanne durchzuführen. Dadurch ergibt sich, dass - sobald es möglich werden sollte, eine solche Funktion auf einfache und schnelle Weise umzukehren - das asymmetrische Verfahren, das diese Funktion verwendet, unbrauchbar geworden ist.

Zum bekanntesten asymmetrischen Verschlüsselungsverfahren zählt RSA, das nach seinen Erfindern (Rivest, Shamir und Adleman) benannt wurde.¹⁷

Da asymmetrische Verfahren sehr zeitaufwendig sind, werden sie selten zur direkten Nachrichtenverschlüsselung verwendet. Stattdessen wird zu Beginn einer Kommunikation ein asymmetrisch codierter Sitzungsschlüssel übertragen, mit dem die darauffolgenden Daten mithilfe eines viel schnelleren symmetrischen Verfahrens codiert werden. Dies wird auch „Hybrides Verfahren“ genannt.

Der Ablauf in zwei Stufen:

1. Verschlüsselung der Daten mit symmetrischem Verfahren und einem zufällig generierten Schlüssel
2. Anschließendes Codieren des symmetrischen Schlüssel mit dem Public Key des Empfängers

2.2.3 Digitale Signatur und digitales Zertifikat

Die digitale Unterschrift ermöglicht die Sicherung der Integrität und, in Verbindung mit Zertifikaten, die Sicherstellung der Authentizität des Kommunikationspartners und auch die Nicht-Abstreitbarkeit.¹⁸

Eine digitale Signatur wird erst durch asymmetrische Verfahren und dem Konzept des Schlüsselpaars ermöglicht. Dabei werden die Daten mit dem Private Key verschlüsselt, der nur dem Besitzer dieses Schlüssels bekannt sein darf. Jeder der den Public Key dieser Person kennt, kann die Daten entschlüsseln und weiß dadurch, dass die Daten nur von einer Person kommen können, die den passenden Private Key besitzt.¹⁹

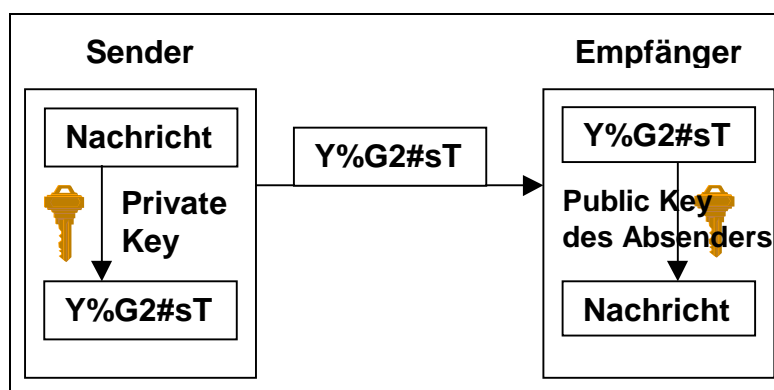


Abbildung 3: Digitale Signatur

¹⁷ Vgl. Rivest / Shamir / Adleman, 1978, S. 120-126

¹⁸ Vgl. Curry, 1997a, S. 1

¹⁹ Vgl. Adams / Lloyd, 1999, S. 20

Eine digitale Signatur entspricht einer handgeschriebenen Unterschrift insofern, dass eine einzelne Person Daten signiert und eine beliebige Anzahl von anderen Personen die Unterschrift lesen und deren Echtheit überprüfen können. Tatsächlich ist sie sogar sicherer als eine echte Unterschrift, da es computertechnisch praktisch unmöglich ist, eine digitale Unterschrift nachzumachen, ohne den privaten Schlüssel zu kennen.²⁰

Es ist jedoch wichtig zu erwähnen, dass die Daten mit dem eben beschriebenen Verfahren nur an das Schlüsselpaar (bzw. den Besitzer des Private Keys) gebunden ist, und nicht an die Identität einer Person. Die Bindung an die Identität wird erst durch digitale Zertifikate ermöglicht. Digitale Zertifikate dienen dazu, eine Person (oder ein Unternehmen) - und unter Umständen bestimmte Attribute, die damit zusammenhängen - mit einem Public Key zu verbinden. Damit wird einem Schlüsselpaar die Identität dieser Person zugeordnet. Die Zusammengehörigkeit des öffentlichen Schlüssels mit der Identität einer Person wird durch eine von allen anerkannte vertrauenswürdige dritte Partei (*Trusted Third Party*) bescheinigt.²¹ Dies wird durch eine digitale Unterschrift des Zertifikats durch die vertrauenswürdige dritte Partei vollzogen.

Das Format eines Zertifikats nach dem weit anerkannten Standard X.509 Version 3 zeigt die nachfolgende Abbildung.

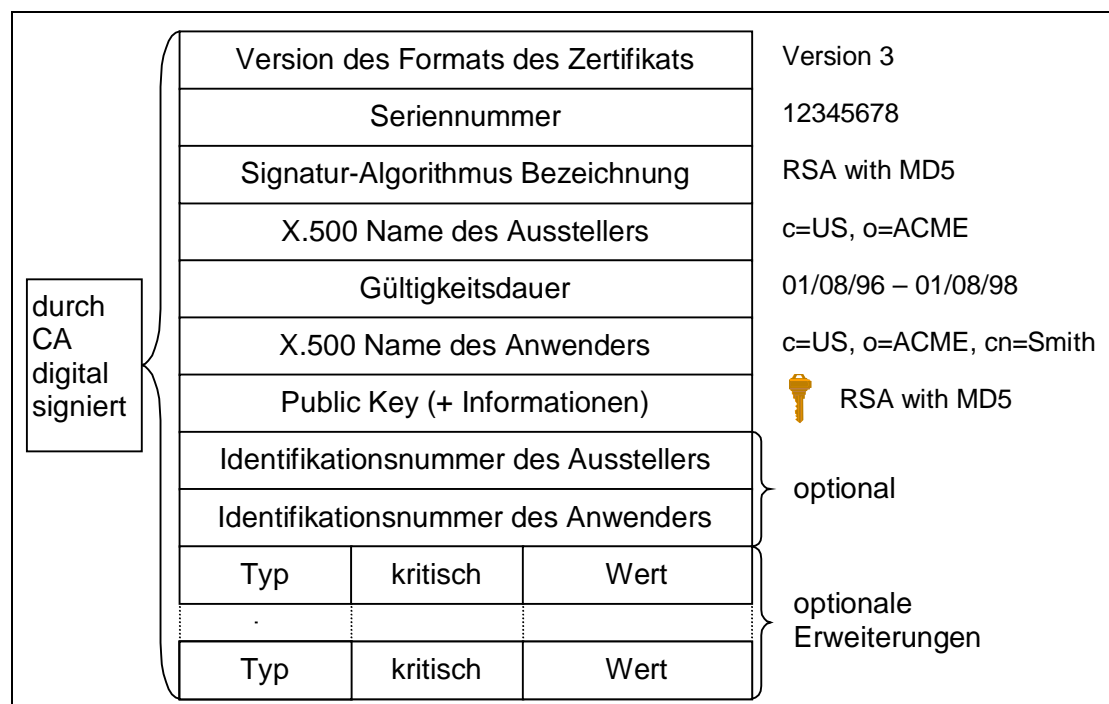


Abbildung 4: X.509 Format für digitale Zertifikate Version 3

Als weiteres Standard für Zertifikate sei hier PGP (*Pretty Good Privacy*) genannt.²²

²⁰ Vgl. Adams / Lloyd, 1999, S. 20

²¹ Vgl. Adams / Lloyd, 1999, S.75

²² Vgl. Stallings, 1995, S. 445

2.2.4 Hashing

In Praxis der digitalen Unterschrift wird aus Sicherheits- und Geschwindigkeitsgründen nicht der gesamte Text verschlüsselt, sondern zuerst ein sogenannter Hashwert über dem Text errechnet und dieser wird dann verschlüsselt.²³

Unter Hashing versteht man ein Verfahren, in dem einem Dokument (oder allgemein Daten) von beliebiger Größe ein Wert einer bestimmten Größe zugeordnet wird.

Dieser Hashwert entsteht unter Anwendung einer kryptographischen Hashfunktion, welche folgende Eigenschaften aufweist:²⁰

- Eine minimale Änderung der Ausgangsdaten ergibt einen völlig anderen Hashwert
- Aus dem Hashwert ist kein Rückschluß auf die Ausgangsdaten möglich, d.h. die Hashfunktion ist nicht umkehrbar
- Der Hashwert identifiziert die Ausgangsdaten, d.h. es ist praktisch unmöglich zwei verschiedene Dokumente mit dem gleichen Hashwert zu finden

Das Vorgehen zur Sicherstellung der Integrität sieht damit wie folgt aus:

1. bilden eines Hashwertes über die Daten
2. diesen Wert mit dem privaten Schlüssel signieren

Der Empfänger bildet ebenfalls den Hashwert des Dokumentes und vergleicht ihn mit dem mitgelieferten Hashwert, der zuvor mit dem Public Key des Absenders decodiert wurde. Wenn die beiden Prüfsummen nicht übereinstimmen, wurde das Dokument modifiziert. Wie bereits erwähnt, kann das Dokument nicht so modifiziert werden, dass danach der gleiche Hashwert herauskommt. Und andererseits kann der mitgelieferte Hashwert nicht den veränderten Daten angepasst werden, da er digital signiert wurde.

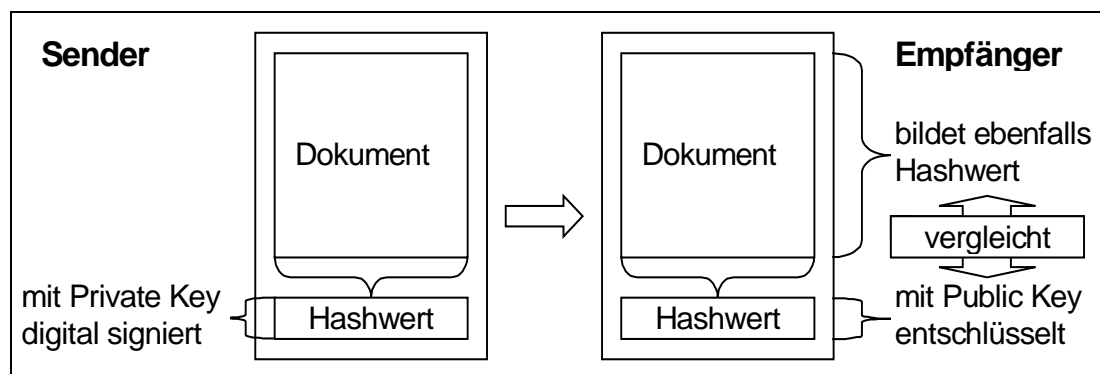


Abbildung 5: Sichern der Integrität durch Hashing

Bekannte Hash-Algorithmen sind MD4²⁴, MD5²⁵ und SHA-1 (*Secure Hash Standard*)²⁶.

2.3 PKI im Überblick

Eine PKI definiert eine Architektur für eine sichere²⁷ digitale Kommunikation und bedient sich dabei der, im vorherigen Kapitel beschriebenen, kryptographischen

²³ Vgl. Curry, 1997b, S. 3

²⁴ Spezifikation in Rivest, 1992a, RFC 1320

²⁵ Spezifikation in Rivest, 1992b, RFC 1321

²⁶ Spezifikation in National Bureau of Standards, 1995

Grundlagen. Im folgenden werden der Aufbau einer PKI und die gebotenen Dienste erläutert. Anschließend werden die Themenbereiche Sicherheitsniveau einer PKI und rechtliche Grundlagen beschrieben.

2.3.1 Aufbau

Eine PKI besteht aus mindestens den folgenden Elementen: Registrierungsstelle (*Registration Authority, RA*), Zertifizierungsstelle (*Certification Authority, CA*), Veröffentlichungsverzeichnis (*Certificate Repository*), den Anwendern, die Zertifikate beantragen sowie Anwendungen, welche sich der PKI-Funktionen bedienen, um einen hohen Sicherheitsgrad zu gewährleisten.

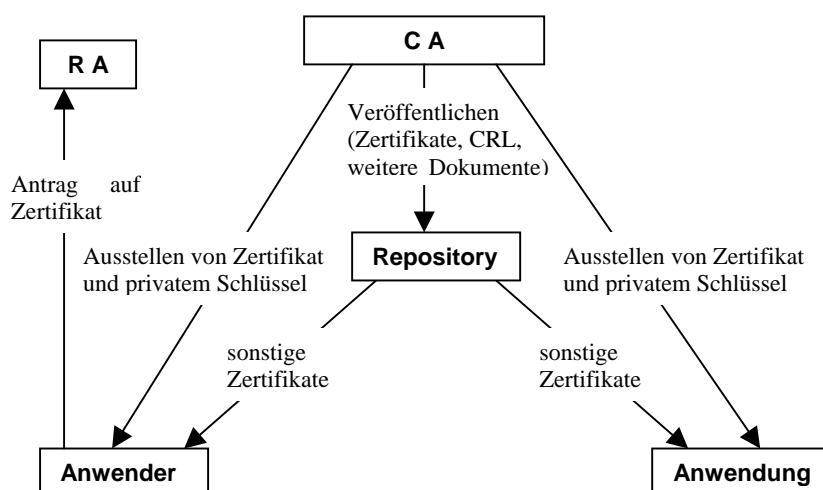


Abbildung 6: Der Aufbau einer PKI

Eine Zertifizierungsstelle stellt Zertifikate aus, d.h. sie bindet ein Schlüsselpaar an eine Identität. Bei sehr kleinen oder geschlossenen Umgebungen können die Benutzer selbst als Zertifizierungsstelle agieren.²⁸ Beispiele für kommerzielle Zertifizierungsstellen sind Verisign und Telesec (Zertifizierungsstelle der Deutschen Telekom). Bei der Registrierungsstelle findet eine Überprüfung der Identität der Antragsteller statt. Die Registrierungsstelle kann auch Teil der Zertifizierungsstelle sein. Ein in dem Zusammenhang oft verwendeter Begriff ist der Trustcenter. Bei geographisch weit verteilten End-Anwendern sind dezentrale Registrierungsstellen von Vorteil.²⁹

Das Veröffentlichungsverzeichnis dient zur Veröffentlichung von Zertifikaten und Zertifikatsrückruflisten (*Certificate Revocation List, CRL*). Des Weiteren können hier weitere Veröffentlichungen vorgenommen werden, beispielsweise die *Certificate Policy*, welche das Sicherheitsniveau der PKI genauer beschreibt.³⁰ Damit eine PKI erfolgreich eingesetzt werden kann, muss ein solcher Verzeichnisdienst hoch verfügbar sein. Im Allgemeinen benutzen diese Verzeichnisse das *Light-weight Directory Access Protocol (LDAP)*. Alternativen sind „X.500 Directory System

²⁷ Vgl. Kapitel 2.1

²⁸ Vgl. Adams / Lloyd, 1999, S. 34

²⁹ Vgl. Adams / Lloyd, 1999, S. 89

³⁰ Vgl. Kapitel 2.3.3

Agents“, Webserver (d.h. *Hypertext Transfer Protocol*, HTTP), *File Transfer Protocol* (FTP) oder Datenbanken.³¹

Anwender einer PKI sind Personen oder Systeme, welche ihre Identität während einer digitalen Kommunikation glaubwürdig erkennen lassen wollen. Ein potentieller Anwender einer PKI beantragt bei der Registrierungsstelle ein Zertifikat, das ihm samt des privaten Schlüssels nach der Überprüfung seiner Identität von der Zertifizierungsstelle zugestellt wird. Die Anwenderprogramme sind dazu verpflichtet ein Zertifikat sowie die dazugehörige digitale Signatur zu überprüfen bzw. den Dienst einer Überprüfungsstelle (*Validation Authority*, VA) in Anspruch zu nehmen.³² Die Überprüfung eines Zertifikats findet in folgenden Schritten statt:³³

