Diplomarbeit:

Verbindliche Verträge in signiertem XML

Verfasser: Thomas Kunz

Betreuer: Prof. Dr. Kurt Geihs
Prof. Dr. Rüdiger Grimm

Johann Wolfgang Goethe - Universität

Frankfurt am Main

Fachbereich Biologie und Informatik
Institut für Informatik
Professur Verteilte Systeme und Betriebssysteme (VSB)

in Zusammenarbeit mit

GMD - Forschungszentrum Informationstechnik GmbH

Darmstadt

Institut für Sichere Telekooperation (SIT)

Frankfurt / M., 22. März 2001

Erklärung

Hiermit versichert der Unterzeichnende, die vorliegende Arbeit **Verbindliche Verträge in signiertem XML** selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel verfasst zu haben.

Frankfurt am Main, den 22. März 2001 _____

(Thomas Kunz)

Danksagung

Mein Dank richtet sich an Omid Amirhamzeh Tafreschi und Dr. Andreas U. Schmidt, die mich beim Verfassen dieser Arbeit mit Rat und Tat unterstützt haben. Ein besonderer Dank richtet sich an Michael Herfert, der es mir ermöglichte, die Ergebnisse dieser Arbeit auf der CeBIT 2001 zu präsentieren.

Zusammenfassung

Eine Arbeitsgruppe des W3C beschäftigt sich gegenwärtig mit der Entwicklung einer Spezifikation für XML - Signaturen. Im Rahmen dieser Arbeit wurde eine API zur Erzeugung und Verifizierung von XML - Signaturen implementiert. Die Arbeit stellt die Spezifikation und die entwickelte API vor, und gibt eine Einführung in die nötigen Grundlagen, wie XML und Kryptographie.
Desweiteren zeigt die Arbeit, wie sich das BAKO - Protokoll mit Hilfe von XML - Signaturen darstellen lässt. Das BAKO - Protokoll ist ein an der GMD entwickeltes Geschäftsprotokoll zum Erzielen eines gegenseitigen nichtabstreitbaren Übereinkommens zwischen zwei Handelspartnern.
Ziel der Arbeit ist die Implementierung einer Demonstrations - Applikation, durch die ein realistisches Handelsszenario auf Basis des BAKO - Protokolls simuliert wird. Aufgrund der besonderen Bedeutung von Online - Auktionen - sie stellen eines der am weitesten verbreiteten Geschäftsmodelle im Internet dar, weisen jedoch noch starke Mängel im Bereich Sicherheit auf - wurde ein sicheres Auktionsmodell als Handelsszenario gewählt.
Ein weiterer wichtiger Aspekt dieser Arbeit ist die Darstellung digitaler Signaturen. Da es sich bei digitalen Signaturen nur um Folgen von Bits handelt, ist ihre Darstellung im Gegensatz zu handschriftlichen Unterschriften völlig offen. Es werden die Probleme bei der Präsentation von Signaturen und die Anforderungen des Signaturgesetzes diskutiert, und es wird eine Lösung auf Basis von XSLT vorgestellt.

Inhaltsverzeichnis

Abkürzungsverzeichnis **xi**

1 Einleitung **1**
1.1 Ziel der Arbeit 2
1.2 Projekthintergrund 3
1.3 Gliederung der Arbeit 3

2 Einführung in XML **5**
2.1 XML 6
2.2 DTD 8
2.3 Namespaces 9
2.4 XSLT und XSL 10
2.5 XPath 12
2.6 DOM und SAX 13

3 Grundlagen der Kryptographie **15**
3.1 Kryptographische Verfahren 16
3.2 Digitale Signaturen 20
3.3 Zertifikate 22
3.3.1 Hierarchische PKI 22
3.3.2 Web of Trust 24
3.3.3 PKIs gemäß Signaturgesetz 25
3.3.4 Formate für Zertifikate und Namen 26
3.4 SSL - Secure Socket Layer 27

4 XML-Signaturen **29**
4.1 Überblick 30
4.2 Beschreibung der wichtigsten Elemente 32
4.2.1 Signature und SignedInfo 32
4.2.2 Reference 33
4.2.3 SignatureValue 34
4.2.4 KeyInfo 34
4.2.5 Object 34

4.2.6 Manifest 35
4.2.7 Verwendete Algorithmen 35
4.3 Canonical XML 35
4.4 Beschreibung der Java-API 37
4.4.1 Paketübersicht 37
4.4.2 Beschreibung der Klassen 38

5 Das BAKO - Protokoll 43
5.1 BAKO mit XML - Signaturen 44

6 Darstellung von digitalen Signaturen 51
6.1 Präsentationsproblem 52
6.2 Darstellung von XML - Signaturen 52
6.3 Bewertung gemäß Signaturgesetz 58

7 Elektronische Auktionen 62
7.1 Eigenschaften 63
7.2 Klassifikationen 64

8 Implementierung einer Online - Auktion 69
8.1 Sicherheitsanforderungen 70
8.2 Umsetzung 72
8.2.1 Allgemeines 72
8.2.2 Protokoll der Gebotsabgabe 73
8.2.3 Anonymität 78
8.2.4 Gewinnerermittlung 79
8.2.5 Fairness 81
8.3 Architektur 82
8.3.1 Server 85
8.3.2 Client 87

9 Fazit und Ausblick 93

A BAKO - XML - Dokumente 96

B Beschreibung einer Auktion in XML 101

Literaturverzeichnis 107

Abbildungsverzeichnis

2.1 XSLT - Transformation 11
2.2 Vergleich von DOM und SAX APIs 13

3.1 Cäsar-Chiffre 16
3.2 Symmetrische Verschlüsselung 17
3.3 Asymmetrische Verschlüsselung 19
3.4 Digitale Signaturen 21
3.5 Zertifizierungshierarchie 23
3.6 Web of Trust 24
3.7 Zertifizierungshierarchie gemäß Signaturgesetz 25
3.8 Das SSL - Protokoll im TCP/IP - Referenzmodell 27

4.1 Grundstruktur einer XML - Signatur 32
4.2 Darstellungsformen eines XML - Dokuments 36
4.3 Klasse „KeyInfo“ 38
4.4 Klasse „Reference“ 39
4.5 Klasse „Manifest“ 40
4.6 Klasse „XMLSignature“ 41
4.7 Klasse „XMLSignatureValidation“ 42
4.8 Klasse „ValidationInfo“ 42

5.1 Schematische Darstellung des BAKO - Protokolls 44
5.2 *Offer* - Dokument 46
5.3 *Accept* - Dokument 48
5.4 *Confirm* - Dokument 50

6.1 Kontext eines signierten XML-Dokuments 54
6.2 Darstellung einer XML - Signatur (Screenshot) 56
6.3 Detaillierte Darstellung (Screenshot) 58

8.1 Protokoll der Gebotsabgabe 73
8.2 Angebot des Auktionators (Screenshot) 75
8.3 Gebotsliste (Screenshot) 77
8.4 Architektur von *sintaX* 82
8.5 Benutzerschnittstelle für den Server (Screenshot) 87

8.6 Dialogbox zur Eingabe des Passworts (Screenshot) 89
8.7 Index der abgegebenen Gebote (Screenshot) 90
8.8 Benutzerschnittstelle für den Proxy (Screenshot) 91