- Überprüfung der Integrität des Zertifikats und der digitalen Signatur
- Sicherstellen, dass das Zertifikat von einer vertrauenswürdigen Zertifizierungsstelle ausgestellt wurde
- Überprüfen, ob die Gültigkeit des Zertifikats noch nicht abgelaufen ist
- Überprüfen, ob das Zertifikat in Übereinstimmung mit den Einschränkungen bezüglich seiner Anwendbarkeit (*Policy Restrictions*³⁴) eingesetzt wird

Eine PKI kann um weitere Elemente wie eine Stelle zur Vergabe von vertrauenswürdigen Zeitstempeln (*Timestamp Server*), Schlüssel Backup- (*Key Backup*) und Wiedergewinnungsserver (*Recovery Server*) und andere erweitert werden.³⁵

2.3.2 Dienste

Die Basisdienste einer PKI sind Authentisierung, Integrität und Vertraulichkeit. Die Authentisierung wird durch das Zusammenspiel von digitalen Signaturen und digitalen Zertifikaten ermöglicht. Diese werden vom Anwender an eine PKI-fähige Anwendung gesendet. Die Anwendung verifiziert die digitale Signatur mittels des öffentlichen Schlüssels, der ein Teil des Zertifikats ist, sowie das Zertifikat selbst.³⁶ Die Integrität ist mit der digitalen Unterschrift sichergestellt. Für die Sicherstellung der Vertraulichkeit ist pro Kommunikationsteilnehmer je ein Schlüsselpaar erforderlich.³⁷

Die Gewährleistung der drei Basisdienste Authentisierung, Integrität und Vertraulichkeit ermöglicht eine Implementierung weiterer Dienste, deren Grundlage die gesamten Basisdienste bilden. Sichere Kommunikation im Sinne von Kapitel 2.1 wird durch die Erweiterung der üblichen Netzwerkprotokolle um die Basisdienste einer PKI erreicht. Beispiele dafür sind:

- sicheres E-Mail (S/MIMEv2)³⁸
- sichere Webserver- Kommunikation mit SSL/TLS¹⁴
- virtuelle private Netzwerke (VPN) mit IPsec/IKE³⁹

³¹ Vgl. Adams / Lloyd, 1999, S. 161-165

³² Vgl. Adams / Lloyd, 1999, S. 55

³³ Vgl. Adams / Lloyd, 1999, S. 101-104

³⁴ Vgl. Kapitel 2.3.3

³⁵ Vgl. Adams/Lloyd, 1999, S. 33 - 40

³⁶ die einzelnen Schritte der Authentisierung mit Zertifikaten wurden bereits im Kapitel 2.3.1 geschildert.

³⁷ Vgl. Kapitel 2.2.2

³⁸ Spezifikation in Dusse u.a., 1998a, RFC 2311 und Dusse u.a., 1998b, RFC 2312

Auf den Basisdiensten aufsetzend kann eine PKI um einen Dienst zur Vergabe von vertrauenswürdigen Zeitstempeln erweitert werden. Mit einem vertrauenswürdigen Zeitstempel kann die Existenz eines Dokumentes zu einer bestimmten Zeit nachgewiesen werden. Solche Zeitstempel werden benötigt, um in Kombination mit einer digitalen Unterschrift die sogenannte Nicht-Abstreitbarkeit zu gewährleisten.⁴⁰ Damit kann nachgewiesen werden, dass ein Dokument von einem Anwender zu einer bestimmten Zeit abgeschickt wurde. Dieser Dienst ist eine wichtige Grundlage für die Abwicklung von elektronischen Transaktionen. Ein weiterer naheliegender PKI-Dienst ist die Verschlüsselung von Daten. Ein wichtiger Dienst innerhalb einer PKI ist die sichere Archivierung von abgelaufenen Zertifikaten, alten Zertifikat-Rückruflisten und weiteren Daten, die in der Zukunft als Nachweis digitaler Kommunikation oder Transaktion dienen könnten.

2.3.3 Sicherheitsniveau

Ein bekanntes Sprichwort besagt, dass eine Kette so stark (oder schwach) ist, wie ihr schwächstes Glied. Der kritischste Punkt einer PKI ist der private Schlüssel der Zertifizierungsstelle. Wenn der private Schlüssel der Zertifizierungsstelle in unberechtigte Hände gelangt, dann besteht für den Anwender nunmehr keine Gewissheit darüber, ob ein vermeintlich von dieser Zertifizierungsstelle signiertes Zertifikat tatsächlich von dieser Zertifizierungsstelle herausgegeben wurde. Das Vertrauen in die Infrastruktur wäre nicht mehr vorhanden. Weitere kritische Punkte einer PKI sind die privaten Schlüssel der Anwender. Sobald ein privater Schlüssel in die Hände eines Dritten gelangt, ist dieser in der Lage, sich während der digitalen Kommunikation für den Besitzer des privaten Schlüssel auszugeben und eventuell Handlungen in seinem Namen durchzuführen.

Abhängig vom Einsatzbereich der PKI sind unterschiedliche Maßnahmen notwendig, um das entsprechende Sicherheitsniveau zu gewährleisten. Eine PKI, die für das Militär betrieben wird, hat selbstverständlich höhere Sicherheitsanforderungen als eine firmenintern betriebene PKI. Die Sicherheitsanforderungen einer PKI werden durch eine sogenannte *Certificate Policy* (CP) beschrieben. Die Inhalte eines solchen Dokumentes werden nach RFC 2527 in acht Abschnitte eingeteilt:

1. Die Einleitung definiert unter anderem den Anwendungsbereich der PKI sowie ihre Anwendergruppen
2. Die Rahmenbedingungen beinhalten die Verpflichtungen der PKI-Teilnehmer sowie rechtliche und finanzielle Angelegenheiten
3. Identifizierung der Anwender bei der Registrierungsstelle und die spätere Authentisierung der Anwender gegenüber der Zertifizierungsstelle
4. Organisatorische Abläufe innerhalb der Zertifizierungsstelle, wie Ausstellung und Rückruf von Zertifikaten oder die Sicherheitsüberwachung
5. Nicht technische Sicherheitsmaßnahmen wie Lage und Zugang zu sicherheitskritischen Räumlichkeiten oder die fachlichen Anforderungen der betreibenden Personen
6. Technische Sicherheitsmaßnahmen wie Schutz der privaten Schlüssel oder die Netzwerksicherheit

³⁹ Spezifikation in Kent / Atkinson, 1998, RFC 2401 und Thayer / Doraswamy / Glenn, 1998, RFC 2411

⁴⁰ Vgl. Adams/Lloyd, 1999, S. 58

7. Die Inhalte der Zertifikate und der Zertifikatsrückruflisten, beispielsweise der Aufbau des Namens des Antragsstellers (*Subject*) bestimmter Anwendergruppen oder der Einsatz von Erweiterungsfeldern
8. Die Verwaltung der Richtlinien, wie Ablauf von Änderungen oder die Veröffentlichungs- und Benachrichtigungsrichtlinien

Eine Certificate Policy ist mittels einer eindeutigen Nummer (OID *Object Identifier*) identifiziert. Diese OID kann in einem speziell dafür vorgesehenem Feld eines X.509 Zertifikats hinterlegt werden. Damit wird festgelegt, dass ein Zertifikat ausschließlich nach Richtlinien dieser Certificate Policy eingesetzt werden darf.

Wie bereits erwähnt definiert eine CP lediglich die Sicherheitsanforderungen die an den Betrieb einer PKI gestellt werden, ohne die dafür notwendigen Praktiken zu nennen. Damit ist eine geeignete Grundlage geboten die Sicherheitsniveaus verschiedener PKIs zu vergleichen. Ein Beispiel dafür ist die gegenseitige Zertifizierung von Zertifizierungsstellen (*Cross Certification*).⁴¹ Die zertifizierende CA ist dabei verpflichtet alle betroffenen CPs der zertifizierten CA zu verifizieren. Des Weiteren kann eine CP einen gemeinsamen Sicherheitsstandard definieren, der von mehreren PKIs akzeptiert wird.⁴²

Das sogenannte *Certification Practice Statement* (CPS) beschreibt wie die Sicherheitsanforderungen in einer PKI im Einzelnen erreicht werden und setzt damit auf der Certificate Policy auf. Im Gegensatz zu CP bezieht sich CPS immer auf eine bestimmte PKI. CPS beschreibt beispielsweise die Vorgehensweisen einer Zertifizierungsstelle bei der Herausgabe, Veröffentlichung, Archivierung, Revokation und Verlängerung von Zertifikaten.⁴³

2.3.4 Rechtliche Grundlagen

Im folgenden werden relevante Gesetze aus Deutschland, Europa und den USA beschrieben und Problemfelder in der Gestaltung einer einheitlichen rechtlichen Grundlage aufgezeigt.

2.3.4.1 Signaturgesetz und Signaturverordnung

Das Signaturgesetz (SigG) wurde im Juli 1997 als Artikel 3 des Informations- und Kommunikationsdienstegesetzes („Multimediasgesetz“) verabschiedet. Es war damit das erste Gesetz zur Regelung der digitalen Unterschrift in Europa. Das SigG umfasst eher allgemein gehaltene Richtlinien einer Infrastruktur für digitale Signaturen auf einem hohen Sicherheitsniveau.⁴⁴

Da sich die Sicherheit von Algorithmen bzw. Schlüssellängen schnell ändern kann, enthält das SigG darüber keine detaillierteren Angaben. Festgelegt ist dagegen in §7, wie ein Zertifikat auszusehen hat. Die Vorgaben darüber entsprechen weitgehend dem X.509 Standard, der in Kapitel 2.2.3 vorgestellt wurde.

Genauere Vorgaben (u.a. Schlüssellänge und Gültigkeitsdauer von Zertifikaten) werden in der ergänzenden Signaturverordnung (SigV) und dem unverbindlichen

⁴¹ Vgl. Austin, 2001, S. 53

⁴² Vgl. Adams, Carlisle u.a. (1999), S. 86

⁴³ Vgl. Chokani / Ford, 1999, RFC 2527, Kapitel 3.4

⁴⁴ Vgl. Häcker, 1998, S. 58-60

Maßnahmenkatalog des Bundesamtes für Sicherheit in der Informationstechnik (BSI) gemacht, da sich diese schneller den neuesten Entwicklungen anpassen lassen. Kritisiert werden die (der behördlichen Kontrolle unterliegenden) hohen Anforderungen an die privaten Zertifizierungsstellen.⁴⁵

2.3.4.2 USA und Europa

In den USA, im Bundesstaat Utah, wurde am 1. Mai 1995 das weltweit erste Gesetz über digitale Signaturverfahren erlassen. Der ‚Utah’s Digital Signature Act‘ spricht digital unterzeichneten Dokumenten die gleiche Gültigkeit wie handschriftlich unterzeichneten Dokumenten zu.⁴⁶ Jedoch gibt es ähnliche Gesetze bis jetzt nur in einzelnen US-Bundesstaaten.

Anfang Dezember 1998 scheiterte eine Entscheidung über eine Signatur-Richtlinie der zuständigen EU-Minister in Brüssel. Deutschland hatte auf klaren Bestimmungen für die Sicherheit digitaler Signaturen bestanden. Die Kritiker - Großbritannien, die Niederlande, Finnland und Schweden - erhielten Rückendeckung durch den für Informationstechnik zuständigen EU-Kommissar Martin Bangemann. Er befürchtete, dass gesetzliche Regelungen den Online-Handel behindern und der schnellen technischen Entwicklung nicht folgen könnten.⁴⁵

2.3.4.3 Problemfelder

Generell besteht bei länderübergreifenden Techniken das Problem der uneinheitlichen Rechtsverhältnisse. So gilt das Internet zur Zeit noch als teilweise rechtsfolgenfreier Raum.⁴⁷ Die Klärung von Garantieansprüchen und Haftungsfragen bei internationalen Geschäftsvorgängen über das Internet wird wohl auch in nächster Zukunft ungeklärt bleiben. Für die Entwicklung des E-Commerce ist dies eher hinderlich: Nach einer Umfrage der KPMG sind 80% der befragten Unternehmen der Meinung, dass die unsichere Rechtssituation eine Hürde bei der Planung oder Durchführung von Ecommerce über das Internet darstellt.⁴⁸

Durch die schnelle Weiterentwicklung der Technik sind gesetzliche Vorgaben, die exakte Angaben zu den Algorithmen und Parametern machen, nicht sinnvoll.

Für den Kryptographie-Export von Verfahren mit ausreichender Sicherheit aus den USA galt bis vor kurzem das gleiche Gesetz, das auch den Waffenexport aus den USA regelt. Nicht nur die Kryptographie-Software, sondern auch die verschlüsselten Daten selbst wurden damit als „Kriegsmaterial“ subsumiert und der Export verboten.⁴⁹

⁴⁵ Vgl. Schulzki-Haddouti, 1999, S. 58

⁴⁶ Vgl. Häcker, 1998, S. 55-56

⁴⁷ Vgl. Häcker, 1998, S. 2

⁴⁸ Vgl. KPMG, 1998, S. 13-14

⁴⁹ Vgl. Häcker, 1998, S. 67-68

3. Überblick über ProAd

Bevor auf die sichere Webintegration von ProAd, als den zentralen Punkt der Diplomarbeit, eingegangen wird, erfolgt eine Beschreibung des Systems. Zu Beginn wird die technische Plattform und die Funktionalität der Standardsoftware für die Agentur-Branche erläutert. Anschließend wird die Notwendigkeit einer Webversion von ProAd begründet.

3.1 Technische Plattform

ProAd ist ein Client-Server System. Die Präsentationsschicht und die Anwendungsebene befinden sich auf dem Client. Das Datenbackend bildet ein relationaler Datenbankserver. Aufgrund von Kundenanforderungen in der Agenturbranche ist man gezwungen sowohl client-seitig als auch server-seitig mindestens die Plattformen MS-Windows und Macintosh zu unterstützen. Deshalb wurde als Entwicklungs- und Laufzeitplattform für den Client Omnis Studio⁵⁰ gewählt. Dabei handelt es sich um ein RAD-Werkzeug (*Rapid Application Development*), das auf den Plattformen MS-Windows, Macintosh, Linux und Sun Solaris lauffähig ist. Omnis Studio bietet leistungsfähige Funktionalität für die Erstellung der Präsentationsschicht, eine objektorientierte Programmiersprache sowie komfortable Zugriffsmöglichkeiten auf einen Datenbankserver.

Das Netzwerkprotokoll zwischen dem Client und dem Datenbank-Server ist mit der Auswahl eines bestimmten Datenbankservers festgelegt. Momentan werden Oracle⁵¹ und Frontbase⁵² unterstützt, wobei das System darauf ausgelegt ist, potentiell mit jedem Datenbankserver arbeiten zu können, für den das Omnis-Laufzeitsystem über ein Zugriffsmodul verfügt. Dies sind neben Oracle unter anderem DB2, Informix und Sybase.

3.2 Funktionalität

ProAd verfügt über eine mehrsprachige Präsentationsschicht. Die Auswahl der Sprache erfolgt bei der Anmeldung des Anwenders in das System. Die Authentisierung ist Benutzer/Passwort basierend. Die Anwenderdaten sind in der Datenbank unverschlüsselt gespeichert. Das Rechtesystem ist profilbasierend. Mehrere Profile können angelegt und mehrere Benutzer einem Profil zugeordnet werden, wobei ein Benutzer nur einem Profil zugeordnet werden kann. Rechte können auf Programmmenüs oder Programmdialoge vergeben werden. Pro Zugriffsobjekt können bis zu fünf Rechte vergeben werden:

1. kein Zugriff
2. Lesen
3. Änderungen durchführen
4. Datensätze anlegen
5. Daten löschen

ProAd unterstützt in seiner Funktionalität die Planung, Steuerung und Kontrolle von geschäftlichen Strukturen und Prozessen in Agenturen und werbetreibenden Unternehmen. Ein besonderes Merkmal von ProAd stellt die starke Anlehnung an

⁵⁰ Vgl. Omnis Web, 2002

⁵¹ Vgl. Oracle Web, 2002

⁵² Vgl. Frontbase Web, 2002

Strukturen und Prozesse dar, die in der Agentur-Branche üblich sind. Der zentrale Begriff ist dabei der Job. Ein Job kann ein eigenständiges Projekt oder ein in sich abgeschlossener Teil eines Projektes sein. Im engeren Sinne kann ProAd als ein System zur Planung, Steuerung und Kontrolle von Agentur-Jobs verstanden werden. In der Agentur-Branche wird eine solche Software oft als Job-Tracking-System bezeichnet.

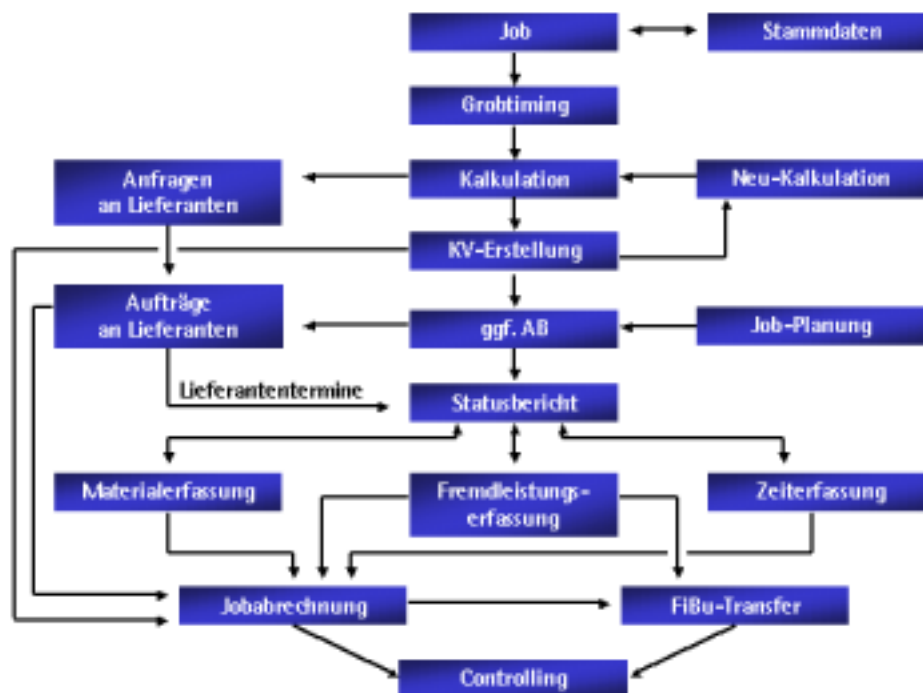


Abbildung 7: Flussdiagramm zum Ablauf eines Jobs

Für die Erfassung der Agenturstruktur verfügt ProAd über Konzepte wie Holding, Profit Center, Projekt und Kostenstelle. Den Kostenstellen können Leistungen zugeordnet werden, die von ihrer Art entweder einer eigenen Tätigkeit (Eigenleistung), einer erworbenen Tätigkeit (Fremdleistung) oder einem Materialverbrauch (Materialleistung) entsprechen. Bezüglich der Leistungsarten, die neben den Jobs ein zentrales Konzept von ProAd bilden, können interne Preise sowie Verkaufspreise definiert werden, welche später für die Abrechnung der Leistungen oder für die Projektkalkulation herangezogen werden können. Die Verkaufspreise können allgemeingültig (also nur leistungsartbezogen) oder kunden- bzw. projektbezogen definiert werden.

Planungsaktivitäten können budgetbasierend durchgeführt werden, wobei Budgets sowohl global als auch kundenbezogen definiert werden können. Budgets können in kleinere Einheiten (Etats) aufgegliedert werden. Für die Planung der wirtschaftlichen Durchführung von Jobs existiert ein Kalkulationsmodul. Die Planungseinheit stellt die Leistungsart dar.

Bei der täglichen Arbeit mit dem System können Kunden/Lieferanten erfasst sowie entsprechende Jobs kundenbezogen angelegt und bearbeitet werden. Die vollbrachten Leistungen können im Rahmen der Zeit- und Materialerfassung und innerhalb des integrierten Fakturierungsmoduls jobbezogen und leistungsartbezogen erfasst werden. Die erfassten Leistungen können bei der Erstellung von Ausgangsrechnungen

abgerechnet werden. Die Ausgangsrechnungen sowie weiterer Dokumente wie Bestellung oder Auftragsbestätigung werden durch das Modul „Dokumentenwesen“ erstellt. Das Modul verwendet als Komponente für die Erstellung der Dokumente Microsoft Word. Die Dokumente können dynamisch mit Daten aus ProAd gefüllt und auf vorhandenen oder eigens erstellten Vorlagen basierend kreiert werden.

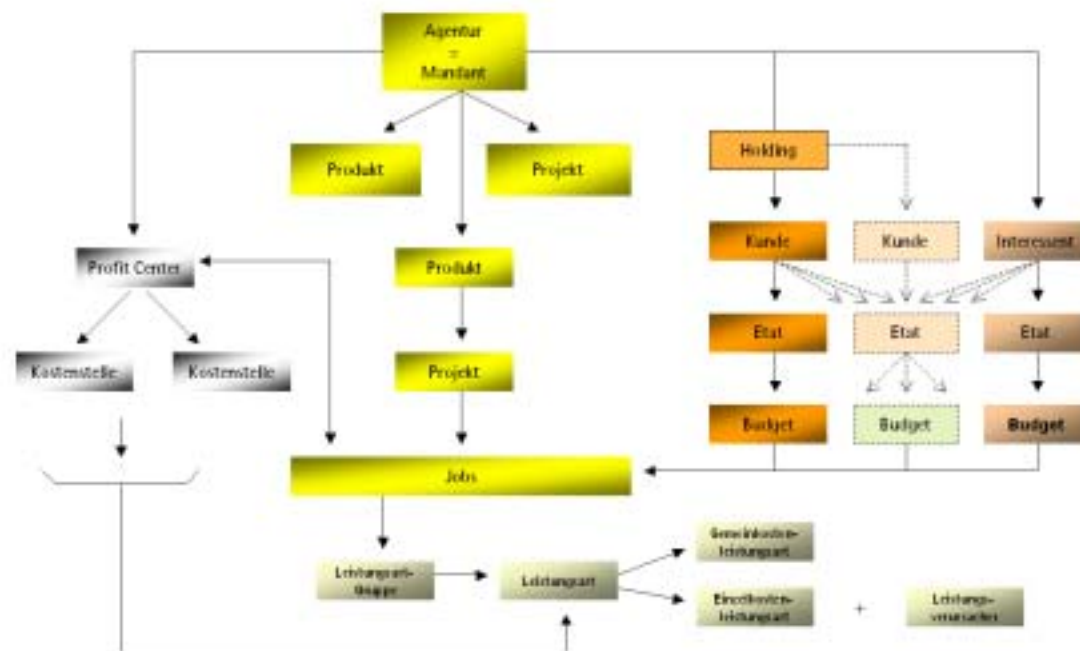


Abbildung 8: Das Hierarchiemodell von ProAd

Das Controllingmodul, das über eine Vielzahl von Auswertungen verfügt, wie Kundenrentabilität, Jobbewertung oder Nachkalkulation, ermöglicht den Überblick über das wirtschaftliche Geschehen im Unternehmen zu bewahren. ProAd verfügt über Schnittstellen zu mehreren Finanzbuchhaltungsprogrammen wie Datev⁵³, Sage KHK⁵⁴ oder Lexware⁵⁵.

3.3 Nächste Generation

ProAd Professional ist bereits die dritte Generation von ERP-Software für Agenturen und werbetreibende Unternehmen aus dem Hause J+D Software AG in über 10 Jahren der Firmengeschichte. Die Gesamtheit der an jede Programmgeneration gestellten funktionalen und technischen Anforderungen resultierte aus Kundenwünschen und eigenen Zielvorstellungen, welche bis zum Zeitpunkt der Entwicklung der Software gesammelt wurden. Beispiele für solche technische Anforderungen aus der Vergangenheit sind der Wechsel der Plattform vom DOS-Betriebssystem auf eine hybride Plattform (Windows und Macintosh), der Einsatz eines Datenbankmanagementsystems (DBMS) als Datenbackend oder der Einsatz von MS-Word als Dokumenteneditor. Die Kundenwünsche bezüglich der Funktionalität wurden situationsabhängig (technische und wirtschaftliche Determinanten) in der jeweils aktuellen Programmgeneration integriert oder gesammelt und bei der

⁵³ Vgl. Datev Web, 2002

⁵⁴ Vgl. Sage Web, 2002

⁵⁵ Vgl. Lexware Web, 2002

Spezifikation der folgenden Generation berücksichtigt. Dies führte letztendlich zu einem ausgereiften betriebswirtschaftlichen Konzept von ProAd mit einer ergonomischen Anwenderschnittstelle.

Die technische Plattform von ProAd basiert auf einem zweischichtigen Client-Server Konzept mit sogenannten ‚fetten Client‘. Damit ist die Integration der Präsentationsebene und Anwendungslogik auf dem Client gemeint, während der Server einzig und allein für die Verwaltung von gemeinsamen Daten verantwortlich ist. Diese Architektur resultiert aus dem Einsatz des Entwicklungs- und Laufzeitsystems Omnis, das seiner Zeit die einzige stabile und vermarktbare Plattform für Anwendungen war, die auf Windows und Macintosh gleichzeitig lauffähig sein sollten.

Die theoretische Alternative zum ‚fetten Client‘, der sogenannte ‚leichter Client‘ (*Thin Client*), besagt, dass nur die Präsentationsschicht auf dem Client implementiert wird und die Anwendungslogik und die Datenebene außerhalb. Diese Architektur weist Vorteile im Hinblick auf die Implementierung einer verteilten Software, wie ProAd, auf. Durch die Auslagerung der Anwendungslogik auf den Server, wird der Client-Rechner entlastet und eine geringere Zahl an Netzwerktransfers verursacht. Wenn für die Realisierung der Datenebene ein Datenbankserver eingesetzt wird, dann muss der Anwendungslogik eine eigene Schicht zugeordnet werden, womit eine dreischichtige Architektur entsteht. Die Vorteile einer mehrschichtigen Architektur sind höhere Robustheit, Skalierbarkeit, Flexibilität und weniger aufwändige Pflege der Anwendung. Das wohl bekannteste Beispiel einer mehrschichtigen Architektur ist das Internet, wo der Webbrowser Dokumente anfordert, die auf fernen Rechnern gespeichert sind, und diese lokal darstellt.⁵⁶

Das Internet stellt auf Grund seiner Plattformunabhängigkeit eine geeignete Alternative für eine Plattform für die nächste Generation von ProAd dar. Ein auf Internet-Technologien basierendes ProAd könnte im Inter-, Intra- und Extranet eingesetzt werden, womit bei Bedarf ein weltweiter Zugriff gewährleistet ist. Der Einsatz erprobter Open-Source Software wie Browser und Webserver als Programmkomponenten würde den Entwicklungsaufwand vermindern und Lizenzkosten für Systemsoftware, wie Omnis, vermeiden.

Bei der Entwicklung von ProAd als Webanwendung müssen sicherheitskritische Aspekte besonders beachtet werden. Bei einer typischen ERP-Anwendung haben noch nicht einmal alle Mitarbeiter das Recht auf den gesamten Datenbestand zuzugreifen. Eine webfähige ERP-Anwendung, bei deren Entwurf sicherheitskritische Aspekte⁵⁷ vernachlässigt wurden, könnte als ein offenes Scheunentor für beliebige Internetteilnehmer aufgefasst werden. Der dabei entstandene Schaden wäre wesentlich größer als der gebotene Nutzen durch den Einsatz von Webtechnologien.

Damit kann insgesamt geschlussfolgert werden, dass die Anforderung an die nächste Generation von ProAd darin besteht, eine auf Internet-Technologien basierende, nach den im Kapitel 2.1 genannten Richtlinien abgesicherte, mehrschichtige Plattform bereitzustellen und dabei die breite Funktionalität von ProAd Professional mindestens beizubehalten.

⁵⁶ Vgl. Edwards / Harkey / Orfali, 1997, S. 20-22

⁵⁷ Vgl. Kapitel 2.1

4. Sichere Webintegration von ProAd mit PKI

Dieses Kapitel dient als konzeptionelle Grundlage für die Erstellung des Prototyps der webfähigen Zeit- und Materialerfassung von ProAd. Zu Beginn werden Vorgaben und Anforderungen definiert, gefolgt von der entworfenen Architektur. Zum Schluß wird eine Überblick über Aspekte verschafft, die zu bedenken sind, um das gewünschte Sicherheitsniveau zu erreichen. Die Grundlage dafür stellt das Kapitel 2.3.3 dar, wo sicherheitskritische Aspekte einer PKI behandelt wurden.

4.1 Vorgaben und Anforderungen

ProAd ist eine kommerzielle Standardsoftware, die darauf ausgelegt ist viele potentielle Anforderungen eines Kunden zu erfüllen.⁵⁸ Diese Richtlinie soll auch für die Gestaltung der Architektur der sicheren Webintegration von ProAd gelten. Dies bedeutet, dass die Entstehung von kundenspezifischen Einzellösungen zu vermeiden ist. Die Architektur soll unabhängig sein von einzelnen Drittprodukten wie Webbrowser, Webserver oder PKI-Lösung und auf Standards basieren. Die Zielsetzung besteht darin, eine flexible und generische Lösung zu liefern, welche die technischen Einschränkungen der aktuellen Programmgeneration eliminiert und ihre Funktionalität beibehält.

Spezifisch betrachtet besteht die Anforderung darin, webfähige Präsentationsschicht und Anwendungsebene zu entwerfen, die

- in ihrer Funktionalität der ProAd Spezifikation entsprechen
- einen bestehenden ProAd-Datenbankserver als Datenbasis verwenden
- sicher, d.h. authentisiert, vertraulich und integer, miteinander kommunizieren können

Die ProAd-Datenbank soll als Komponente behandelt werden, d.h. bestehende Objekte (Tabellen, Views etc.) dürfen im Rahmen der Webintegration nicht geändert werden. Als DBMS sollen mindestens Oracle und Frontbase unterstützt werden. Die Absicherung der Kommunikation ist mittels PKI-Technologien, wie in Kapitel 2 beschrieben, vorzunehmen. Falls der Betrieb in einer gesicherten Umgebung stattfindet, dann soll auf die Verschlüsselung der Leitung verzichtet werden können, die in diesem Fall unnötigen Aufwand erfordert. Die Authentisierung soll zertifikatbasiert nach dem Standard X.509v3 ablaufen. Die Architektur soll die zur Vergabe von Zertifikaten erforderlichen Abläufe berücksichtigen. Zertifikate einer bereits bestehenden PKI sollen verwendet werden können.

Ein paralleles Arbeiten der neuentwickelten Webversion und der bestehenden Omnis Version von ProAd soll möglich sein. Im Einzelnen bedeutet dies, dass beide Versionen identische Datensätze auf der Datenbank erzeugen sowie identische Absicherungsstrategien bezüglich gleichzeitiger Zugriffe von mehreren Clients implementieren (Transaktionen, Locking).