Abkürzungsverzeichnis

AES Advanced Encryption Standard

API Application Programming Interface

B2B Business to Business

B2C Business to Consumer

BAKO Basis Kommunikation

C2C Consumer to Consumer

CA Certification Authority

CAST Kryptoalgorithmus von Carlisle und Stafford

CSS Cascading Stylesheet

DES Digital Encryption Standard

DN Distinguished Name

DOM Document Object Model

DSA Digital Signature Algorithm

DTD Document Type Definition

EU Europäische Union

FIFO First In First Out

FTP File Transfer Protocol

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

IAIK Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie an der TU Graz

ID Identification Number

IDEA International Data Encryption Algorithm

IETF Internet Engineering Taskforce

IOTP Internet Open Trading Protocol

IP Internet Protocol

ISO International Organisation for Standardization

JCE Java Cryptography Extension

MD Message Digest

NIST National Institute of Standards and Technology

NSA National Security Agency

OFX Open Financial Exchange

PCA Policy Certification Authorities

PCMCIA Personal Computer Memory Card International Association

PI Processing Instruction

PIN Personal Identification Number

PKI Public Key Infrastructure

RegTP Regulierungsbehörde für Telekommunikation und Post

RFC Request for Comments

RIPEMD RACE Integrity Primitives Evaluation Message Digest

RSA Kryptoverfahren von Rivest, Shamir und Adleman

SAX Simple API for XML

SGML Standard Generalized Markup Language

SHA Secure Hash Algorithm

SigG Signaturgesetz

SigV Signaturverordnung

sintaX Secure Internet Auction with XML

SSL Secure Socket Layer

TCP Transmission Control Protocol

TLCA Top Level Certification Authority

UMTS Universal Mobile Telecommunication System

USB Universal Serial Bus

UTF Unicode Transformation Format

URI Uniform Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

WWW World Wide Web

XHTML eXtensible Hypertext Markup Language

XML eXtensible Markup Language

XPath XML Path Language

XSL eXtensible Style Language

XSLT eXtensible Style Language Transformation

Kapitel 1

Einleitung

E - Commerce, der Handel über das Internet, hat sich in letzter Zeit sehr stark verbreitet. Ein Ende dieses Wachstums ist nicht in Sicht. Immer mehr Unternehmen bieten ihren Kunden die Möglichkeit, nicht nur auf traditionellen Wegen wie Post oder Telefon, sondern auch online Bestellungen bei ihnen aufzugeben, und immer mehr Kunden nutzen diese Möglichkeit. Während noch vor einiger Zeit Unternehmen den Online - Handel in erster Linie aus Marketing - und Image - Gründen anboten, um zu demonstrieren, dass sie ein modernes und fortschrittliches Unternehmen sind, erzielen heute bereits große Konzerne Umsätze im Milliarden - Bereich allein im Online - Geschäft.

Jedoch nutzen heutige E - Commerce - Lösungen nur ansatzweise das Potential, das das Medium Internet bietet. Technologien wie Authentifizierung durch digitale Signaturen oder komplexere, rechtssichere Geschäftsprotokolle werden kaum verwendet, obwohl sie zur Verfügung stehen und als ausgereift gelten. *E - Commerce* bedeutet heutzutage in der Regel klassisches katalogbasiertes Einkaufen, das heißt, der Kunde wählt Produkte aus einem Katalog zu festgelegten Preisen aus, und bestellt diese. Dabei sind auch beim elektronischen Handel weitere Handelswege und - methoden, wie etwa Verhandlungen, Auktionen und Börsen denkbar. Besondere Bedeutung kommt hierbei insbesondere Auktionen zu. Neben dem katalogbasierten Einkaufen stellen Auktionen auch heute schon das am weitesten verbreitete Geschäftsmodell im Internet dar. Nicht nur im *Business* - Bereich gewinnen Auktionen an Bedeutung, auch im *Consumer* - Bereich erfreuen sich Auktionen großer Beliebtheit, wie der Erfolg von Auktionsplattformen wie eBay eindrucksvoll demonstriert. Allerdings zeigt sich, dass bestehende Online - Auktionen im Bereich Sicherheit noch starke Mängel aufweisen, und von daher keine ausreichende Rechtssicherheit bieten.

Ein weiterer, in erster Linie aus technologischer Sicht wichtiger Aspekt im E - Commerce ist das zugrunde liegende Datenformat. HTML als das wichtigste Datenformat im WWW (*World Wide Web*) mag zwar für einfache Shopping - Lösungen geeignet sein, für die Darstellung komplexer Verhandlungen ist es jedoch ungeeignet. Daneben existieren zahlreiche nicht miteinander kompatible Datenformate, die zudem auf die Modellierung spezieller Geschäftsvorfälle zugeschnitten sind. Als universelles und leistungsfähiges Datenformat bietet sich die *eXtensible Markup Language XML* an. XML ist ein offener Standard, ist einfach aufgebaut und zudem beliebig erweiterbar. Sie ist aufgrund ihrer Vorteile ausgezeichnet für die Darstellung von Geschäftsprozessen geeignet.

An dieser Stelle stellt sich jedoch trotzdem die Frage, wieso sich noch eine weitere Diplomarbeit mit den Themen XML und E - Commerce beschäftigt. Schließlich wurde zu diesen Themen schon sehr viel geschrieben, und es wurden bereits zahlreiche Beiträge veröffentlicht, die zeigen, wie sich XML im E - Commerce einsetzen lässt. Der Grund hierfür ist, dass das Internet ein sehr schnellebiges Medium ist, das ständig neue Entwicklungen hervorbringt. Den Hintergrund für diese Arbeit stellen die gegenwärtigen Aktivitäten des W3C (*World Wide Web Consortium*) im Bereich „XML und digitale Signaturen“ dar. Eine Arbeitsgruppe des W3C beschäftigt sich mit der Entwicklung einer Spezifikation zur Darstellung digitaler Signaturen in XML. Dieser Entwicklung wird eine große Bedeutung beigemessen, da gerade bei elektronischen Geschäftsmodellen das Thema Sicherheit eine entscheidende Rolle spielt. Anforderungen wie die Integrität von Daten oder die Nichtabstreitbarkeit von Verträgen müssen sich realisieren lassen. Hierzu könnten die Ergebnisse dieser Arbeitsgruppe einen entscheidenden Beitrag liefern, gerade wenn es darum geht, XML als wichtigstes Datenformat im E - Business zu etablieren.

1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Implementierung einer Demonstrations - Applikation, durch die ein realistisches Handelsszenario simuliert wird. Bei dem Szenario geht es um die Herstellung eines nichtabstreitbaren gegenseitigen Übereinkommens, wie es mit dem an der GMD entwickelten BAKO - Protokoll erzielt wird. Der Nachrichtenaustausch in dem Szenario soll durch signierte XML - Dokumente gemäß W3C - Spezifikation realisiert werden. Hierzu ist eine geeignete API zur Generierung von XML - Signaturen zu entwickeln. Bei der Implementierung soll so weit wie möglich auf bereits existierende Standardanwendungen zurückgegriffen werden, um das Integrationspotential von XML zu demonstrieren. Ein wichtiger Teil

der Arbeit stellt die Entwicklung geeigneter XML - Stylesheets zur allgemeinen, intuitiven Darstellung von XML - Signaturen für den Verifizierer dar, die auch einer Bewertung nach dem deutschen Signaturgesetz (SigG / SigV) standhalten. Vor diesem Hintergrund entstand im Rahmen dieser Arbeit *sintaX* (*Secure Internet Auction with XML*), ein Prototyp einer Online - Auktion, die zeigt, wie sich durch den Einsatz von XML - Signaturen eine Online - Auktion realisieren lässt, die auch hohen Sicherheitsanforderungen gerecht wird. Die Implementierung der für *sintaX* benötigten Komponenten, wie ein Signaturwerkzeug und einer API für XML - Signaturen, aber auch entsprechender Stylesheets, stellt in ihrem Umfang den wesentlichen Teil dieser Arbeit dar.

1.2 Projekthintergrund

Die vorliegende Arbeit entstand im Rahmen des EU geförderten Projekts *eBroker* der beiden GMD - Institute SIT und IPSI in Zusammenarbeit mit kommerziellen Partnern. Es handelt sich dabei um die Realisierung einer Plattform für vermittelten E - Commerce, die alle wesentlichen Handelsschritte unterstützt. Dazu gehören die Spezifikation und Abfrage von Produktangebot und - nachfrage, von individuellen Handelspräferenzen, sowie die Bereitstellung marktspezifischer Handelsmethoden wie Auktion, Börse, Verhandlung und Festpreisangebot. Die Vermittlerdienste sollen sich den marktspezifischen Gegebenheiten anpassen. Abhängig von Haltbarkeit und Verfügbarkeit der Waren, dem Verhältnis von Angebot und Nachfrage, sowie dem durchschnittlichen Transaktionswert und - volumen empfehlen sich unterschiedliche Handelsmethoden. Von zentraler Bedeutung ist die einheitliche Modellierung der gehandelten Produkte, der formalen Handelsbedingungen und Geschäftsprozesse. Hierfür bietet sich XML als Infrastruktur an. *[EBrok]*

1.3 Gliederung der Arbeit

In den ersten Kapiteln dieser Arbeit werden die für das Verständnis der behandelten Themen notwendigen Grundlagen erläutert. Kapitel 2 bietet eine Einführung in XML. Es werden XML selbst, sowie einige in dieser Arbeit verwendete Entwicklungen und Begriffe aus dem Umfeld von XML beschrieben. In Kapitel 3 werden die für das Verständnis von digitalen Signaturen nötigen Grundlagen dargelegt. Dazu gehören symmetrische und asymmetrische Kryptografie sowie Hashalgorithmen.