4.2 Architektur der Webanwendung

Die Architektur ist dreischichtig: Präsentationsebene, Anwendungsebene und Datenbackend und wendet damit das Konzept aus Kapitel 3.3 an. Die

⁵⁸ Vgl. Kapitel 3

Präsentationsebene ist auf dem Client angesiedelt. Ein Webserver bildet die Zugriffsschnittstelle der Anwendungsebene für einen Client. Als Backend dient eine relationale Datenbank, die Daten für die Anwendungsebene liefert. Eine optionale Komponente ist der Trustcenter der angewandten PKI.⁵⁹

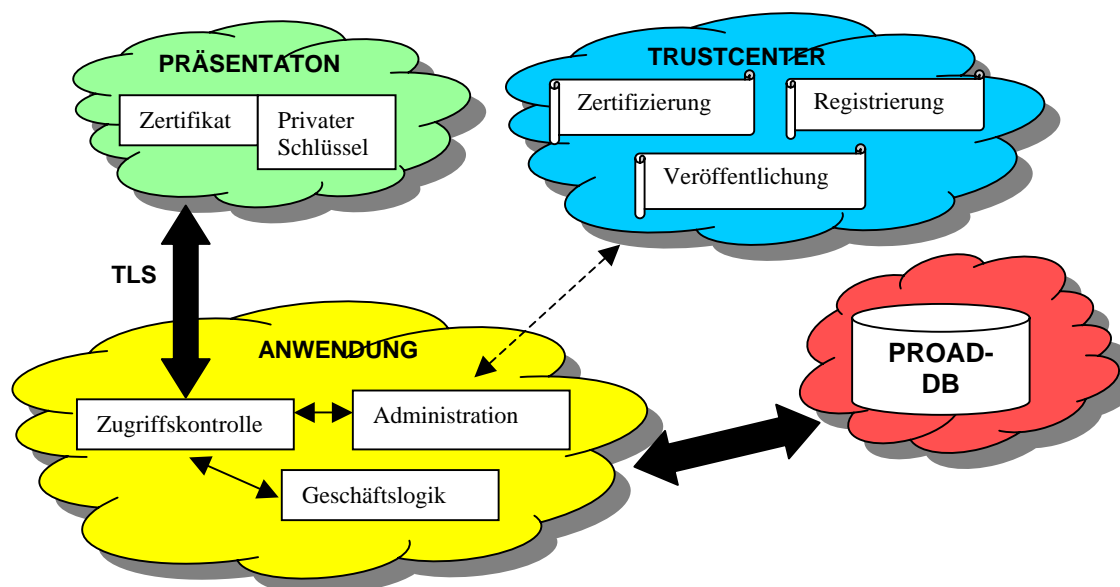


Abbildung 9: Architektur der Webanwendung im Überblick

4.2.1 Präsentationsebene

Die Präsentationsebene bildet die Anwenderschnittstelle, d.h. sie nimmt Anwendereingaben entgegen, löst bei Bedarf eine Aktion auf der Anwendungsebene aus und zeigt ihre Ergebnisse an. Die Präsentationsebene ist auf dem Client angesiedelt und soll gemäß den vorhandenen ProAd Spezifikationen gestaltet werden. Der relevante Teil der Spezifikationen betrifft das Design und die Ergonomie der Anwenderschnittstelle.

Die typische Client-Software einer Webanwendung ist der Webbrowser. Eine Alternative könnte eine eigens implementierte Anwenderschnittstelle sein. Ausgereifte Netzwerkfunktionalität und insbesondere die Unterstützung von Zertifikaten gehören zur Standardausstattung moderner Webbrowser und müssten für einen eigens entwickelten Client erst implementiert werden. Dies ist von besonderer Bedeutung, da der Client in der Lage sein muss ein Anwenderzertifikat samt privaten Schlüssel sicher zu verwalten und einzusetzen. Ein weiterer Vorteil des Webrowsers ist die allgemeine Verfügbarkeit und damit ein geringer Aufwand für die Installation. Der Nachteil eines Webbrowser gegenüber einer eigens entwickelten Client-Software ist die vermeintlich weniger leistungsfähige Anwenderschnittstelle, was das Design und die Ergonomie betrifft. Hier ist der Entwickler auf die Funktionalität von HTML beschränkt. Dieser Nachteil kann allerdings durch den Einsatz von Erweiterungen, wie Javascript, wettgemacht werden. Dabei sollten jedoch eventuelle

⁵⁹ Die Begründung folgt im Kapitel 4.2.2.3

Sicherheitslöcher und/oder Einschränkungen der Plattformenunabhängigkeit von HTML bedacht werden.

4.2.2 Anwendungsebene

Ein Webserver ist die Zugriffsschnittstelle des Clients auf die Anwendungsebene. Die Anwendungsebene besteht aus der Geschäftslogik, Zugriffskontrolle und Administration.

4.2.2.1 Geschäftslogik

Die Geschäftslogik versorgt die Präsentationsebene mit Anwendungsdaten und führt die dort initiierten Aktionen aus. Dabei werden die Anwendungsdaten vom Datenbackend geliefert.

Die bereits bestehende Geschäftslogik kann leider in keiner Weise wieder verwendet werden, da der vorhandener Code aus nativen Omnis-Modulen besteht.⁶⁰ Die neu implementierte Geschäftslogik hat, genauso wie die Anwenderschnittstelle, die bestehenden ProAd Spezifikationen einzuhalten. Der für die Geschäftslogik relevante Teil der Spezifikationen betrifft die durchzuführenden Aktionen sowie ihre Autorisierung.

4.2.2.2 Zugriffskontrolle

Die Zugriffskontrolle ist für die Authentisierung von Client-Sitzungen⁶¹ (*Client-Session*) verantwortlich. Eine einmal authentifizierte Sitzung behält ihre Identität, bis eine explizite Neuansmeldung durch den Client stattfindet. Die Lebensdauer einer ungenutzten Sitzung wird auf 30 Minuten festgelegt. Zwecks Authentisierung muss der Client sein Zertifikat und seine digitale Signatur an die Anwendungsebene übermitteln.

Die Authentisierung erfolgt in drei Schritten:

1. Digitale Signatur prüfen
2. Zertifikate prüfen (Client-Zertifikat, CA-Zertifikate, Policy)
3. Identifizierung des zum Zertifikat gehörenden Proad-Anwender

Diverse Webserver (beispielsweise Apache⁶²) integrieren Mechanismen für die Prüfung der digitalen Signatur und der Zertifikate. Wenn der verwendete Webserver solche Funktionalität nicht anbietet, dann muss die Software eines Drittanbieters eingesetzt bzw. eine Eigenimplementierung durchgeführt werden. Die Vorgehensweise bei der Durchführung der ersten beiden Schritte ist standardisiert und wurde bereits in den Kapiteln 2.2 und 2.3 ausführlich behandelt. Die Identifizierung des zum Zertifikat gehörenden Proad-Anwenders verlangt die Entwicklung einer eigenen Vorgehensweise.

Eine naheliegende Möglichkeit dafür ist, einen eindeutigen Verweis auf das Zertifikat mit den übrigen Benutzereigenschaften in der Datenbank zu speichern. Dafür kommen beispielsweise der Antragsteller, der Antragsteller und der Herausgeber oder

⁶⁰ Vgl. Kapitel 3.1

⁶¹ Da HTTP ein zustandsloses Protokoll ist, muss eine solche Sitzung künstlich verwaltet werden. Vgl. Crawford / Hunter, 2001

⁶² Vgl. Apache Web, 2002

die Zertifikat-ID in Frage. Dies erfordert die Existenz entsprechend dafür vorgesehenen Tabellenfelder in der Datenbank. Anderenfalls müssten neue Felder hinzugefügt werden, was eine Verletzung der Anforderung bedeuten würde, die jegliche Änderungen der Datenbank verbietet. Leider sieht die ProAd-Datenbank entsprechende Felder nicht vor, womit das Konzept nicht weiter berücksichtigt werden kann.

Eine zweite Vorgehensweise zur Identifizierung des zum Zertifikat gehörenden ProAd-Anwender könnte darin bestehen, im Zertifikat einen eindeutigen Verweis auf den dazugehörenden ProAd-Anwender zu speichern. Beispielsweise könnte der eindeutige Anwendername als der *Common Name* (CN) des Antragsteller, die geschäftliche E-Mail-Adresse im entsprechenden Zertifikatfeld, der primäre Schlüssel oder ein Schlüsselkandidat in einem Erweiterungsfeld gespeichert werden. Eine solche Verknüpfung des ProAd-Anwenders mit seinem Zertifikat stellt ein elegantes Konzept dar, da weder die Datenbank noch eine weitere Komponente benötigt werden. Bei Anwendung dieser Vorgehensweise ist jedoch zu bedenken, dass ein Zertifikat mit allen seinen Informationen veröffentlicht ist, was für den Anwendernamen nicht zutrifft. Da die Erweiterungsfelder eines Zertifikats nicht von allen Zertifizierungsstellen gleichermaßen unterstützt werden, ist es nicht möglich mit der Anwendung der Datensatzschlüssel aus der Datenbank ein generisches Konzept aufzubauen. Würde die E-Mail-Adresse eines Anwenders als eindeutiger Verweis verwendet, so müsste bei jeder Änderung der E-Mail-Adresse ein neues Zertifikat erstellt werden. Wahrscheinlich könnte eine weitere, hier nicht behandelte Möglichkeit, gefunden werden, um eindeutige Anwenderinformationen im Zertifikat zu speichern. Spätestens an dieser Stelle sollte die Anforderung über die Verwendung bereits vorhandener Zertifikate betrachtet werden. Diese Anforderung würde nur in Ausnahmefällen zu erfüllen sein und somit zu ungewollten Einzellösungen führen.

Eine Folge der geschilderten Überlegungen ist, dass ein generisches Konzept zur Verbindung des Antragsstellers eines Zertifikates und seiner Identität als ProAd-Anwender nur über die Verknüpfung der entsprechenden Informationen außerhalb des Zertifikates und der ProAd-Datenbank aufgebaut werden kann. Möglichkeiten hierfür bieten, unter anderen, ein LDAP-Verzeichnis, eine weitere Anwenderdatenbank oder eine Konfigurationsdatei. Ein Eintrag in einem LDAP-Verzeichnis ist dafür besonders geeignet, da die Struktur des Namens eines Antragsstellers auf der LDAP-Baumstruktur basiert.⁶³ Des Weiteren ist LDAP ein international anerkannter Standard und eine breit eingesetzte Technologie, beispielsweise für Veröffentlichung von Zertifikaten innerhalb einer PKI.⁶⁴ Die betroffenen Einträge müssten lediglich um Attribute erweitert werden, welche auf die Anwenderdaten in der ProAd-Datenbank verweisen.

Eine generische Unterstützung von bestehenden Anwenderdatenbanken ist insofern problematisch, da die Tabellenstruktur von Datenbank zu Datenbank unterschiedlich sein wird. Für den Einsatz einer Konfigurationsdatei spricht die Einfachheit und Flexibilität des Konzepts. Die Installation von weiteren Servern (LDAP, Datenbank) bleibt erspart, da die Konfigurationsdatei innerhalb des lokalen Dateisystems der Anwendungsebene gespeichert werden kann. Für die Entwicklung einer solchen Konfigurationsdatei könnte beispielsweise XML (*Extensible Markup Language*) eingesetzt werden, das auf diesem Gebiet einen immer breiteren Einsatz findet.⁶⁵

⁶³ Vgl. Hardcastle-Kille, 1993, RFC 1485

⁶⁴ Vgl. Kapitel 2.3.1

⁶⁵ Vgl. Phillips, 2000, S. 8

4.2.2.3 Administration

Die Aufgabe der Administration ist die Verwaltung der Verknüpfungen von ProAd-Anwendern und ihren Zertifikaten. Für eine solche Verwaltung werden folgende Operationen benötigt:

- Verknüpfung eines Zertifikats mit einem Anwender anlegen
- Verknüpfung eines Zertifikats mit einem Anwender aufheben
- Anwenderinformationen zu einem Zertifikat anzeigen
- Gespeicherte Zertifikatinformationen zu einem Anwender anzeigen

Diese Operationen definieren eine Schnittstelle für den Zugriff auf die Verknüpfungsdaten. Diese Schnittstelle wird während der Authentisierung von der Zugriffskontrolle benutzt, die dadurch von der eingesetzten Implementierung der Verknüpfung (LDAP, Konfigurationsdatei) unabhängig ist. Optimalerweise implementiert die Administration beide Verknüpfungsarten. Durch Konfigurationseinstellungen kann die entsprechende Verknüpfungsart aktiviert werden.

Für den Zugriff auf die Administration sollten anwenderbezogen Rechte vergeben werden können. Da die Funktionalität der Verknüpfung der Anwender und ihrer Zertifikate in der Spezifikation von ProAd nicht vorgesehen wurde, existiert auch kein entsprechendes Recht. Eine naheliegende Möglichkeit ein geeignetes Recht einzusetzen besteht darin, für den Zugriff auf die Administration der Webversion die entsprechenden Zugriffsrechte (Lesen, Ändern usw.) auf die Administration der Omnis-Version zu verwenden. Alternativ könnte ein Verfahren eingesetzt werden, nach dem jeder Anwender vollen Zugriff auf seine persönlichen Verknüpfungsdaten erhält. Zusätzlich wird die Rolle des Administrators vorgesehen, womit alle Daten manipuliert werden dürfen.

Eine weitere Aufgabe der Administration könnte die automatisierte Kommunikation mit dem eingesetzten Trustcenter sein. Dazu gehören unter anderen Verfahren für die Beantragung, Verlängerung und den Rückruf von Zertifikaten von einem oder mehreren Anwendern. Besonders die gestapelte Verarbeitung von Anträgen verspricht große Zeitersparnis bei der Administration einer PKI.

Eine Voraussetzung für die Implementierung einer automatisierten Kommunikation ist die Existenz einer Schnittstelle, über die ein Trustcenter Nachrichten von anderen Programmen empfangen kann. Ein generisches, also von einzelnen Produkten unabhängiges, Konzept erfordert den Einsatz von Standards bei der Spezifikation einer solchen Schnittstelle. Beispiele für solche Standards sind RFC 2510, das die Kommunikation innerhalb einer PKI spezifiziert und das RFC 2511, das ein Standard für Anträge an ein Trustcenter definiert. Leider werden diese und andere Standards in vorhandenen Produkten bisweilen uneinheitlich unterstützt, was den Entwurf eines generischen Konzepts unmöglich macht.⁶⁶ Falls auf die automatische Kommunikation mit dem Trustcenter nicht verzichtet werden soll, muss also die Entscheidung für ein Produkt oder eine Produktgruppe gefällt werden.

Eine weitere grundsätzliche Anforderung, ist die Erreichbarkeit des Trustcenter durch die Webanwendung. Die Erreichbarkeit des Trustcenter ist bei Eigenbetrieb oder, im

⁶⁶ Vgl. Adams / Lloyd, 1999, S. 227-235

Falle des Fremdbetriebs, bei geeigneter Verfügbarkeit des Anbieters gewährleistet. Die Überlegungen führen zum Schluss, dass die Möglichkeit des Einsatzes der automatisierten Kommunikation mit dem Trustcenter von den spezifischen Anforderungen einer einzelnen Installation abhängig ist, was die oben getroffene Spezifikation des Trustcenters als optionale Komponente gerechtfertigt.