In Kapitel 4 wird ausführlich die W3C - Spezifikation zur Darstellung digitaler Signaturen in XML vorgestellt. Die Ausführungen basieren auf der *Candidate Recommendation* vom 31. Oktober 2000. In diesem Kapitel wird auch auf die Kanonisierung von XML - Dokumenten eingegangen.

In Kapitel 5 wird das an der GMD entwickelte BAKO - Protokoll zum Erreichen gegenseitiger Nichtabstreitbarkeit beschrieben. Im Anschluss wird gezeigt, wie sich das Protokoll mit Hilfe von XML - Signaturen modellieren lässt.

Kapitel 6 beschäftigt sich mit dem Problem der Darstellung digitaler Signaturen, bekannt unter der Bezeichnung *Präsentationsproblem*. Desweiteren wird eine mögliche Lösung dieses Problems zur Darstellung von XML - Signaturen präsentiert und es erfolgt eine Bewertung gemäß Signaturgesetz.

Kapitel 7 gibt einen Überblick über elektronische Auktionen. Es werden die Vorteile elektronischer Auktionen diskutiert und es werden gängige Auktionstypen vorgestellt.

Kapitel 8 beschäftigt sich mit der Implementierung von *sintaX*. Es werden die Anforderungen an eine sichere Online - Auktion beschrieben und es wird gezeigt, wie sich diese Anforderungen umsetzen lassen. Im Anschluss wird die Architektur von *sintaX* vorgestellt.

Zum Abschluss dieser Arbeit wird in Kapitel 9 ein kurzes Fazit gezogen und es findet ein Ausblick auf zukünftige Erweiterungen und Aktivitäten statt.

Kapitel 2

Einführung in XML

Die *eXtensible Markup Language* XML ist ein in letzter Zeit häufig verwendetes Schlagwort im IT - Bereich. Gerade im E - Business, der elektronischen Geschäftsabwicklung, spielt XML eine immer größere Rolle. Bislang ist die *HyperText Markup Language* HTML das Datenformat Nummer eins im Internet. Mit Hilfe von HTML werden die Inhalte des *World Wide Web* (WWW) dargestellt. Dabei hat HTML einen entscheidenden Nachteil. HTML beschreibt, wie ein Browser die Daten darstellen soll, wie Texte, Bilder und sonstige Elemente auf einer Seite plaziert werden sollen, aber mit HTML lassen sich nicht die Daten selbst beschreiben.

> *The problem with „What You See Is What You Get" is that what you see is all you've got.* (Brian Kernighan)

So gehen beispielsweise zwangsläufig Informationen verloren, wenn Daten aus einer Datenbank in HTML konvertiert werden sollen, um sie im Web darzustellen. Solche Daten besitzen im allgemeinen eine komplexe Struktur, die man verliert, wenn man die Daten in das relativ starre und unflexible HTML - Format presst. Auf der Client - Seite ist diese Struktur der Daten dadurch praktisch nicht mehr vorhanden. Eine Weiterverarbeitung der Daten, die über das Betrachten hinausgeht, ist somit nur bedingt möglich. Einem Informationsaustausch zwischen Client und Server in beide Richtungen und einer Weiterverarbeitung der Daten auf beiden Seiten steht HTML im Wege. Für zukünftige E - Business - Anwendungen ist jedoch gerade dies zwingend notwendig, um beispielsweise die Integration von Backend - Servern oder Systemen wie SAP zu gewährleisten.

Als Lösung für dieses Problem bietet sich XML als universelles Datenformat an. Mit XML lässt sich der Inhalt eines Dokuments beschreiben. Daten können mit XML beliebig fein strukturiert werden, so dass sich auch komplexe Datenstrukturen darstellen lassen. Die Vorteile von XML sind

- *Plattformunabhängigkeit*. XML ist nicht an spezielle Anwendungen oder Betriebssysteme gekoppelt.

- *Offener Standard*. XML wurde vom World Wide Web Consortium W3C entwickelt. Die Spezifikation ist frei verfügbar. Somit ist XML herstellerunabhängig.

- *Einfache Struktur*. XML wurde bewusst so entwickelt, dass es mit wenigen einfachen Regeln auskommt.

- *Erweiterbarkeit*. In XML lassen sich leicht neue Elemente für eigene Anwendungen definieren. Daneben existieren bereits viele standardisierte Erweiterungen, beispielsweise zur Darstellung oder der Verknüpfung von XML - Dokumenten mit Hyperlinks, die selbst auch in XML formuliert sind.

- *Für Menschen lesbare Form*. Zum einen werden XML - Dokumente in normalem Textformat gespeichert[1], so dass keine besonderen Anwendungen zum Anzeigen der Dokumente notwendig sind. Zum anderen sind XML - Dokumente durch die einfache Struktur und durch die Verwendung aussagekräftiger Namen gut lesbar.

Dieses Kapitel soll eine kurze Einführung in XML geben[2] und einige Technologien aus dem Umfeld von XML erläutern, die in den folgenden Kapiteln häufig verwendet werden. *[Mac97][BPS98]*

2.1 XML

Die *eXtensible Markup Language* XML (*[BPS98]*) wurde von einer Arbeitsgruppe des *World Wide Web Consortiums* (W3C) entwickelt. Es handelt sich bei XML um eine Untermenge der *Standard Generalized Markup Language* SGML (ISO 8879).

SGML stellt Vorschriften bereit, um Auszeichnungssprachen, sogenannte SGML - Anwendungen, zu definieren. Die wohl bekannteste SGML - Anwendung dürfte HTML sein. Dies stellt einen wesentlichen Unterschied zwischen XML und HTML dar: XML ist eine Teilmenge von SGML, HTML ist eine Anwendung von SGML. Der Nachteil von SGML ist die extrem komplexe und nur schwer durchschaubare Syntax.

[1] XML verwendet den Unicode-Zeichensatz

[2] Eine ausführliche Einführung in XML bietet z.B. [Har99]

Vor diesem Hintergrund entstand XML. In die XML - Spezifikation wurden nur die wichtigsten und am häufigsten verwendeten Merkmale von SGML übernommen. Es entstand dadurch eine leistungsfähige Auszeichnungssprache, die nur auf wenigen einfachen Regeln beruht. Ein Vergleich der beiden Spezifikationen verrät, wieviele Eigenschaften von SGML bei der Definition von XML über Bord geworfen wurden. Die SGML - Spezifikation füllt über 500 Seiten und benötigt damit mehr als zehnmal soviel Platz wie die XML - Spezifikation.

Ein XML - Dokument besteht im wesentlichen aus einer Menge von Elementen, sogenannter *Tags*, die die Daten auszeichnen. Diese Elemente sind nicht festgelegt, sondern frei definierbar. XML legt nur die generelle Syntax der Elemente fest. Damit lässt sich ein Dokument beliebig fein strukturieren. Die Elemente sollten nach Möglichkeit so gewählt werden, dass sie die Daten, die sie umschließen, benennen.

Eine endliche Menge festgelegter Elemente zusammen mit einer Beschreibung der Syntax der Dokumente, die mit diesen Elementen gebildet werden sollen, stellen eine XML - Anwendung dar. Ein Beispiel für eine XML - Anwendung stellt XHTML (*[XHTML]*) dar. Dabei handelt es sich um eine leichte Umformulierung von HTML, so dass die XML - Syntax erfüllt ist. XHTML besteht aus einer Menge genau spezifizierter Tags und auch ihre Verwendung ist genau bestimmt. Beispielsweise muss ein XHTML - Dokument aus einem *HTML* - Element bestehen, in dem sich ein *HEAD* - und ein *BODY* - Element befinden, aber eben nicht andersherum.

Ein einfaches XML - Dokument besteht aus der XML - Deklaration, gefolgt von beliebig vielen Elementen, die ineinander verschachtelt werden dürfen.