Zum Kapitelende wird auf die Frage der Betriebsform eines Trustcenters eingegangen. Der Eigenbetrieb eines Trustcenters kann nicht zwingend vorgeschrieben werden, weil die Entscheidung im Einzelfall von verschiedenen wirtschaftlichen, organisatorischen und technischen Begebenheiten des Kunden abhängen wird. Für den Eigenbetrieb spricht die hohe Flexibilität bei der Gestaltung sowie die hohe Kontrolle bei Betrieb der PKI. Selbstverständlich entstehen dabei keine Kosten für die Erstellung von Zertifikaten. Andererseits verursacht der Eigenbetrieb eines Trustcenters vorher nicht da gewesene Betriebskosten (Hardware, Lizenzkosten für die Software, Personal) und bedarf den Aufbau von technischem Spezialwissen. Es ist anzunehmen, dass die Betriebskosten viel weniger von der Größe der PKI abhängen, als die Kosten für Zertifikate bei der Inanspruchnahme einer entsprechenden Dienstleistung, welche linear ansteigen. Folglich werden sich größere Unternehmen eher für den Eigenbetrieb entscheiden, auch weil die dabei gewöhnlich anfallenden Investitionskosten oft auf Grund von Synergieeffekten mit vorhandener IT-Ausstattung gesenkt werden können. Kleinere Unternehmen werden sich dagegen wahrscheinlich eher, durch den hohen Investitions- und Know-How-Bedarf abgeschreckt, für den Fremdbetrieb entscheiden. Die jeweilige Entscheidung hängt im Einzelfall von der spezifischen Kostenlage und dem Vorhandensein von benötigtem Spezialwissen ab.⁶⁷

4.2.2.4 Sichere Verbindung

Verläuft die Kommunikation zwischen der Präsentationsschicht und der Anwendungsebene durch ein unsicheres Netz (z.B. Internet), so ist eine Absicherung erforderlich. Eine erprobte Technologie für die Gewährleistung eines vertraulichen und integren Datenaustausches in unsicheren Netzen basiert auf der SSL-Protokollfamilie (*Secure Socket Layer*). SSL wurde von der Firma Netscape⁶⁸ entwickelt und inzwischen als Internet-Standard unter dem Namen *Transport Layer Security* (TLS) als RFC 2246 anerkannt. SSL verwendet Schlüssel gespeichert in eingesetzten Zertifikaten, um die Verbindung zu verschlüsseln. Für die Herkunft der verwendeten Zertifikate gibt es zwei Möglichkeiten:

- Verwendung eines einzigen Server-Zertifikats, wobei der Client das Server-Zertifikat empfängt und es annehmen oder ablehnen kann (Serverauthentisierung)
- Erweiterung der Serverauthentisierung um die Clientauthentisierung, wobei beim Verbindungsaufbau die Zertifikate gegenseitig getauscht werden

Das zu verwendende Verfahren bestimmt der Server. In der hier vorgestellten Architektur wird selbstverständlich das Verfahren der Clientauthentisierung eingesetzt. Falls keine Sicherung der Leitung zwischen der Präsentations- und Anwendungsebene erforderlich ist, so wird SSL nur für die Übermittlung der Client-Zertifikate eingesetzt.

⁶⁷ Vgl. Adams / Lloyd, 1999, 255-256

⁶⁸ Vgl. Netscape Web, 2002

4.3 Sicherheitsniveau der eingesetzten PKI

Wie bereits im Kapitel 2.3.3 beschrieben sind die Sicherheitsanforderungen an eine PKI von ihrem Anwendungsbereich abhängig und durch eine Certificate Policy dokumentiert. Da die Angabe einer vollständigen Empfehlung einer CP nach RFC 2527 den Rahmen dieser Ausarbeitung sprengen würde, wird hier eine Checkliste mit den relevanten Punkten vorgestellt. Diese Checkliste kann sowohl für die Erstellung einer CP, als auch für die Bewertung der CP eines PKI-Anbieters verwendet werden.

Berechtigung und Anwendungsbereich

Die Zertifizierungsstelle (*Certificate Authority*, CA) dient dem Aufbau einer PKI für sicheren Zugriff auf eine webfähige Version von ProAd. Die Aufgabe der CA ist die Ausstellung von Zertifikaten, sowie die Gewährleistung weiterer Lebenszyklusfunktionen eines Zertifikats wie Verlängerung, Rückruf und Suspendierung. Jeder ProAd-Anwender ist prinzipiell berechtigt ein Zertifikat zu erhalten.⁶⁹ Da jeder ProAd-Anwender dem eigenen Unternehmen bereits bekannt ist, ist der Einsatz einer Registrierungsstelle nicht erforderlich. Die ausgestellten Zertifikate sind für Verschlüsselung und Signaturen einsetzbar und sollen für eine authentische, integre und vertrauliche Kommunikation im Internet sorgen. Der Einsatz der Zertifikate für rechtsgültige Unterschriften und finanzielle Geschäfte ist nicht erforderlich. Für die Installation des Zertifikats innerhalb von ProAd kann ein Administrator oder der Anwender selbst verantwortlich sein.

Rahmenbedingungen

Alle beteiligten Parteien - also die CA mit ihrem Veröffentlichungsverzeichnis, die Zertifikatnehmer (ProAd-Anwender) und die Anwendungen (ProAd Server) – verpflichten sich, die in der CP festgelegten Regeln einzuhalten. Die CA stellt die Zertifikate inklusive aller passender Schlüssel her und verpflichtet sich für einen verantwortungsvollen Umgang mit allen Schlüsseln. Des Weiteren veröffentlicht die CA das eigene Zertifikat, alle herausgegebenen Zertifikate, die aktuelle CRL (*Certificate Revocation List*) und die CP. Pro Anwender wird nur ein Zertifikat ausgestellt. Die Anwender sind verpflichtet ihre privaten Schlüssel vertraulich aufzubewahren. Die Anwendungen sind verpflichtet die Zertifikate gewissenhaft zu Prüfen.

Ablauforganisation

Falls die Administration der webfähigen Version von ProAd über die Funktionalität der automatischen Kommunikation mit dem Trustcenter verfügt, so ist diese zu verwenden um ein Zertifikat zu beantragen, es zu verlängern oder zu widerrufen. Anderenfalls muss der eingesetzte Trustcenter entsprechende Funktionalität anbieten. Die Herausgabe des Zertifikats und des privaten Schlüssels als PKCS#12-Softtoken wird innerhalb des erwarteten Einsatzbereichs als ausreichend bewertet. Der Versand des Softtoken kann per E-Mail erfolgen. Das Softtoken ist durch ein Passwort zu schützen, das dem Anwender auf einem dritten Weg zugeschickt wird (z.B. PIN-Brief). Falls eine höhere Sicherheitsstufe erwünscht ist, kann der Einsatz eines Hardoken – also einer Chipkarte – für den Transport und die Aufbewahrung des

⁶⁹ Es ist denkbar hierfür ein spezielles Recht einzuführen

privaten Schlüssels erfordert werden. Allerdings weist diese wesentlich sicherere Vorgehensweise zwei Nachteile auf:⁷⁰

1. Pro Client ist ein entsprechendes Lesegerät erforderlich, damit Chipkarten eingesetzt werden können, was eine weitere Investition bedeutet.
2. Um absolute Sicherheit zu gewährleisten, darf der private Schlüssel die Chipkarte niemals verlassen. Damit müssen alle Berechnungen, für die der private Schlüssel erforderlich ist, auf der Chipkarte ausgeführt werden, was die Laufzeit erheblich erhöht.

Die eingesetzte CA muss über Funktionalität zur Sicherheitsüberwachung verfügen und entsprechende Maßnahmen im Falle von Kompromittierung definieren.

Sicherheitsmaßnahmen

Die heutzutage als sicher angesehene Schlüssellänge beträgt 1024 Bit.⁷¹ Der private Schlüssel der CA, als das empfindlichste Glied der Kette, sollte eine größere Länge haben. Zu weiteren technischen Sicherheitsmaßnahmen zählt der Schutz des privaten Schlüssels bei der Erzeugung, Installation und Betrieb sowie die Netzwerksicherheit. Die eingesetzte CA muss auch in nicht technischen Bereichen abgesichert sein, d.h. in der Infrastruktur, organisatorisch und durch personelle Maßnahmen.⁷² Diese nicht technischen Sicherheitsmaßnahmen wie die Lage und der Zugang zu sicherheitskritischen Räumlichkeiten, eingeschränkte Anwendergruppen oder hohe fachliche Anforderungen an die betreibende Personen bilden eine wichtige Stütze im Gesamtkonzept einer PKI.

Profile für Zertifikate und Revokationslisten

Die ausgestellten Zertifikate sollen, wie bereits erwähnt, dem Standard X.509v3 entsprechen. Für die veröffentlichten CRLs gilt der Standard X.509v2. Als Kryptographie-Algorithmus wird RSA und als Hash-Algorithmus wird SHA-1 empfohlen. Diese Kombination gilt als sicher und findet breite Anwendung.⁷³

Folgende Zertifikatsprofile werden maximal erforderlich sein:

- Zertifikat der Zertifizierungsstelle, ist nur bei Eigenbetrieb der Zertifizierungsstelle erforderlich
- SSL-Server-Zertifikat
- Anwender Zertifikat

Neben den üblichen Inhalten sind für jedes Zertifikatsprofil jeweils die erforderlichen bzw. gewünschten Erweiterungen festzulegen und ist ein Format für den eindeutigen Namen des Antragsstellers zu definieren.⁷⁴

⁷⁰ Vgl. Adams / Lloyd, 1999, S. 177

⁷¹ Vgl. Kapitel 2.2.2

⁷² Vgl. Chokani / Ford, 1999, RFC 2527, Kapitel 4.5

⁷³ Vgl. Kapitel 2.2.2 und Kapitel 2.2.4

⁷⁴ Vgl. Housley / Ford / Polk / Solo, 1999, RFC 2459

5. Entwicklung eines Prototyps

Der praktische Teil der Diplomarbeit besteht aus der Implementierung eines Prototyps nach den konzeptionellen Überlegungen aus dem Kapitel 4. Der zu implementierende Teil der Geschäftslogik ist, wie bereits erwähnt, die Zeit- und Materialerfassung. Der Prototyp der webfähigen Zeit- und Materialerfassung von ProAd wird im folgenden mit dem Namen ProAdWeb bezeichnet.

Der erste Abschnitt verschafft einen Überblick über die umgesetzten Konzepte und eingesetzten Technologien. Die darauf folgenden Abschnitte beschreiben den Aufbau und Ablauf der Initialisierung, Datenbankankündigung, Authentisierung und Autorisierung von ProAdWeb.

Es wird darauf hingewiesen, dass der Package-Name aller in diesem Kapitel beschriebenen Java-Klassen und Java-Schnittstellen von ProAdWeb mit dem Präfix `com.jdsoftware.proadweb` beginnt.

5.1 Umgesetzte Konzepte und eingesetzte Technologien

Die zeitlichen und personellen Ressourcen während einer Diplomarbeit sind begrenzt. Gleichzeitig ist die Aufstellung von zeitlich exakten Plänen für komplexe Software-Entwicklungsprojekte stets problematisch. Daher sollte beim Entwurf der Architektur für ProAdWeb das Risiko der Nicht-Fertigstellung zum Abgabetermin berücksichtigt werden. Dieses Risiko kann, durch die Maßnahme, nur die wichtigsten, d.h. für die Lauffähigkeit von ProAdWeb notwendigen, Module zu implementieren, minimiert werden. Dies sind:

- Die Anwenderschnittstelle
- Das Zugriffsmodul
- Die Geschäftslogik incl. Autorisierung
- Die Datenbankschnittstelle

Auf die Administration wurde aus Zeitgründen vollständig verzichtet, woraus die Notwendigkeit der manuellen Konfiguration der Verknüpfung von Zertifikaten und ProAd-Anwendern resultiert.⁷⁵

Da Omnis Studio, die aktuelle Plattform von ProAd, nicht über die erforderliche Funktionalität verfügt, um die im Kapitel 4.2 entworfene Architektur entsprechend den im Kapitel 4.1 definierten Anforderungen zu implementieren⁷⁶, müssen dafür andere Technologien eingesetzt werden. Die Anwenderschnittstelle besteht aus dynamischen HTML-Seiten, die in einem Webbrowser dargestellt und in der Anwendungsebene erzeugt werden. Die Anwendungsebene besteht aus folgenden Komponenten:

- Apache Webserver (Version 1.3.24 mit `mod_ssl`⁷⁷ und `mod_jk`⁷⁸)
- Tomcat (Version 4.0.3)⁷⁹
- eigentlicher Anwendung, die auf Java-Technologien basiert

⁷⁵ Einzelheiten diesbezüglich vgl. Kapitel 5.2

⁷⁶ Omnis Webclient Web, 2002

⁷⁷ `mod_ssl` ist eine Erweiterung für den Apache Webserver um SSL-Funktionalität, vgl. `Mod_ssl` Web, 2002

⁷⁸ `mod_jk` ist eine Erweiterung für den Apache Webserver um das AJP Protokoll, das u.a. für die Kommunikation mit Tomcat verwendet werden kann, vgl. Tomcat Web, 2002

⁷⁹ Vgl. Jakarta Tomcat Web, 2002

Diese Architektur ist auf allen gängigen Betriebssystemen (Unix, Macintosh, Windows) lauffähig, da entsprechende Versionen von Apache und Tomcat erhältlich sind und Java bekanntlich plattformunabhängig ist.

Der Apache Webserver ist für die Kommunikation mit dem Client, Verschlüsselung der Leitung mittels SSL (falls erforderlich) und Überprüfung der Zertifikate verantwortlich. Alle ProAdWeb-Anfragen werden von Apache an Tomcat mittels AJP 1.3 (*Apache Jserv Protocol*) weitergeleitet, der als Laufzeitumgebung (Servlet-Container) für ProAdWeb fungiert. ProAdWeb ist eine serverseitige Java-Anwendung (JDK 1.4), die mittels der Servlet-Technologie⁸⁰ mit Tomcat kommuniziert. Die Erstellung von dynamischen HTML-Seiten für die Anwenderschnittstelle erfolgt mittels Velocity, das, ähnlich wie JSP⁸¹ (*Java Server Pages*), HTML-Vorlagen verwendet, die mit eigener Skriptsprache ergänzt werden können.⁸² Die Kommunikation mit der Datenbank erfolgt mittels JDBC 1.2⁸³ (*Java Database Connectivity*). Für die Authentisierung und Autorisierung wird JAAS⁸⁴ (*Java Authentication and Authorisation Service*) eingesetzt, das ins JDK 1.4 integriert ist. Dabei wurden die Ergebnisse eines neulich im Fachgebiet abgeschlossenen Praktikums als Basis verwendet. Multisprachfähigkeit von ProAdWeb wird mittels bekannten Java-Mechanismen für Internationalisierung sichergestellt.⁸⁵ Die Entwicklungsplattform ist Windows 2000. Als Entwicklungswerkzeug wird eine aktuelle Version von Eclipse⁸⁶ eingesetzt, das u.a. über einen Tomcat-Debugger verfügt.

Der Einsatz von Apache ist durch die bereits vorhandene Funktionalität zur Überprüfung der Client-Zertifikate und durch die hohe Performanz begründet. Java ist eine plattformunabhängige Technologie mit breiter Funktionspalette, insbesondere für Internet-Anwendungen. Tomcat ist die Referenzimplementierung der Java Servlet Spezifikation (Tomcat 4.0.x → Servlet 2.3, Tomcat 3.3.x → Servlet 2.2) und ohne Lizenzkosten erhältlich. Sowohl JDBC, als auch JAAS sind entsprechende Java-Standards für den Zugriff auf Datenbanken bzw. für die Authentisierung und Autorisierung.