Beispiel 1:

```
[ 1] <?xml version="1.0">
[ 2]   <Element>
[ 3]     Inhalt
[ 4]   </Element>
```

Beispiel 2:

```
[ 1] <?xml version="1.0">
[ 2]   <Katalog>
[ 3]     <Produkt>
[ 4]       <Bezeichnung>
[ 5]         Java In A Nutshell
```

```
[ 6]       </Bezeichung>
[ 7]       <Preis>
[ 8]         60.00
[ 9]       </Preis>
[10]     </Produkt>
[11]     <Produkt>
[12]       <Bezeichnung>
[13]         Java Servlets
[14]       </Bezeichung>
[15]       <Preis>
[16]         80.00
[17]       </Preis>
[18]     </Produkt>
[19]   </Katalog>
```

Elemente können zudem Attribute enthalten, mit denen sich ihre Funktion näher spezifizieren lässt, zum Beispiel

```
<Produkt Id="4711">...</Produkt>
```

Alle Elemente eines XML - Dokuments müssen korrekt geöffnet und geschlossen werden, und sie müssen sich als Baumstruktur darstellen lassen. Ein XML - Dokument, das diesen Anforderungen genügt, bezeichnet man als „wohlgeformt“ (*well-formed*). Somit stellt beispielsweise das folgende Codefragment keinen wohlgeformten XML - Code dar, da es sich nicht in einer Baumstruktur darstellen lässt.

```
<a><b></a></b>
```

2.2 DTD

Wie schon erwähnt, besteht eine XML - Anwendung nicht nur aus einer festgelegten Menge von Elementen. Auch die Verwendung der Elemente, das heißt die Syntax der mit ihnen gebildeten Dokumente, muss spezifiziert werden. Dies geschieht mit Hilfe einer *Document Type Definition* (DTD). Eine DTD stellt die Grammatik einer XML - Anwendung dar. Eine DTD besteht im wesentlichen aus einer Menge von Regeln. Diese Regeln definieren die Namen der Elemente und spezifizieren die Struktur eines Dokuments. Sie machen jedoch keine Aussage über die Bedeutung der Elemente. Eine Regel könnte beispielsweise besagen, dass ein *Produkt* - Element aus Beispiel 2 genau ein *Bezeichnung* - und ein *Preis* - Element enthalten muss. Ein XML - Dokument, das den Regeln der DTD genügt, heißt „gültig“ (*valid*).

Beispiel 3:

```
[ 1] <?xml version="1.0">
[ 2] <!DOCTYPE Katalog [
[ 3]   <!ELEMENT Katalog(Produkt+)>
         <!--Katalog enthält mindestens 1 Produktelement-->
[ 4]   <!ELEMENT Produkt(Bezeichung, Preis)>
         <!--Produkt enthält Bezeichnung und Preis-->
[ 5]   <!ELEMENT Bezeichnung(#PCDATA)>
         <!-- Bezeichnung enthält beliebigen Text-->
[ 6]   <!ELEMENT Preis(#PCDATA)>
         <!-- Preis enthält beliebigen Text-->
[ 7]   <!ATTLIST Produkt Id ID #REQUIRED>
         <!-- Produkt muss Id-Attribut enthalten-->
[ 8]   <!ATTLIST Preis Waehrung CDATA #REQUIRED>
[ 9] ]>
[10] <Katalog>
[11]   <Produkt>
[12]     ...
[13]   </Produkt>
[14]   ...
[15] </Katalog>
```

Die Möglichkeiten von DTDs zur Definition von XML - Anwendungen sind jedoch begrenzt. Insbesondere kennen DTDs keine Datentypen, es lässt sich beispielsweise nicht ausdrücken, dass ein Element nur Zahlen enthalten darf. Aus diesem Grund könnten DTDs in Zukunft von XML - Schemata (*[Fal00]*) abgelöst werden. XML - Schemata dienen ebenfalls der Definition von XML - Anwendungen, sie gehen jedoch weit über die Möglichkeiten von DTDs hinaus. So unterstützen XML - Schemata etwa unterschiedliche Datentypen.

2.3 Namespaces

Es ist durchaus möglich, mehrere XML - Anwendungen miteinander zu kombinieren. Das bedeutet, es lassen sich XML - Dokumente bilden, deren Elemente aus verschiedenen XML - Anwendungen stammen. Dabei kann es jedoch zu Konflikten kommen, wenn zwei Anwendungen gleichlautende Elemente enthalten. Es ist nun nicht mehr feststellbar, zu welcher das Element gehört und in welchem Kontext es zu verwenden ist.

Abhilfe für dieses Problem schaffen sogenannte Namensräume (*Namespaces*, *[BHL99]*). Ein Namensraum wird durch ein „xmlns:prefix“ - Attribut definiert. Der Wert dieses Attributes ist eine eindeutige URI (*Uniform Resource Identifier*, *[BL96]*), die die XML - Anwendung benennt. Jedem Element, das zu dieser Anwendung gehört, wird das Präfix vorangestellt.

Beispiel 4:

```
[ 1] <haendler:Katalog xmlns:haendler="http://www.haendler.de">
[ 2]   <haendler:Produkt Id="4711">
[ 3]     ...
[ 4]   </haendler:Produkt>
[ 5]   ...
[ 6] </haendler:Katalog>
```

2.4 XSLT und XSL

XML - Dokumente dienen dazu, ihren Inhalt zu beschreiben. Sie enthalten aber im allgemeinen keine Informationen darüber, wie sie dargestellt werden sollen. Wie und mit welchem Ausgabemedium sie präsentiert werden sollen, ist völlig offen. Gerade das ist einer der Vorteile von XML, die klare Trennung von Layout und logischer Struktur eines Dokumentes.

Um XML - Dokumente darzustellen, kann man sich der XSLT - Sprache (*eXtensible StyleLanguage - Transformation, [Cla99]*) bedienen. XSLT selbst ist auch eine XML - Anwendung, die dazu dient, XML - Dokumente in andere XML - Dokumente zu transformieren. Das klassische Beispiel hierfür ist die Überführung eines XML - Dokuments in ein XHTML - Dokument, das dann im Browser dargestellt werden kann.

Für die Transformation wird ein XSLT - Prozessor, beispielsweise *Xalan* von Apache[3] oder Microsoft Internet Explorer 5.x[4] benötigt. Der XSLT - Prozessor bekommt als Eingabe das XML - Dokument und das XSLT - Stylesheet. Als Ausgabe liefert er das Ergebnis der Transformation. Ein XSLT - Stylesheet besteht im wesentlichen aus einer Menge von Transformationsregeln, sogenannten *Templates*. Für jedes Element im Ausgangsdokument gibt es ein Template, das beschreibt, in was das Element transformiert werden soll. Der XSLT - Prozessor traversiert rekursiv das Ausgangsdokument. Für jedes Element, auf das er stößt, schaut er, ob es ein entsprechendes Template im Stylesheet gibt, und wendet es an. Auf diese Weise wird das Ergebnis der Transformation gebildet. Abbildung 2.1 zeigt den schematischen Ablauf einer Transformation eines XML - Dokuments in XHTML. Die folgenden beiden Beispiele sollen die Arbeitsweise eines XSLT - Prozessors verdeutlichen. Die Beispiele zeigen zwei Templates, so wie sie in einem beliebigen XSLT - Dokument stehen können.

[3] http://xml.apache.org/xalan
[4] http://www.microsoft.com

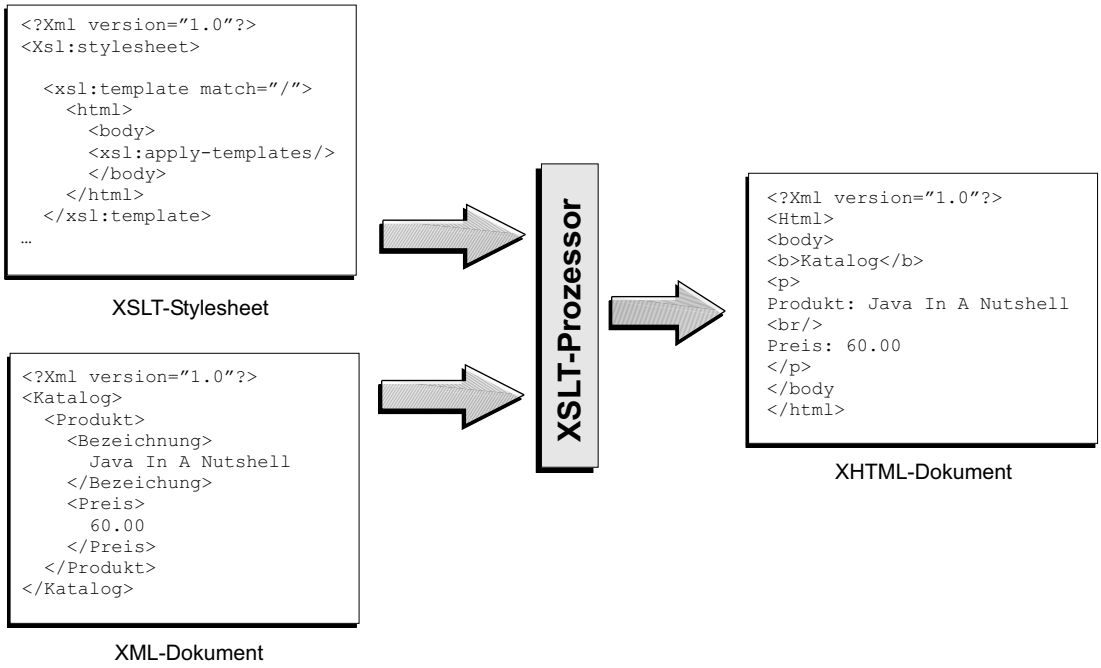

Abbildung 2.1: XSLT - Transformation

Beispiel 5:

```
[ 1] <xsl:template match="/">
[ 2]   <html>
[ 3]     <body>
[ 4]     <xsl:apply-templates/>
[ 5]     </body>
[ 6]   </html>
[ 7] </xsl:template>
```