Im Folgenden wird der Verzicht auf JSP zu Gunsten von Velocity begründet. Die Implementierung der Authentisierung verlangt eine zentrale „Eintrittsstelle“ für alle Clients, an der ihre Identität festgestellt werden kann. ProAdWeb verfügt aus diesem Grund über ein zentrales Servlet, an das alle Client-Anfragen umgeleitet werden.⁸⁷ Damit die Umleitung entsprechend funktioniert sind diverse Abbildungsregeln („mapping“) erforderlich, die im ‚Deployment Descriptor‘ – der Konfigurationsdatei der Webanwendung (web.xml) – codiert werden. Leider gelten diese Abbildungsregeln auch für die Weitergabe der Anfrage (mittels `RequestDispatcher.forward()` oder `sendRedirect()`) an eine JSP, womit erneut

⁸⁰ Vgl. Java Servlet Web, 2002

⁸¹ Vgl. JSP Web, 2002

⁸² Vgl. Jakarta Velocity Web, 2002

⁸³ Vgl. JDBC Web, 2002

⁸⁴ Vgl. JAAS Web, 2002

⁸⁵ Vgl. Crawford / Hunter, 2001, S. 391-421

⁸⁶ Vgl. Eclipse Web, 2002

⁸⁷ Vgl. Kapitel 5.2

das zentrale Servlet angesprochen und ein Zirkelbezug entstehen würde. Die Auflösung dieses Zirkelbezugs würde weitere komplexe Konzepte und Abbildungsregeln erfordern. Ein weiterer Nachteil der Verwendung von JSP resultiert daraus, dass der Kontext innerhalb dessen der Client authentisiert wird, bei einer Weitergabe der Anfrage an eine JSP seine Gültigkeit verliert und damit auch die erfolgte Authentisierung. Dies hängt mit dem Verlust des aktuellen Thread-Stacks, bei der Weiterleitung der Anfrage, zusammen.

Velocity kennt solche Schwierigkeiten nicht, da für die Erstellung der HTML-Seiten die Kontrolle nicht an den Servlet-Container abgegeben wird, sondern im aktuellen Thread verbleibt. Praktisch bedeutet dies die Weiterleitung der Anfrage durch einen simplen Java Methodenaufruf. Des Weiteren verfügt Velocity über komfortable und leistungsfähige Konzepte in der Servlet-Einbindung sowie Erstellung und Verwendung von HTML-Vorlagen.

ProAdWeb unterstützt die aktuellen Versionen der Datenbankserver Frontbase und Oracle mittels entsprechender JDBC 1.2 Treiber. Das implementierte Konzept erlaubt eine einfache Unterstützung weiterer Datenbank-Server. Der zu verwendende JDBC-Treiber samt statischer Zugangsdaten wird per Konfigurationsdatei angegeben und zur Startzeit eingelesen. Der datenbankabhängige SQL-Code, wie das Datumsformat oder datenbankspezifische Funktionen, werden durch ein besonderes Objekt gekapselt, das für jede unterstützte Datenbank zu implementieren ist. Dieses Objekt implementiert die Schnittstelle `backend.db.Database`, welche die drei folgenden Methoden spezifiziert:

1. `String sqlDateToken (java.util.Date date)` – konvertiert das Datum-Objekt in das erforderliche Datenbankformat
2. `String timestampToken ()` – liefert den Bezeichner für einen Zeitstempel
3. `BigDecimal getSequence (String key)` – liefert einen eindeutigen Wert (für einen künstlichen Primärschlüssel) in Abhängigkeit vom Parameter

Die vollständige Bezeichnung dieser Klasse wird in der Konfigurationsdatei angegeben und zur Startzeit initialisiert.

5.2 Authentisierung

Die Authentisierung stellt bei der Entwicklung von ProAdWeb ein zentrales Thema dar und wird aus diesem Grund sehr ausführlich behandelt. Im ersten Unterabschnitt wird ein Überblick über das implementierte Authentisierungsverfahren gegeben. Der darauf folgende Unterabschnitt beschreibt die Aspekte der Implementierung. Dabei wird zuerst die eingesetzte Technologie erläutert, gefolgt von der eigentlichen Umsetzung.

5.2.1 Konzept

Die Authentisierung eines Anwenders durch ProAdWeb erfordert den Besitz eines Zertifikats (das in ProAdWeb und im Browser installiert wird) sowie des passenden privaten Schlüssels (der ausschließlich im Browser installiert wird). Das Zertifikat samt privaten Schlüssel sollte, falls noch nicht vorhanden, beim eingesetzten Trustcenter beantragt werden. Die Installation des Zertifikats in einem Webbrowser erfolgt gemäß den Richtlinien der eingesetzten Software. Die Installation des Zertifikats in ProAdWeb erfolgt durch die Verknüpfung des Antragstellers, aus dem Zertifikat und des entsprechenden Anwendernamens in einer Konfigurationsdatei. Dieser Vorgang wird im Kapitel 5.2.3 ausführlich erläutert. Die Anmeldung ins

Programm erfolgt über eine Anmeldeseite. Jeder unauthentierte Zugriffsversuch auf eine andere Programmseite wird auf die Anmeldeseite umgeleitet. Während der Anmeldung ist die Eingabe eines gültigen Mandanten und der Sprache für die Anwenderschnittstelle erforderlich. Das Mandanten-Konzept stammt aus ProAd, wobei ein Mandant als eine ProAd-Instanz mit eigenem Datenbestand definiert werden kann. Auf Datenbankebene entspricht ein Mandant einem Datenbankbenutzer. Das Konzept des Datenbankbenutzers erlaubt die Aufteilung einer Datenbank in beliebig viele diskrete Bereiche. Der Betrieb von ProAd erfordert die Existenz von mindestens einem Mandanten (bzw. Datenbankbenutzer) und erlaubt den gleichzeitigen Einsatz beliebig vieler Mandanten (bzw. Datenbankbenutzer).

Technisch betrachtet wird die Authentisierung wie in Kapitel 4.2.2.2 spezifiziert durchgeführt. Die dabei anfallenden Aufgaben werden durch zwei getrennte Komponenten abgewickelt. Die Überprüfung der Integrität und Gültigkeit der digitalen Signatur und des Anwender-Zertifikats übernimmt der Apache Webserver, der, wie bereits erwähnt, für die gesamte Client-Kommunikation verantwortlich ist. Bei erfolgreicher Überprüfung wird die Anfrage an Tomcat bzw. an das Zugriffsmodul der Applikation weitergeleitet. Das Zugriffsmodul besteht aus einem Java-Servlet, das alle ankommenden Anfragen empfängt und nach der Authentisierung entsprechend weiterleitet.⁸⁸ Die Authentisierung im Zugriffsmodul besteht aus der Sicherstellung, dass das empfangene Zertifikat einen bekannten ProAd-Anwender identifiziert. Dafür sind folgende Schritte erforderlich:

- Überprüfen, ob ein Zertifikat und ein Mandant übermittelt wurden
- Überprüfen, ob das empfangene Zertifikat bekannt ist – dafür wird die entsprechende Konfigurationsdatei ausgelesen
- Überprüfen, ob das empfangene Zertifikat einen bekannten ProAd-Anwender identifiziert – dafür muss die Verbindung mit der Datenbank unter entsprechendem Benutzer aufgebaut werden (falls noch nicht geschehen) und eine passende SQL-Abfrage abgeschickt werden
- Laden der zur Laufzeit benötigten Anwenderinformationen

Da HTTP ein zustandsloses Protokoll ist, müsste dieser Vorgang bei jeder Anwenderaktion durchgeführt werden, um unauthentierte Zugriffe auf ProAdWeb zu vermeiden. Dieser Aufwand wird umgangen, in dem die Informationen eines einmal authentisierten Clients in einer, von Tomcat mittels ‚Cookies‘ oder URL-Beschreiben⁸⁹, künstlich geschaffener Sitzung gespeichert. Eine Authentisierung findet damit nur dann statt, wenn das Zugriffsmodul in der aktuellen Sitzung keine Daten einer vorangegangenen Authentisierung findet und die zugegriffene Seite nicht die Anmeldeseite ist.

5.2.2 Angewandte Technologie

Die Implementierung des Teils der Authentisierung, der von Apache durchgeführt wird, beschränkt sich auf eine eventuelle Übersetzung der Quelldateien sowie die Konfiguration der Funktionalität für die Zertifikatsprüfung und die Kommunikation mit Tomcat. Die eingesetzte Apache-Konfiguration kann im Anhang eingesehen werden. Damit Apache mit Tomcat kommunizieren kann wird die Datei mod_jk.dll in das „modules“-Verzeichnis vom Apache kopiert, Konfiguration von mod_jk.conf

⁸⁸ Vgl. Kapitel 5.3

⁸⁹ Vgl. Kapitel 4.2.2.2

durchgeführt und im Tomcat das Protokoll AJP 1.3 aktiviert. Bezüglich der einzelnen Konfigurationseinstellungen wird auf den entsprechenden Abschnitt im Anhang verwiesen.

Der verbliebene Teil der Authentisierung findet in der Applikation statt und basiert auf JAAS. JAAS ist eine Java-Technologie für Authentisierung und Autorisierung in Anwendungen. Da die Erläuterung der Autorisierung erst im Kapitel 5.3 erfolgt, wird hier zunächst nur die, für die Authentisierung eingesetzte, JAAS-Funktionalität beschrieben. Zuvor wird darauf hingewiesen, dass bei offen gebliebenen Fragen die ausführlichen Dokumentationen über JAAS im Internet unter java.sun.com/products/jaas eingesehen werden können.

Ein Konzept von JAAS ist das Loslösen des speziellen Authentisierungsverfahrens von der eigentlichen Applikation mit dem Ziel das Authentisierungsverfahren ändern zu können, ohne den Code der Applikation bearbeiten zu müssen. Dabei implementiert ein sogenanntes Login-Modul die Logik eines eingesetzten Authentisierungsverfahrens. Ein solches Login-Modul wird niemals direkt instanziiert, sondern sein voller Klassenname wird in einer Konfigurationsdatei mit weiteren Attributen angegeben und zur Startzeit durch die Java-Laufzeitumgebung initialisiert. Die Konfiguration des Login kann wie folgt aussehen:

```
ProAdWeb {  
    com.jdsoftware.proadweb.auth.cert.CertLoginModule required;  
};
```

Unter dem Bezeichner ‚ProAdWeb‘ wird diese Login-Konfiguration innerhalb einer Java-Anwendung identifiziert. Eine JAAS-Login-Konfiguration kann aus einem (Beispiel oben) oder mehreren Login-Modulen bestehen. Der Bezeichner ‚required‘ spezifiziert, dass ein insgesamt erfolgreicher Login-Vorgang das betroffene Login-Modul erfolgreich passieren muss.

Um Informationen zu erhalten, die benötigt werden, um eine Anmeldung durchführen zu können, führt ein Login-Modul sogenannte ‚Callbacks‘ durch. Dies geschieht durch Instanzieren der entsprechenden Callback-Objekte, welche die Schnittstelle `javax.security.auth.callback.Callback` implementieren, und deren Weitergabe an einen ‚CallbackHandler‘, der die Callback-Objekte mit Daten versorgt. Ein ‚CallbackHandler‘, d.h. ein Objekt welches die Schnittstelle `javax.security.auth.callback.CallbackHandler` implementiert, wird einem Login-Modul während seiner Initialisierung übergeben. Die Initialisierung erfolgt durch Instanzieren des Objektes `javax.security.auth.login.LoginContext`, das als Parameter den Bezeichner der betroffenen Login-Konfiguration und eine `CallbackHandler`-Instanz, die alle erforderlichen Callbacks behandeln kann, erhält. Die Authentisierung wird durch den Aufruf der Methode ‚login‘ des instanziierten `LoginContext`-Objekts initiiert. Für den Programmierer ist die Ausführung dieser Methode transparent. Entweder wird ein authentisierter Subject zurückgeliefert oder eine ‚Exception‘ vom Typ `javax.security.auth.login.LoginException` ausgelöst.

Die Authentisierung verläuft in zwei Schritten. Im ersten Schritt ruft `LoginContext` die Methode ‚login‘ aller beteiligten Login-Module auf, wobei jedes Modul das jeweils implementierte Authentisierungsverfahren durchführt. Wenn die

Authentisierung insgesamt erfolgreich⁹⁰ war, dann ruft LoginContext die Methode ‚commit‘ aller beteiligten Login-Module auf. Hier können mit dem authentisierten ‚Subject‘ diverse Anwenderdaten (‚Principals‘ und/oder ‚Credentials‘) verknüpft werden. Ein ‚Subject‘, genauer eine Instanz der Klasse `javax.security.auth.Subject`, repräsentiert in Java eine authentifizierte Person bzw. einen authentisierten Client. Bei Fehlschlagen einer Authentisierung, ruft LoginContext die Methode ‚abort‘ aller beteiligten Login-Module auf, womit ein korrekter Zustand der Login-Module wiederhergestellt werden soll.

5.2.3 Implementierung

Das JAAS-Konzept wurde innerhalb von ProAdWeb wie folgt implementiert. In einer XML-Konfigurationsdatei, deren Pfad ein Konfigurationsparameter von ProAdWeb ist, werden die Anwender mit ihren Zertifikaten verknüpft. Die Konfigurationsdatei hat das folgende Format:

```
<security>
  <principals>
    <identity>
      <cert>
        <unique-name>CN=Daniel Kwiotek, OU=RD, O=JD,
          C=DE</unique-name>
      </cert>

      <principal
        class="com.jdsoftware.proadweb.auth.ProAdWebPrincipal">
        ADMIN</principal>
    </identity>

    <identity>
    ...
    </identity>

    ...

  </principals>
</security>
```

Die eigentlichen Verknüpfungsdaten werden durch die Tags ‚security‘ und ‚principals‘ umklammert. Das ‚identity‘-Tag entspricht einer Zertifikat-Anwender-Verknüpfung. Die Zertifikatsdaten werden mit dem ‚cert‘-Tag eingeleitet. Das Zertifikat wird, wie bereits erwähnt, durch den eindeutigen Namen des Antragstellers identifiziert. Innerhalb des ‚Principal‘-Tags ist der Bezeichner anzugeben, mit dem sich der Anwender über den bestehenden Omnis ProAd-Client im System anmeldet. In der Datenbank ist dieser Bezeichner in der Tabelle ‚usertab‘ unter dem Feldnamen ‚username‘ gespeichert.

In Java wird ein Anwender durch eine Klasse repräsentiert, welche die Schnittstelle `java.security.Principal` implementiert. Ein Objekt dieser Klasse wird oft als ‚Principal‘ bezeichnet. Als Attribut ‚class‘ ist innerhalb des ‚Principal‘-Tags, der Konfigurationsdatei der Anwender, eine solche ‚Principal‘-Klasse anzugeben. Dies ist immer die Klasse `auth.ProAdWebPrincipal`, die den konfigurierten Anwendernamen lädt und in der Anwendung verfügbar macht.

⁹⁰ Die Interpretation einer Authentisierung als erfolgreich bzw. fehlgeschlagen erfolgt nach definierten Regeln, Vgl. JAAS Web, 2002

Das eingesetzte Login-Modul wird durch die Klasse `auth.cert.CertLoginModule` repräsentiert, die in seiner `,login'-Methode` zwei Callbacks durchführt:

- `auth.cert.CertCallback`, um das empfangene Client-Zertifikat anzufordern
- `auth.ProAdWebCallback`, um den Mandanten und die gewählte Sprache für die Anwenderschnittstelle anzufordern

Diese Callbacks werden durch den `CallbackHandler` `auth.ProAdWebCallbackHandler` bearbeitet. `,ProAdWebCallbackHandler'` erfordert zur seiner Instanziierung die Parameter der aktuellen HTTP-Anfrage, die durch eine Instanz von `javax.servlet.http.HttpServletRequest` gekapselt werden. Die Klasse `auth.cert.CertPrincAssigner` liest die Anwenderkonfiguration aus und liefert mittels der Methode `,assignPrincipal'`, die als Parameter das empfangene Client-Zertifikat erwartet, das erzeugte `Principal`-Objekt zurück, das eine Instanz der oben erwähnten Klasse `auth.ProAdWebPrincipal` ist. Der `Principal` enthält den konfigurierten Anwendernamen, der über die Methode `,getUsername'` erhältlich ist und als Parameter für die Datenbankabfrage zur Überprüfung der Existenz und der Gültigkeit des entsprechenden Anwenderkontos eingesetzt wird. Die Abfrage lautet wie folgt:

```
select URNO, username from usertab where username = [Anwendername aus  
der Konfiguration] and activeflag='Y' and expireddate>[aktuelles  
Datum]
```

Mit der Sicherstellung der Gültigkeit des Anwenderkontos ist die `,login'-Methode`, also der erste Schritt der JAAS-Authentisierung, abgeschlossen. In der Methode `,commit'` vom `,CertLoginModule'` wird der erhaltene `Principal` und das Client-Zertifikat mit dem authentisierten `Subject` verknüpft.