Beispiel 6:

```
[ 1] <xsl:template match="Produkt">
[ 2]   <p>
[ 3]     Produkt: <xsl:value-of select="Bezeichnung"/>
[ 4]     <br/>
[ 5]     Preis: <xsl:value-of select="Preis"/>
[ 6]   </p>
[ 7] </xsl:template>
```

Das Element *template* enthält in dem Attribut *match* den Namen des Elements, auf das das Template angewandt werden soll. Das Template aus Beispiel 5 wird auf das Wurzelelement ("/") angewendet. Es ersetzt das Wurzelelement durch ein *html* - Element, in dem ein *body* - Element enthalten ist. Das Element *apply-templates* bewirkt, dass an dieser Stelle das

XSLT - Stylesheet rekursiv auf die Nachfolger des Wurzelelements angewendet wird, und damit das Ergebnis dieser Transformationen an dieser Stelle eingefügt wird. Das Template aus Beispiel 6 ersetzt ein *Produkt* - Element durch den angegebenen Text. Das Element *value-of* dient als Platzhalter. An der Stelle dieses Elements wird der Wert des in dem Attribut *select* enthaltenen Elements eingefügt. Die Angaben in diesem Attribut können, wie im Beispiel, relativ oder aber auch absolut, also bezogen auf das Wurzelelement, gesetzt werden. Die Werte *Bezeichnung* und *Preis* in dem Beispiel bedeuten, dass die Elemente *Bezeichnung* und *Preis* jeweils direkte Nachfolger des Elements *Produkt* sein müssen.

Neben der Transformationssprache XSLT wurde vom W3C auch die Formatierungssprache XSL (eXtensible Style Language, *[ABC00]*) spezifiziert. XSL - Stylesheets, sogenannte *XSL formatting objects*, dienen der Darstellung von XML - Dokumenten. Die XSL - Spezifikation ist sehr umfangreich, die Mächtigkeit von XSL überschreitet bei weitem das, was man von HTML kennt. Im Gegensatz zu HTML wurde XSL nicht nur zur Darstellung im Web entwickelt. XSL ist wesentlich allgemeiner definiert, so dass es zu Präsentationen auf beliebigen Ausgabemedien dient.

In der Regel verwendet man XSLT - Stylesheets zur Transformation von XML - Dokumenten in *XSL formatting objects*. Bislang existieren jedoch so gut wie keine Implementationen von XSL, was wohl zum einen daran liegt, dass sich XSL noch in der Entwicklung befindet, und zum anderen an dem extremen Umfang der Spezifikation. Insbesondere gibt es noch keine vollständigen XSL - Implementationen. Eine der wenigen Implementationen von XSL stellt FOP[5] dar. FOP ist ein Java - Programm, das *XSL formatting objects* in PDF - Dokumente konvertiert.

2.5 XPath

Die *XML Path Language* (XPath, *[CR99]*) definiert eine Syntax zum Adressieren von Teilen eines XML - Dokuments. XPath selbst stellt keine XML - Anwendung dar. Stattdessen wurde eine sehr kompakte Notation gewählt, die sich auch innerhalb von URLs verwenden lässt. Mit Hilfe von XPath kann man relativ einfach Teile eines XML - Dokuments herausfiltern oder entfernen, und man kann das Vorhandensein bestimmter Elemente in einem Dokument testen. XPath findet unter anderem in XSLT - Stylesheets Verwendung. Dort wird XPath innerhalb der Templates zum Adressieren der zu transformierenden Elemente verwendet. *[Har99][Mac97]*

[5] http://www.jtaubner.com/fop/

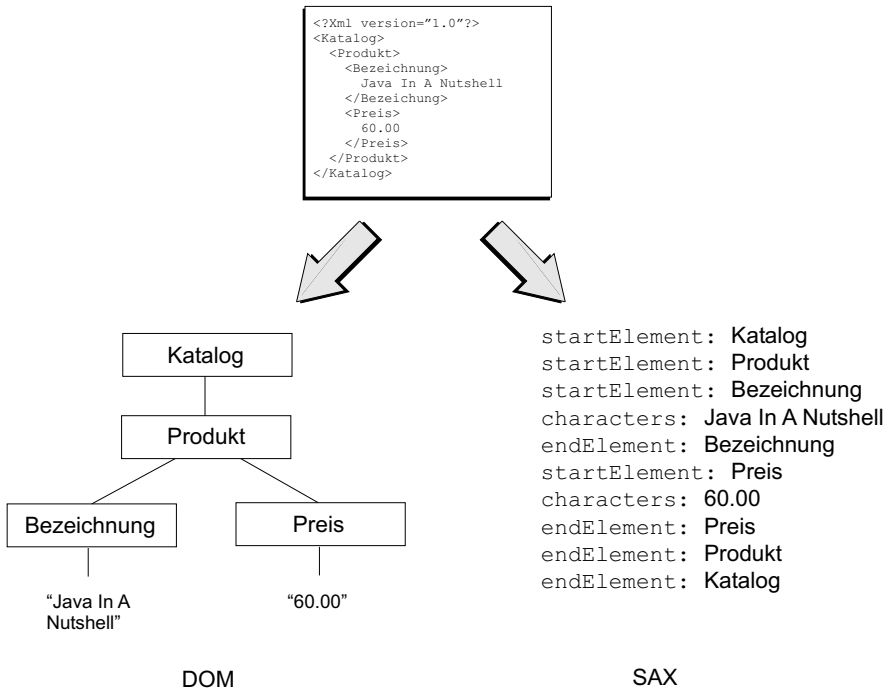

Abbildung 2.2: Vergleich von DOM und SAX APIs

2.6 DOM und SAX

Document Object Model (DOM) und *Simple API for XML* (SAX) beschreiben zwei standardisierte APIs, die von XML - Parsern unterstützt werden.

DOM stellt ein XML - Dokument in einer Baumstruktur dar. Die Elemente, Attribute, der Text usw. repräsentieren die Knoten des Baums. Die DOM - API stellt Methoden zur Verfügung, mit denen sich diese Knoten beliebig manipulieren lassen. DOM - basierte XML - Parser halten das komplette XML - Dokument im Speicher, weswegen DOM für extrem große Dokumente weniger geeignet ist. DOM bietet sich an, wenn man die Struktur eines XML - Dokuments verändern möchte oder wenn mehrere Applikationen auf ein Dokument im Speicher zugreifen.

Bei SAX handelt es sich um eine *event* - basierte API. SAX erzeugt keine Datenstruktur. Stattdessen liest SAX ein XML - Dokument ein, und generiert Events, beispielsweise für jeden Start - oder End - Tag. Da SAX keine Datenstruktur erstellt, ist SAX wesentlich effizienter als DOM. Dafür ist SAX nicht geeignet, die Struktur eines Dokuments zu manipulieren. SAX operiert, im Gegensatz zu DOM, nicht rekursiv auf dem Dokument,

das heißt, alle Operationen finden während des Parsens statt. Die Verwendung von SAX bietet sich an, wenn sehr große Dokumente bearbeitet werden, und wenn man in erster Linie Informationen aus dem Dokument herausfiltern möchte. Abbildung 2.2 zeigt die Baumstruktur eines XML - Dokuments, die eine DOM - API erzeugen würde, und im Vergleich dazu die Events, die SAX für dasselbe Dokument generieren würde. *[MTU99]*

Kapitel 3

Grundlagen der Kryptographie

Ein wichtiges Thema dieser Arbeit sind digitale Signaturen. Zum Verständnis sind einige Grundlagen der Kryptographie erforderlich, die in diesem Kapitel dargelegt werden[1]. Mit Hilfe kryptographischer Verfahren erreicht man die folgenden Ziele.

- *Vertraulichkeit*. Der Inhalt von Daten soll nur autorisierten Personen zugänglich sein.

- *Datenintegrität*. Daten sollen nicht unerlaubt manipuliert werden können. Zumindest soll sich aber eine eventuelle Manipulation aufdecken lassen.

- *Authentifizierung*. Parteien, die miteinander kommunizieren, sollen sich gegenseitig identifizieren. Das gleiche gilt auch für Daten, die ausgetauscht werden. Auch sie sollen sich anhand vom Ursprung, vom Sendezeitpunkt usw. eindeutig identifizieren lassen.

- *Nichtabstreitbarkeit*. Es soll die Möglichkeit bestehen, dass eine Partei verbindliche Aktionen durchführen kann. Sie soll später nicht abstreiten können, diese Aktion durchgeführt zu haben.

Vertraulichkeit lässt sich durch die Verschlüsselung der Daten erreichen. Integrität, Authentifikation und Nichtabstreitbarkeit erhält man durch die Verwendung digitaler Signaturen in Verbindung mit authentischen, öffentlichen Schlüsseln. Die Authentizität von Schlüsseln wird durch die Verwendung von Zertifikaten ermöglicht. Die genannten Anforderungen sind insbesondere in offenen, anonymen Medien, wie dem Internet wichtig. *[BS96]*

[1] Ausführliche Einblicke in die Kryptographie bieten z.B. [MOV97] und [BS96]

3.1 Kryptographische Verfahren

Kryptographische Verfahren lassen sich im wesentlichen in zwei Kategorien unterteilen, symmetrische und asymmetrische Verfahren.

Die ältesten bekannten Verschlüsselungsverfahren beruhen auf symmetrischer Verschlüsselung. Bei diesen Verfahren einigen sich beide Parteien, also Sender und Empfänger der verschlüsselten Nachricht, auf einen gemeinsamen Schlüssel. Der Sender benutzt diesen Schlüssel zur Verschlüsselung, der Empfänger entschlüsselt die Daten mit dem selben Schlüssel. Dies stellt zugleich die große Schwachstelle symmetrischer Kryptographie dar. Sender und Empfänger müssen sich zunächst über einen sicheren Kanal auf einen gemeinsamen Schlüssel einigen, erst dann können sie über ungesicherte Kanäle verschlüsselte Nachrichten austauschen (siehe Abbildung 3.2). Gelangt ein Angreifer in Besitz des Schlüssels, kann er sämtliche Nachrichten, die die beiden Parteien austauschen, entschlüsseln. Die Kenntnis über den verwendeten Algorithmus liefert einem Angreifer jedoch keinen Schlüssel, der Algorithmus muss also nicht geheim gehalten werden.

Klartextalphabet:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geheimtextalphabet:	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Klartext:	eine geheime botschaft.
Geheimtext:	HLQH JHKHMPH ERWVFKDIW.

Abbildung 3.1: Cäsar-Chiffre

Ein sehr einfaches und zugleich eines der ältesten Beispiele für symmetrische Verschlüsselung ist der sogenannte *Cäsar Chiffre*. Bei diesem Verfahren wird das Alphabet um eine bestimmt Stellenanzahl verschoben. Mit diesem dadurch entstehenden Geheimtextalphabet wird die verschlüsselte Nachricht gebildet. Jeder Buchstabe der Nachricht wird also durch einen weiter hinten im Alphabet liegenden Buchstaben substituiert. Der verwendete Schlüssel entspricht der Anzahl der Stellen, um die die Buchstaben verschoben werden. Abbildung 3.1 zeigt ein Beispiel eines Cäsar - Chiffres. In diesem Beispiel wird ein *a* durch ein *d* ersetzt, ein *b* durch ein *e* usw. Der Schlüssel ist damit die Zahl drei.

Einer der bekanntesten heute gebräuchlichen symmetrischen Verschlüsselungsalgorithmen ist DES (*Digital Encryption Standard*) aus dem Jahr

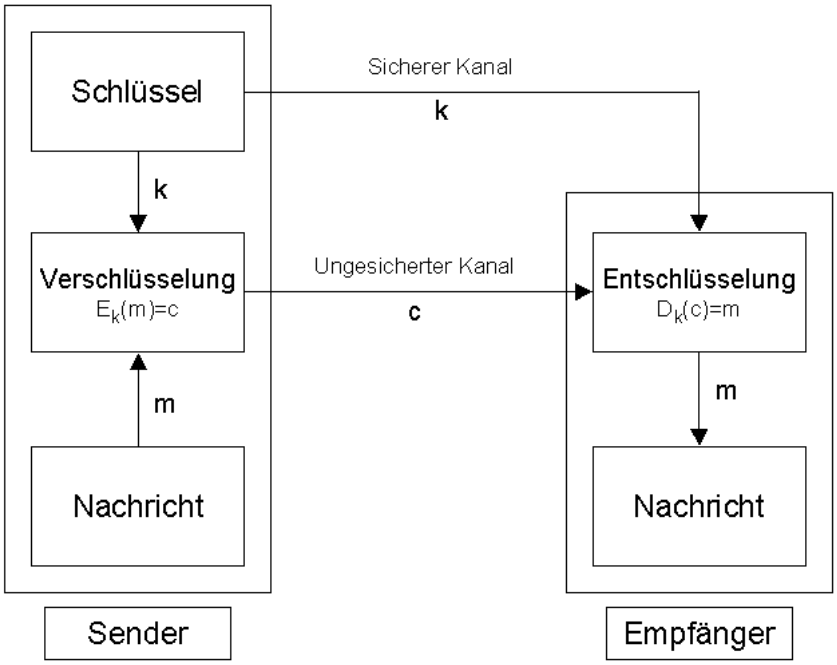

Abbildung 3.2: Symmetrische Verschlüsselung

1976. Durch seine Schlüssellänge von 56 Bit ist er auf Grund der heutigen Rechnerleistung nicht mehr ausreichend gegen *Brute Force* - Angriffe gesichert. Bei *Brute Force* - Attacken werden systematisch alle Möglichkeiten ausprobiert, bis man den richtigen Schlüssel gefunden hat. Unbestätigten Gerüchten zufolge hat die NSA (*National Security Agency*) bei der Entwicklung von DES Hintertüren einbauen lassen, durch die sie mit DES verschlüsselte Nachrichten entschlüsseln kann. Als Erweiterung wurde Triple-DES eingeführt, der den DES - Algorithmus dreifach anwendet und auch heute noch als sicher gilt. Anfang Oktober 2000 kürte das amerikanische *National Institute of Standards and Technology* (NIST)[2] einen neuen Algorithmus zum neuen Standard - Verschlüsselungsalgorithmus. Dieser Algorithmus, mit dem Namen *Rijndael*, soll 2001 unter der Bezeichnung AES (*Advanced Encryption Standard*) offiziell den DES Algorithmus ablösen. *Rijndael*, benannt nach seinen beiden belgischen Entwicklern, unterstützt Schlüssellängen von 128, 192 und 256 Bits.

Weitere Algorithmen sind unter anderem IDEA (*International Data Encryption Algorithm*) und CAST128 (benannt nach Carlisle und Stafford). Beide Algorithmen verwenden Schlüssel mit einer Länge von 128 Bit.

[2]http://www.nist.gov

Wie schon erwähnt, liegt das Problem symmetrischer Kryptoalgorithmen in der Schlüsselverteilung. Man muss, vereinfacht ausgedrückt, um ein Geheimnis zu übermitteln, vorher ein anderes Geheimnis senden, nämlich den vereinbarten Schlüssel. Zwar stellten Diffie, Hellman und Merkle 1976 ein auf Einwegfunktionen basierendes Verfahren vor, mit dem sich Schlüssel auch über ungesicherte Kanäle vereinbaren lassen, doch ist die Handhabung dieses Verfahrens recht umständlich. Ein weiterer Nachteil symmetrischer Kryptographie ist die Anzahl der Schlüssel. Um eine sichere Kommunikation zu gewährleisten, muss mit jedem Kommunikationspartner ein anderer Schlüssel vereinbart werden. Das bedeutet, wenn n Personen miteinander kommunizieren wollen, sind $\frac{n(n-1)}{2}$ Schlüssel erforderlich.