5.3 Autorisierung

Zur Beginn werden die Konzepte zur Autorisierung von ProAd und von Java beschrieben. Anschließend wird die Implementierung der Autorisierung beschrieben.

5.3.1 Konzept von ProAd

Jedes Modul von ProAd (Menu, Programmfenster) verlangt, wie bereits in Kapitel 3.2 erläutert, den Besitz bestimmter anwenderbezogener Rechte, um den Zugriff bzw. eine Aktion zu erlauben. Die Rechte eines Anwenders sind in der ProAd-Datenbank gespeichert und können mittels eindeutiger Namen identifiziert werden. Jedes Recht hat einen Wert, der zwischen 1 und 5 liegt und die folgende Bedeutung hat:

- 1 = kein Zugriff
- 2 = Lesen
- 3 = Änderungen durchführen
- 4 = Datensätze anlegen
- 5 = Daten löschen

Für die Autorisierung eines Vorgangs muss der erforderliche Wert eines Rechts mit dem Wert des aktuellen Anwender verglichen werden. Ist der Wert größer oder gleich dem erforderlichen, erfolgt eine Autorisierung des Vorgangs.

5.3.2 Konzept von Java

Die Autorisierung in Java, genauer die JAAS-Autorisierung, basiert auf der Vergabe von sogenannten ‚Permissions‘ und ist auf die vorausgegangene JAAS-Authentisierung angewiesen. Bei ‚Permissions‘ handelt es sich um Java-Klassen, welche die abstrakte Klasse `java.security.Permission` implementieren bzw. erweitern. ‚Permissions‘ werden ausgewählten ‚Principals‘ in einer Konfigurationsdatei, der Java-Policy-Datei, erteilt und zur Laufzeit an betroffenen Programmstellen überprüft. Für den ordnungsmäßigen Einsatz der JAAS-Autorisierung ist damit der Betrieb von Java mit sogenanntem „SecurityManager“ erforderlich, der die Information aus Policy-Datei im System verfügbar macht.⁹¹ Die Konfiguration der ‚Permissions‘ erfolgt nach folgendem Schema:⁹²

```
grant [signiert von], [Herkunft des Codes], [Verweis auf einen
Principal] {
    permission [Permission-Klasse] [Zielobjekt], [Aktion];
    ....
    permission [Permission-Klasse] [Zielobjekt], [Aktion];
};
```

Das Vorkommen und die Reihenfolge der drei Bezeichner hinter dem Schlüsselwort ‚grant‘ ist optional. Für dieses Projekt sind nur Einträge mit einem Verweis auf einen ‚Principal‘ von Interesse. Ein ‚Permission‘-Eintrag beginnt mit dem Schlüsselwort ‚permission‘ gefolgt von dem vollständigen Klassennamen der erteilten ‚Permission‘. Die Angabe eines Zielobjekts bzw. einer Aktion ist von der erteilten ‚Permission‘ abhängig. Der Verweis auf einen ‚Principal‘ hat die folgende Form:

```
principal [Principal-Klasse] [Principal-Name]
```

Ein ‚Principal‘ wird anhand seines Namens identifiziert, der durch den Aufruf der Methode ‚getName‘ erhältlich ist.

Damit eine Überprüfung stattfinden kann, muss der zuvor authentifizierte ‚Subject‘, mit dem aktuellen ‚Access Control Context‘⁹³ verknüpft werden, wodurch die gehaltenen ‚Principals‘ dem Laufzeitsystem bekannt werden. Dies erfolgt über den folgenden Aufruf:

```
Subject.doAs(subject, privilegedAction)
```

Der Parameter ‚subject‘ ist eine Referenz auf den bereits authentifzierten ‚Subject‘. Der Parameter ‚privilegedAction‘ verweist auf die Instanz einer Klasse, welche die Schnittstelle `java.security.PrivilegedAction` implementiert. Diese Schnittstelle besteht aus einer einzigen Methode ‚run‘, die den Code, der mit dem ‚Subject‘ verknüpft wird, kapseln soll. Die eigentliche Überprüfung der Existenz einer entsprechenden ‚Permission‘ erfolgt über den folgenden Aufruf:

```
AccessController.checkPermission(permission)
```

⁹¹ Vgl. Java 1.4 Security Web, 2002a

⁹² Die vollständige Syntax der Java-Policy-Datei wird unter Java 1.4 Security Policy Web, 2002c, beschrieben

⁹³ Dabei handelt es sich im Wesentlichen um den Stack des aktuell ausgeführten Threads. Für weiteres zu diesem Konzept vgl. Java 1.4 Security Architecture Web, 2002b

Der Parameter ist eine Referenz auf die zu überprüfende ‚Permission‘. Der Aufruf hat zur Folge, dass alle vom aktuellen Subject gehaltenen ‚Principals‘ auf den Besitz einer ‚Permission‘ überprüft werden, welche die geforderte ‚Permission‘ impliziert. Die Beziehung *Permission A impliziert Permission B* kann intuitiv als *Permission A erlaubt mindestens genauso viel wie Permission B* verstanden werden und wird im Detail von der Methode ‚implies‘ spezifiziert, die jede ‚Permission‘ implementieren muss. Diese Methode erhält als Parameter eine Referenz auf eine zu überprüfende ‚Permission‘. Sobald mindestens ein ‚Principal‘, der vom überprüften ‚Subject‘ gehalten wird, über keine Permission verfügt, welche die geforderte impliziert, schlägt die Überprüfung fehl. Anderenfalls ist der Subject autorisiert die entsprechende Aktion durchzuführen.

5.3.3 Implementierung

Beim Entwurf des Konzepts zur Autorisierung wurde besonderer Wert auf einen geringen Konfigurationsaufwand der Policy-Datei gelegt. Würde beispielsweise ein ProAd-Recht auf eine Java-Permission abgebildet werden, dann müsste pro Anwender ein ‚grant‘-Eintrag mit entsprechenden erteilten Permissions gepflegt werden. Ein solcher zusätzlicher Aufwand sollte möglichst vermieden werden.

Das in ProAdWeb eingesetzte Konzept zur Autorisierung integriert das Konzept aus ProAd in bestehende, und im vorherigen Kapitel erläuterte, JAAS-Mechanismen und baut dabei auf zwei Klassen auf:

- Ein Objekt der Klasse `auth.ProAdWebPrincipal` repräsentiert den Anwender und stellt alle erforderlichen Anwenderdaten zur Verfügung
- Die Klasse `auth.ProAdWebPermission` bildet das vorhandene ProAd-Rechtesystem auf das Konzept der Java-Permissions ab

Die Konfiguration der Policy-Datei beschränkt sich auf die wenigen folgenden Zeilen:

```
grant codeBase "file:[Pfad zur Webapplikation]-" principal
com.jdsoftware.proadweb.auth.ProAdWebPrincipal "ProAdWebUser" {
    permission com.jdsoftware.proadweb.auth.ProAdWebPermission
        "userRights";
};
```

Diese Konfiguration hat zur Folge, dass jeder ‚ProAdWebPrincipal‘ mit dem Namen ‚ProAdWebUser‘⁹⁴, der den Code der Webanwendung ausführt, eine ‚ProAdWebPermission‘ mit dem Namen ‚userRights‘ erteilt bekommt. Die Namen des Principals ‚ProAdWebPrincipal‘ bzw. der Permission ‚ProAdWebPermission‘ können zur Programmlaufzeit durch den Aufruf der jeweiligen ‚getName‘-Methode abgefragt werden und haben keine weitere Verwendung als eben diese Verknüpfung herzustellen. Den eigentlichen Namen des Principals, genauer den Anwendernamen wie in der ProAd-Datenbank gespeichert, liefert die Methode ‚getUsername‘, wie bereits im Abschnitt über die Authentisierung erwähnt.

Die Methode ‚implies‘ der Klasse ‚ProAdWebPermission‘ erwartet als Parameter eine Permission gleicher Klasse, die über eine ‚action‘ der folgenden Form verfügen muss:

```
[Name des Rechts].[erforderlicher Wert des Rechts]
```

⁹⁴ Das sind alle Principal-Objekte dieser Klasse

Der folgende Aufruf überprüft das Recht auf Löschen in der Zeit- und Materialerfassung:

```
AccessController.checkPermission(  
    new ProAdWebPermission("userRights",  
        "time_material_registration.5"));
```

Die Überprüfung fällt positiv aus, wenn der Aufruf keine ‚Exception‘ auslöst bzw. wenn die Methode ‚implies‘ der aufgerufenen Permission den Wert ‚false‘ (‚falsch‘) zurück liefert. In der Methode ‚implies‘ der Klasse ‚ProAdWebPermission‘ wird zuerst der Name des aktuellen Anwenders ermittelt und die ‚action‘ der als Parameter erhaltenen ‚ProAdWebPermission‘ in die Teile ‚Name des Rechts‘ und ‚erforderlicher Wert des Rechts‘ getrennt. Anschließend wird der vom Anwender gehaltene Wert des betroffenen Rechts von der Datenbank abgefragt. Falls der gehaltene Wert kleiner ist als der erforderte oder falls die Datenbankabfrage fehlschlägt wird ‚false‘ zurück geliefert, andernfalls ist der Anwender autorisiert.

Der letzte noch verbliebene Stichpunkt ist die Verknüpfung des authentisierten Subject mit dem Code. Dafür ist das Servlet verantwortlich, das die Authentisierung durchführt. Wenn in den Daten der aktuellen Sitzung ein Subject gefunden oder ein solcher durch eine erfolgreiche Authentisierung erzeugt wurde, dann wird die Kontrolle an den ‚RequestDispatcher‘, der die Schnittstelle `java.security.PrivilegedAction` implementiert, über den folgenden Aufruf übergeben:

```
Subject.doAs(subject, new RequestDispatcher(...))
```

Der ‚RequestDispatcher‘ ermittelt anhand der angeforderten URL, an welches Modul der Geschäftslogik die Kontrolle weitergegeben werden soll.

6. Fazit

Die Entwicklung einer sicheren Web-Schnittstelle für die bestehende Enterprise-Anwendung ProAd Professional mit PKI-Technologien ist durch die Integration der drei bereits bewährten Konzepte – Internet, PKI, ProAd – charakterisiert. Mit SSL/TLS steht eine breit unterstützte Technologie zur Verfügung, die PKI-Konzepte in Internet-Protokolle integriert. Für die Integration von Internet und PKI mit ProAd ist Omnis Studio, die aktuelle Plattform von ProAd, nicht geeignet. Auf der Anwendungsebene wird ein proprietäres Protokoll eingesetzt, und die erforderlichen Sicherheitskonzepte werden nicht ausreichend unterstützt. Mit Java steht dagegen eine frei erhältliche Plattform zur Verfügung, die eine plattformunabhängige Programmiersprache und mit der Servlet-Technologie ein ausgereiftes und auf Standards basierendes Konzept für die Erstellung von Webanwendungen bietet. Das vorhandene Sicherheitskonzept unterstützt alle erforderlichen PKI-Funktionen und verfügt mit JAAS über ein einheitliches und flexibles Konzept zur Authentisierung und Autorisierung. Des Weiteren sind für Java diverse Komponenten kostenlos erhältlich. Das *Jakarta*-Projekt von Apache liefert mit Tomcat und Velocity zwei Komponenten für den Prototyp und dient daher als Beispiel.

Der implementierte Prototyp erfüllt die ProAd Professional Spezifikation der Zeit- und Materialerfassung vollständig, d.h. er agiert genau wie ein klassischer ProAd-Client und kann parallel dazu betrieben werden. Der Zugriff auf eine bestehende ProAd-Datenbank erfordert keinen weiteren Aufwand, als die Zugangsparameter in einer Konfigurationsdatei einzutragen. Laufzeittests des Prototyps zeigten, dass bei ausreichender Netzwerkbandbreite, die Datenbank den Engpass ausmacht. Laufzeitprobleme dieser Art traten lediglich bei Abfrage großer Datenbestände auf. In sonstigen Fällen waren die Antwortzeiten akzeptabel.

Der Prototyp hat, wie ProAd, den Charakter einer Standardsoftware, ist allerdings nicht vollständig - wie im Kapitel 4.1 gefordert - auf Standards basierend, da als Webserver Apache vorausgesetzt wird. Diese Einschränkung kann durch die Entwicklung eines eigenen Moduls zur Zertifikatüberprüfung eliminiert werden. Grundsätzlich könnte diese Aufgabe der eingesetzte Servletcontainer übernehmen, falls dieser über solche Funktionalität verfügt. Leider ist die Zertifikatunterstützung von Tomcat noch nicht ausreichend funktionsfähig und kann deshalb nicht eingesetzt werden.

Die im Kapitel 4.2 entworfene Architektur konnte nicht vollständig implementiert werden. Das Modul der Administration wurde aus Zeitgründen vollständig ausgelassen. Für den Echtbetrieb ist jedoch eine Konfiguration der Anwender-Zertifikat-Verknüpfung über eine leistungsfähige Anwenderschnittstelle wünschenswert, insbesondere wenn die Anwenderzahl hoch ist. In einen weiteren Schritt sollte die Administration um die LDAP-Unterstützung erweitert werden. Dagegen wird es im Moment nicht möglich sein, eine - von einzelner Lösung unabhängige - automatisierte Trustcenter-Kommunikation zu implementieren. Dafür fehlt es an entsprechenden Standards bzw. an der einheitlichen Unterstützung vorhandener Standards von den Herstellern. Dies ist insbesondere bedauerlich, da durch eine automatisierte Kommunikation mit dem Trustcenter der Prozess der Verwaltung von Anwenderzertifikaten wesentlich effizienter gestaltet werden könnte.

Abschließend ist festzustellen, dass PKI grundsätzlich dazu geeignet ist eine bestehende Enterprise-Anwendung mit einem sensiblen Datenbestand bei der Erweiterung um eine Webschnittstelle abzusichern. Java ist dabei eine geeignete Plattform für die Implementierung der Webschnittstelle. Der Aufwand für die Implementierung ist von den besonderen Gegebenheiten der Enterprise-Anwendung abhängig. Für ProAd war es erforderlich, neben der Anwenderschnittstelle, auch die Anwendungslogik erneut zu implementieren.

Ein weiteres Resultat der vorliegenden Arbeit ist, dass die entworfene und implementierte Architektur eine wirkliche Alternative für die nachfolgende Generation von ProAd Professional darstellt.

Quellennachweis

Literatur

Adams, Carlisle / Lloyd, Steve (1999): Understanding Public-Key Infrastructure: Concepts, Standards and Deployment Considerations, Indianapolis, 1999

Austin, Tom (2001): PKI, John Wiley & Sons, 2001

Crawford, William /Hunter, Jason (2001): Java Servlet Programming, 2. aktualisierte Auflage, O'Reilly, 2001

Curry, Ian (1997a): Trusted Public-Key Infrastructures, Entrust Technologies Limited, 1997

Curry, Ian (1997b): An Introduction to Cryptography, Entrust Technologies Limited, 1997

Edwards, Jeri / Harkey, Dan / Orfali, Robert (1997): Abenteuer Client/Server – The Essential Client/Server Survival Guide, Bonn, 1997

Günther, Hans-Otto / Tempelmeier, Horst (2000): Produktion und Logistik, 4. neubearb. u. erw. Aufl., Berlin 2000

Häcker, Joachim (1998): Internet-Banking – Gestaltungsformen, Rechtsfragen, Sicherheitsaspekte, Wiesbaden, 1998

KPMG (1998): „Electronic Commerce Survey“, 1998

National Bureau of Standards (1995): Federal Information Processing Standards Publication 180-1: Secure Hash Standard, National Technical Information Service, Springfield, Virginia, 1995;
auch im Internet erhältlich unter <http://www.itl.nist.gov/fipspubs/fip180-1.htm>, 29.08.2002

Phillips, Lee Ann (2000): Using XML, 2000

Rihaczek, Karl (1984): Datenverschlüsselung in Kommunikationssystemen, 1984

Rivest, R.L. / Shamir, A. / Adleman, L.M. (1978): A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM (2) 21, 1978

Schulzki-Haddouti, Christiane (1999): Markt- oder Staatsmacht - Streit um digitale Signaturen, in: c't 1/1999, S. 58

Sewberry, Jennifer / Pieprzyk, Joseph (1989): Cryptography – An Introduction to Computer Security, 1989

Stallings, William (1995): Sicherheit im Datennetz, München, 1995

Request for Comments

Adams C. / Farrell, S. (1999): Internet X.509 Public Key Infrastructure Certificate Management Protocols, Internet Request for Comments (RFC) 2510, 1999

Adams C. / Kemp D. / Myers M. / Solo D. (1999): Internet X.509 Certificate Request Message Format, Internet Request for Comments (RFC) 2511, 1999

Chokani, S. / Ford, W. (1999): Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, Internet Request for Comments (RFC) 2527, 1999

Dierks, T. / Allen C. (1999): The TLS Protocol Version 1.0, Internet Request for Comments (RFC) 2246, 1999

Dusse, S./ Hoffman, P. / Ramsdell, B. / Lundblade, L. / Repka, L. (1998a): S/MIME Version 2 Message Specification, Internet Request for Comments (RFC) 2311, 1998

Dusse, S./ Hoffman, P. / Ramsdell, B. / Lundblade, L. / Repka, L. (1998b): S/MIME Version 2 Certificate Handling, Internet Request for Comments (RFC) 2312, 1998

Hardcastle-Kille (1993): A String Representation of Distinguished Names (OSI-DS 23 (v5)), Internet Request for Comments (RFC) 1485, 1993

Housley, R. / Ford, W. / Polk W. / Solo, D. (1999): Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Internet Request for Comments (RFC) 2459, 1999

Kent, S. / Atkinson R. (1998): Security Architecture for the Internet Protocol, Internet Request for Comments (RFC) 2401, 1998

Rivest, R.L. (1992a): The MD4 Message Digest Algorithm, Internet Request for Comments (RFC) 1320, 1992

Rivest, R.L. (1992b): The MD5 Message Digest Algorithm, Internet Request for Comments (RFC) 1321, 1992

Thayer R. / Doraswamy N. / Glenn R. (1998): IP Security Document Roadmap, Internet Request for Comments (RFC) 2411, 1998

Internet

AES Web, (2002): AES Home Page, <http://csrc.nist.gov/encryption/aes/>, 29.08.2002

Apache Web (2002): TheApache HTTP Server Project, <http://httpd.apache.org>, 29.08.2002

Datev Web (2002): DATEVstadt, www.datev.de, 29.08.2002

Eclipse Web (2002): Eclipse.org Main Page, <http://www.eclipse.org>, 29.08.2002

Frontbase Web (2002): Frontbase Firmen-Homepage, <http://www.frontbase.com>, 29.08.2002

JAAS Web (2002): Java (TM) Authentication and Authorization Service, <http://java.sun.com/products/jaas/>, 29.08.2002

Jakarta Velocity (2002): Velocity, <http://jakarta.apache.org/velocity>, 29.08.2002

Java 1.4 Security (2002a): Security, <http://java.sun.com/j2se/1.4/docs/guide/security/>, 29.08.2002

Java 1.4 Security (2002b): Java Security Architecture, <http://java.sun.com/j2se/1.4/docs/guide/security/spec/security-specTOC.fm.html>, 29.08.2002

Java 1.4 Security Web (2002c): Default Policy Implementation and Policy File Syntax, <http://java.sun.com/j2se/1.4/docs/guide/security/PolicyFiles.html>, 29.08.2002

Java Servlet Web (2002): Java (TM) Servlet Technology, <http://java.sun.com/products/servlet/>, 29.08.2002

JDBC Web (2002): JDBC (TM) Technology, <http://java.sun.com/products/jdbc/>, 29.08.2002

JSP Web (2002): Java Server Pages (TM) Technology, <http://java.sun.com/products/jsp/>, 29.08.2002

Lexware Web (2002): Lexware Produkte, www.lexware.com, 29.08.2002

Mod_ssl Web (2002): mod_ssl: The Apache Interface to OPenSSL, <http://www.modssl.org>, 29.08.2002

Netscape Web (2002): Netscape.com, www.netscape.com, 29.08.2002

Omnis Web (2002): Omnis Studio – web application development tool for Windows, Linux, Solaris and Mac OS, <http://www.omnis-software.com>, 29.08.2002

Omnis Webclient Web (2002): Omnis – Web Client, <http://www.omnis.net/products/webclient/index.html>, 29.08.2002

Oracle Web (2002): Oracle Corporation, <http://www.oracle.com>, 29.08.2002

RSA Security (2002): RSA Laboratories Cryptography FAQ, <http://www.rsasecurity.com/rsalabs/faq/>, 29.08.2002

Sage Web (2002): The Sage Group plc, <http://www.sage.com>, 29.08.2002

SAP Web (2002): SAP, <http://www.sap.com>, 29.08.2002

Tomcat Web (2002): The Jakarta Site – Apache Tomcat,
<http://jakarta.apache.org/tomcat>, 29.08.2002

Anhang

Installation und Konfiguration

Für die Lauffähigkeit von ProAdWeb ist die Installation, Konfiguration und Initialisierung von folgenden Komponenten erforderlich:

- Apache
- Tomcat
- ProAdWeb-Anwendungsdateien
- ProAd-Datenbank

Installation und Konfiguration von Apache und Tomcat

Apache und Tomcat sind entsprechend der Richtlinien der eingesetzten Distribution zu installieren. Da Apache mit SSL erfordert wird, muss eine Distribution durch manuelle Übersetzung von Apache mit einer passenden ‚mod_ssl‘-Version und ‚openssl‘-Version erzeugt werden.

Die Konfiguration von mod_ssl erfolgt entweder in der Datei httpd.conf oder in einer eigenen Datei (z.B. mod_ssl.conf), die per ‚INCLUDE‘ in httpd.conf eingebunden wird. Die Standardkonfiguration von mod_ssl ist um den Export des Zertifikats an ProAdWeb wie folgt zu erweitern:

```
<Location ~ "/proadweb/">  
    SSLOptions +StdEnvVars +ExportCertData +CompatEnvVars +StrictRequire  
</Location>
```

Desweiteren ist die Angabe von Zertifikatspfaden des Servers, der CA und der CRL sowie die Aktivierung der Client-Authentisierung erforderlich.

Damit Apache die entsprechenden Anfragen an Tomcat weiterleiten kann, muss die Kommunikation zwischen den Komponenten konfiguriert werden, wofür das Apache-Modul ‚mod_jk‘ verwendet wird. Damit mod_jk von Apache verwendet werden kann, ist eine passende mod_jk-Binärdatei in das Apache-Unterverzeichnis ‚modules‘ zu kopieren und die folgende Konfiguration in httpd.conf einzubinden:

```
LoadModule jk_module modules/mod_jk.dll  
  
<IfModule mod_jk.c>  
  
    JkWorkersFile conf/workers.properties  
    JkMount /proadweb/* ajp13  
  
</IfModule>
```

Unter ‚JkWorkersFile‘ wird eine weitere Konfigurationsdatei angegeben, die Einstellungen bezüglich AJP sowie Angaben der Pfade von Tomcat und Java beinhaltet. Eine solche, ausführlich dokumentierte, Konfigurationsdatei liegt meistens der mod_jk-Binärdatei bei. Die mod_jk-Binärdatei kann entweder von der Apache-Homepage oder von der Tomcat-Homepage bezogen werden .

Innerhalb der Tomcat-Konfigurationsdatei server.xml muss der folgende Code für die Aktivierung des Protokolls AJP1.3 auskommentiert werden:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector className="org.apache.ajp.tomcat4.Ajp13Connector"
           port="8009" minProcessors="5" maxProcessors="75"
           acceptCount="10" debug="0"/>
```

Installation und Konfiguration der ProAd-Datenbank

ProAdWeb ist ein Client der ProAd Professional Datenbank, welche auf einem unterstützten Datenbankserver – Oracle oder Frontbase – zu installieren ist. Die Installation der ProAd-Datenbank besteht aus der Anlage der Datenbankobjekte (Tabellen, Views, Constraints usw.) und Initialisierungsdaten. Die ausführliche Anleitung der Installation der ProAd Professional Datenbank kann bei der J+D Software AG angefragt werden.

Installation und Konfiguration von ProAdWeb

Da ProAdWeb eine Java-Webanwendung ist, erfolgt die Installation einfach durch das Kopieren des Archivs proadweb.war in das Verzeichnis TOMCAT_HOME/webapps. Alternativ kann unter TOMCAT_HOME/webapps ein Unterverzeichnis ‚proadweb‘ für die Programmdateien erstellt werden.

ProAdWeb wird über drei Dateien konfiguriert. In der Datei web.xml wird die Umleitung aller ankommenden Anfragen auf das Authentisierungsservlet wie folgt konfiguriert:

```
<web-app>

  <servlet>
    <servlet-name>
      RequestListener
    </servlet-name>
    <servlet-class>
      com.jdssoftware.proadweb.RequestListener
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>RequestListener</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>

</web-app>
```

Da keine Angaben bezüglich des Session-Timeouts vorgenommen werden, beträgt diese standardgemäß 30 Minuten.

In ein aus ProAdWeb zugängliches Verzeichnis (Bsp. WEB-INF/classes) sind die Hauptkonfigurationsdatei ‚proadweb.properties‘ sowie die Dateien mit einzusetzenden Programmtexten zu kopieren. Die Parameter von proadweb.properties dienen im wesentlichen der Konfiguration des Datenbankzugangs und der Pfade (beispielsweise der Pfad der Anwenderkonfigurationsdatei) und sind im einzelnen hinreichend dokumentiert. Der Name einer Datei mit Programmtexten muss die folgende Form haben:

```
TextBundle_[Sprache]_[Land].properties
```

Sprache und Land sind die ISO-Kürzel für die Sprache und das Land der verfassten Texte. Das zur Programmlaufzeit erzeugte Locale-Objekt wird ebenfalls mit solchen

ISO-Kürzeln initialisiert und dient unter anderem der Identifizierung der einzusetzenden Datei mit Programmtexten.

In der Policydatei von Tomcat, abhängig von der eingesetzten Version tomcat.policy bzw. cataliny.policy, müssen der Anwendung folgende Permissions, neben der ProAdWebPermission wie in Kapitel 5.3.3 beschrieben, zugewiesen werden:

```
//authentication permissions
permission javax.security.auth.AuthPermission "createLoginContext.ProadWeb";
permission javax.security.auth.AuthPermission "modifyPrincipals";
permission javax.security.auth.AuthPermission "modifyPublicCredentials";
permission javax.security.auth.AuthPermission "doAsPrivileged";
permission javax.security.auth.AuthPermission "getSubject";

//db permission
permission java.net.SocketPermission "192.168.5.241:1201-", "connect,resolve";

//permissions vor velocity.jar, remove when get it running from WEB-INF/lib
permission java.util.PropertyPermission "*", "read";
permission java.io.FilePermission "${proadweb.path}${/}velocity.log", "write";
```

Der Parameter 'proadweb.path' steht für den Pfad von ProAdWeb und wird dem System beim Start von Tomcat mitgeteilt.

Beim Einsatz von Tomcat 4.0.x hat die Vergabe von Permissions an jar-Dateien im ,lib'-Verzeichnis bzw. an das gesamte Verzeichnis keinen Effekt. Hierzu konnte keine Erklärung auf den Tomcat-Webseiten bzw. in den entsprechenden Foren gefunden werden. Ein möglicher Ausweg ist die Erteilung der betroffenen Permissions ohne den ,codebase'-Parameter. Falls dies aus diversen Gründen nicht gewollt ist, besteht eine weitere Alternative im Einsatz von Tomcat 3.3.x.

Initialisierung

Die Initialisierung von ProAdWeb erfolgt in drei Schritten:

- Starten der ProAd-Datenbank,
- Starten von Apache,
- Starten von Tomcat

Falls die Datenbank noch nicht gestartet ist, muss dieser Vorgang durchgeführt werden. Beim Start von Apache muss eventuell das Passwort für die Verwendung des privaten Schlüssels des Server-Zertifikats angegeben werden.

Beim Start von Tomcat müssen folgende Laufzeitparameter angegeben werden:

- Der Parameter `java.security.manager` veranlasst Tomcat-Betrieb mit Security-Manager
- Der Parameter `java.security.auth.login.config == [JAAS-Konfigurationsdatei]` übermittelt dem Laufzeitsystem den Pfad der JAAS-Konfigurationsdatei
- Der Parameter `proadweb.path = [ProAdWeb-Pfad]` übermittelt dem Laufzeitsystem den Pfad zur Webanwendung, der für die Konfiguration von Permissions in der Policydatei erforderlich ist

Programmcode

Der Programmcode besteht aus Java-Quelldateien und Velocity-Templates. Die Java-Quelldateien implementieren die Anwendung und sind in folgende Packages aufgeteilt:

- Das Package `com.jdsoftware.proadweb` beinhaltet die allgemein für den Betrieb benötigten Klassen, wie das Servlet zur Authentisierung.
- Das Package `com.jdsoftware.proadweb.auth` beinhaltet die generell für die Authentisierung und Autorisierung erforderlichen Klassen, wie `CallbackHandler`, `Principals` und `Permissions`.
- Das Package `com.jdsoftware.proadweb.auth.cert` beinhaltet das auf Zertifikaten basierendes Login-Modul und weitere dafür erforderlichen Klassen. Ein großer Teil des hier notierten Codes basiert auf einem JAAS-Praktikum, das am Fachgebiet ‚Theoretische Informatik‘ von Marcus Lippert betreut wurde.
- Das Package `com.jdsoftware.proadweb.backend.db` beinhaltet alle für die Kommunikation mit der ProAd-Datenbank erforderlichen Klassen.
- Das Package `com.jdsoftware.proadweb.dialog` beinhaltet alle für den Aufbau von Dialogen mittels Velocity-Templates und das Management von Dialogdaten verantwortlichen Klassen.
- Das Package `com.jdsoftware.proadweb.util` beinhaltet Klassen die allgemein benötigte Dienste anbieten, wie Formatierungen von Daten oder Laden von Properties.
- Das Package `com.jdsoftware.proadweb.util.calendar` beinhaltet alle für das Kalender-Modul erforderliche Klassen.

Die Velocity-Templates dienen der Darstellung von Programmdialogen. Die Templates werden zur Programmlaufzeit geladen, mit Daten gefüllt und als HTML-Seiten zum Client gesendet.

Der vollständige Programmcode kann der beigelegten CD-ROM entnommen werden.