1976 stellten Diffie und Hellman jedoch ein völlig neues Paradigma der Kryptographie vor, das das Problem der Schlüsselverteilung mit einem Schlag lösen sollte: *Asymmetrische kryptographische Verfahren* (*[DH76]*). Während man bei symmetrischen Kryptoalgorithmen denselben Schlüssel zum Ver- und Entschlüsseln verwendet, kommen bei asymmetrischen Algorithmen Schlüsselpaare, bestehend aus einem geheimen Schlüssel (*Private Key*) und einem öffentlichen Schlüssel (*Public Key*) zum Einsatz. Den geheimen Schlüssel bewahrt man an einem sicheren Ort auf, den öffentlichen Schlüssel stellt man seinen Kommunikationspartnern zur Verfügung, er kann ohne weiteres über ungesicherte Kanäle übertragen werden. Möchte man eine Nachricht verschlüsseln, verwendet man dazu den öffentlichen Schlüssel des Empfängers. Mit diesem Schlüssel ist es jedoch nicht möglich, die Nachricht zu entschlüsseln, das kann nur der Empfänger mit seinem dazugehörigen privaten Schlüssel. Der private und der öffentliche Schlüssel verhalten sich also invers zueinander. Man braucht also weder Schlüssel über sichere Kanäle auszutauschen, noch braucht man für jeden Kommunikationspartner einen anderen Schlüssel zu vereinbaren, ein Schlüsselpaar genügt für die sichere Kommunikation mit beliebig vielen Partnern. Wenn also n Personen miteinander kommunizieren möchten, sind bei der Verwendung asymmetrischer Kryptographie nur n Schlüssel nötig. Das Prinzip der asymmetrischen Kryptographie ist in Abbildung 3.3 dargestellt.

Mathematisch gesehen basieren asymmetrische Kryptoalgorithmen auf einer speziellen Klasse von Einwegfunktionen, sogenannten *trapdoor one-way functions*. Einwegfunktionen sind Funktionen, bei denen es zwar einfach ist, den Funktionswert zu berechnen, die Umkehrung jedoch extrem schwer durchzuführen ist. Das bedeutet, zu einem gegebenen Wert x lässt

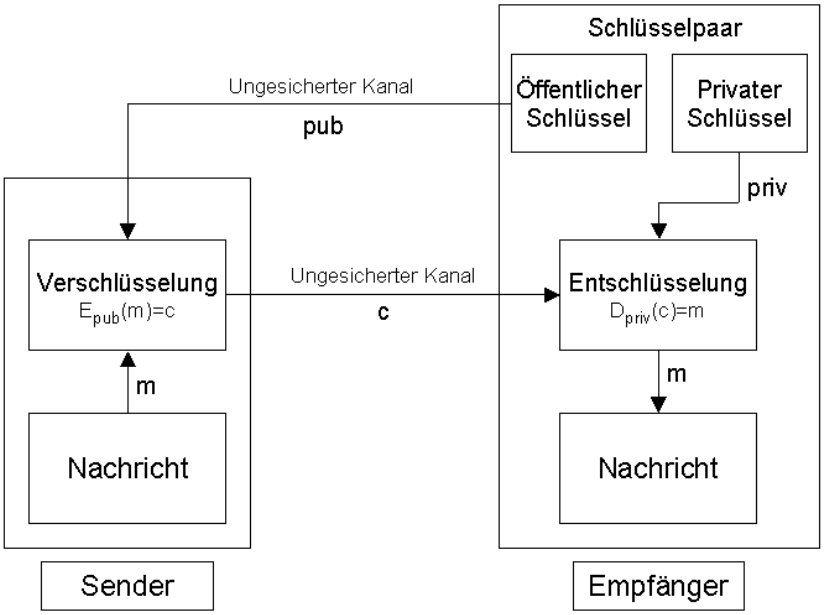

Abbildung 3.3: Asymmetrische Verschlüsselung

sich $f(x)$ leicht berechnen[3], zu einem gegebenen Wert $y = f(x)$ ist es jedoch nahezu unmöglich, den Wert x zu bestimmen. Der Begriff „nahezu unmöglich“ ist natürlich sehr vage. Gemeint ist damit, dass selbst mehrere miteinander vernetzte moderne Rechner sehr lange, beispielsweise einige tausend Jahre, benötigen würden, eine Lösung zu finden. Wichtig hierbei ist, dass bislang nicht bewiesen ist, dass es Einwegfunktionen tatsächlich gibt. Vielmehr existieren lediglich einige Funktionen, von denen vermutet wird, dass es sich um Einwegfunktionen handelt, weil bislang kein effizienter Algorithmus zur Lösung der Umkehrfunktion gefunden wurde. Man muss jedoch jeder Zeit damit rechnen, dass solch ein Algorithmus entdeckt wird.

Trapdoor one-way functions sind nun spezielle Einwegfunktionen, die mit einem zusätzlichen Stück Information die Umkehrung ermöglichen. Solche Funktionen eignen sich für asymmetrische kryptographische Verfahren. Jeder kann die Funktion benutzen, um eine Nachricht zu verschlüsseln. Die Umkehrung, also die Entschlüsselung, ist jedoch nur mit einem speziellen Geheimnis, dem privaten Schlüssel, möglich.

[3]genauer: $f(x)$ lässt sich in polynomieller Zeit berechnen.

Das wohl bekannteste asymmetrische Kryptoverfahren ist RSA, benannt nach seinen Entwicklern Rivest, Shamir und Adleman (*[RSA78]*). RSA gilt als sehr sicher, bislang sind keine erfolgreichen Angriffe auf RSA bekannt. Der Algorithmus beruht auf der Tatsache, dass es sehr schwierig ist, eine ausreichend große Zahl in ihre Primfaktoren zu zerlegen. Heute gelten Schlüssellängen von 512 Bit als sicher. Es wird jedoch bereits empfohlen, Schlüssel mit einer Länge von 1024 Bit zu verwenden, auch wenn oftmals aus Performancegründen darauf verzichtet wird.

Asymmetrische Kryptoalgorithmen sind im Vergleich zu symmetrischen Algorithmen sehr rechenintensiv. Beispielsweise ist RSA ca. tausendmal langsamer als DES. Außerdem sind asymmetrische Verschlüsselungsverfahren anfällig für „Klartext - Attacken“. Da der Schlüssel, der zum verschlüsseln benutzt wird, öffentlich ist, kann ein Angreifer alle möglichen Nachrichten systematisch verschlüsseln und mit der verschlüsselten Nachricht, deren Inhalt er erfahren möchte, vergleichen. Stimmen die beiden verschlüsselten Nachrichten überein, hat er die richtige gefunden. Deshalb verwendet man häufig Mischverfahren. Hierbei wird die eigentliche Nachricht mit Hilfe symmetrischer Kryptoverfahren verschlüsselt. Dazu wird ein einmaliger *session key* erzeugt. Dieser *session key* wird mit einem asymmetrischen Verfahren verschlüsselt und auf diese Weise gesichert ausgetauscht. *[BS96][Sin99][MOV97][SH00]*

3.2 Digitale Signaturen

Bei einigen asymmetrischen Kryptoalgorithmen, wie zum Beispiel RSA oder DSA (*Digital Signature Algorithm*), ist es möglich, auch den privaten Schlüssel zum Verschlüsseln und den öffentlichen Schlüssel zum Entschlüsseln zu verwenden. Solche Algorithmen sind geeignet, um digitale Signaturen zu erzeugen. Der Signierer verschlüsselt seine Nachrichten mit seinem privaten Schlüssel. Natürlich kann die Nachricht nun von jedem, der den dazugehörigen öffentlichen Schlüssel besitzt, entschlüsselt und gelesen werden. Aber darum geht es bei digitalen Signaturen auch nicht. Vielmehr ist damit sicher gestellt, dass die Nachricht auch tatsächlich von dem Signierer stammt, da sie nur mit seinem öffentlichen Schlüssel entschlüsselt werden kann. Wie schon erwähnt, sind asymmetrische Kryptoalgorithmen sehr rechenaufwendig. Deshalb verschlüsselt man statt der kompletten Nachricht einen Hashwert der Nachricht.

Hashfunktionen sind Funktionen, die eine Eingabe variabler Länge, zum Beispiel eine Nachricht, auf einen Wert fester Länge abbilden. Dieser Wert

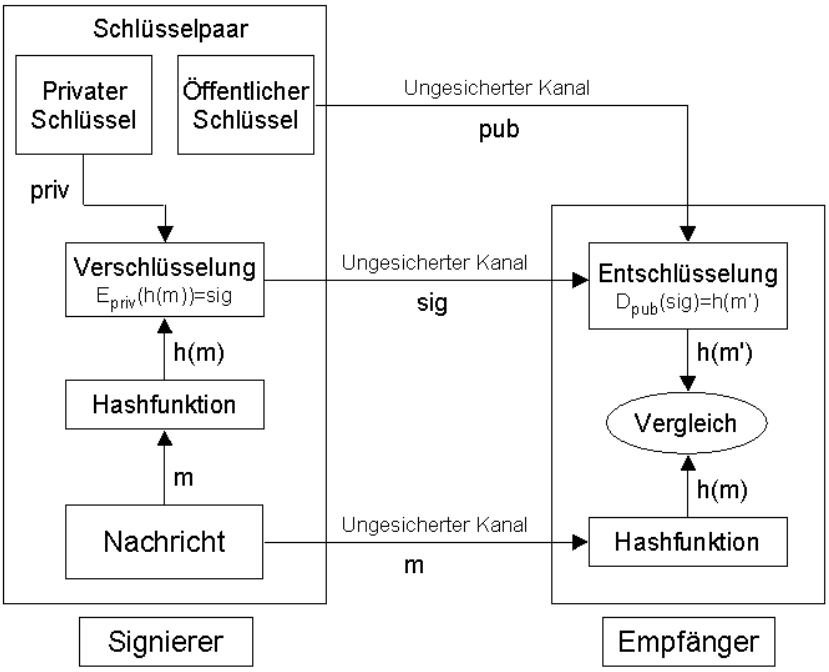

Abbildung 3.4: Digitale Signaturen

fester Länge, der *Hashwert*, dient als digitaler Fingerabdruck. Da die Menge der möglichen Eingaben größer ist, als die Menge der möglichen Hashwerte, können diese digitalen Fingerabdrücke nie eindeutig sein. Eine gute Hashfunktion zeichnet sich jedoch dadurch aus, dass es sehr schwer ist, solche *Kollisionen* zu erzeugen. Insbesondere bedeutet dies, dass kein effizienter Algorithmus bekannt sein darf, mit dessen Hilfe sich gezielt Kollisionen erzeugen lassen. Hashfunktionen mit dieser Eigenschaft nennt man *kollisionsfrei*. Bekannte Hashfunktionen sind u.a. MD5 (*Message Digest 5*), SHA-1 (*Secure Hash Algorithm 1*) und RIPEMD-160 (*RACE Integrity Primitives Evaluation Message Digest*). MD5 ist zwar sehr schnell, auf Grund seiner Hashwertlänge von 128 Bit jedoch nicht mehr als sicher anzusehen, weswegen von seiner Verwendung abgeraten wird. SHA-1 wurde vom NIST als Standard Hashalgorithmus ausgezeichnet. Es ist zwar langsamer als MD5, da er mit 160 Bit Hashwerten arbeitet, gilt er als ausreichend sicher. RIPEMD-160 verwendet ebenfalls 160 Bit Hashwerte und ist ebenfalls recht langsam.

Die Erzeugung digitaler Signaturen mit Hilfe von Hashfunktionen läuft nun wie folgt ab (siehe auch Abbildung 3.4). Der Signierer bildet einen digitalen Fingerabdruck, den Hashwert, seiner Nachricht. Diesen Has-

hwert verschlüsselt er mit seinem privaten Schlüssel. Als Ergebnis erhält er eine digitale Signatur seiner Nachricht. Er sendet nun sowohl die Signatur als auch die unverschlüsselte Nachricht an den Empfänger. Der Empfänger entschlüsselt den Hashwert mit dem öffentlichen Schlüssel des Signierers. Dann bildet er einen Hashwert über der Nachricht. Stimmen die beiden Hashwerte überein, wurde die Signatur erfolgreich *validiert*. *[BS96][MOV97][GKW97]*

3.3 Zertifikate

Bei digitalen Signaturen, die auf asymmetrischen kryptographischen Verfahren basieren, werden die Signaturen mit dem öffentlichen Schlüssel geprüft. Der öffentliche Schlüssel einer Person ist frei zugänglich, er kann beispielsweise von einer Internet - Seite heruntergeladen werden. Dabei ergibt sich jedoch ein Problem. Wie kann sicher gestellt werden, dass der Schlüssel auch tatsächlich der angegebenen Person gehört. Es könnte schließlich auch eine andere Person unter falschem Namen einen Schlüssel verbreiten. Es fehlt die Möglichkeit, Personen zu identifizieren. Abhilfe schaffen sogenannte *Zertifikate*. Zertifikate stellen die Verbindung zwischen öffentlichen Schlüsseln und Identitäten, also beispielsweise juristischen Personen, her. Solche Zertifikate können, ähnlich einem Personalausweis, von einer zentralen, vertrauenswürdigen Instanz ausgestellt werden, oder von einer beliebigen Person, die den Inhaber des öffentlichen Schlüssels kennt (dezentraler Ansatz). Der Aussteller eines Zertifikats signiert das Zertifikat und garantiert damit seine Echtheit. Entscheidend ist hierbei das Vertrauen, das dem Aussteller eines Zertifikats entgegengebracht wird. Es gibt im wesentlichen zwei unterschiedliche Modelle, sogenannte *Public Key Infrastrukturen* (PKI), die Vertrauen in die Aussteller von Zertifikaten schaffen sollen.

3.3.1 Hierarchische PKI

Das leistungsfähigste Modell ist das hierarchische Modell. Zertifikate werden von Zertifizierungsinstanzen (*Certification Authority, CA*) nach der Prüfung der Identität ausgestellt und unterzeichnet. Die CAs selbst sind wiederum von übergeordneten CAs zertifiziert usw. Auf diese Weise entsteht eine Zertifizierungshierarchie, wie sie in Abbildung 3.5 zu sehen ist. Innerhalb solcher Zertifizierungshierarchien wird zwischen verschiedenen Arten von Instanzen unterschieden.

An der Spitze einer PKI steht die *Top Level Certification Authority* (TL-CA). Sie legt globale Verfahrensweisen fest, die für die gesamte PKI gelten.

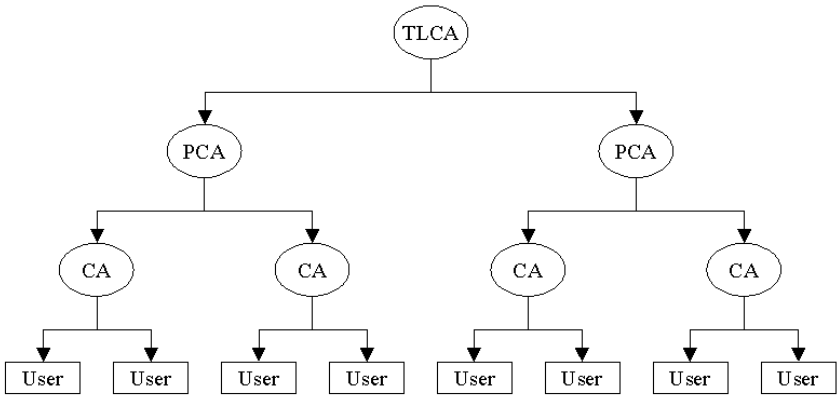

Abbildung 3.5: Zertifizierungshierarchie

Außerdem stellen sie die Zertifikate für die *Policy Certification Authorities* (PCA) aus.

Die *Policy Certification Authorities* (PCA) sind der TLCA direkt untergeordnet. Die PCAs legen Richtlinien und Regeln (*Policies*) für die Organisation der ihnen unterstellten Instanzen fest. Desweiteren zertifizieren die PCAs die ihnen untergeordneten *Certification Authorities* (CA), so dass diese für den Betrieb zugelassen sind.

Die *Certification Authorities* stellen die Zertifikate aus. Sie können sowohl die öffentlichen Schlüssel der eigentlichen Teilnehmer der PKI, also die von juristischen Personen, zertifizieren, als auch ihnen untergeordnete CAs zertifizieren. Für die CAs gelten die Regeln der für sie zuständigen PCA.

Zu den Regeln, die innerhalb einer hierarchischen PKI gelten, gehören der Sicherheitsstandard, die Richtlinien, die den Umgang und die Verwendung von Schlüsseln betreffen, und die Voraussetzungen, die ein Antragsteller erfüllen muss, um ein Zertifikat zu erhalten.

In einer hierarchischen PKI lassen sich leicht Zertifikate sperren, da dies zentral vorgenommen werden kann. Die Sperrung von Zertifikaten ist wichtig, falls beispielsweise ein Benutzer seinen privaten Schlüssel verliert. Natürlich ist es erforderlich, dass die Benutzer die Gültigkeit eines Zertifikats vor der Verwendung prüfen. Möchte ein Benutzer einen öffentlichen Schlüssel überprüfen, prüft er alle Zertifikate entlang des sogenann-

ten *Zertifizierungspfades*. Das bedeutet, ausgehend von dem zu prüfenden Schlüssel prüft er das Zertifikat der jeweils übergeordneten Instanz, bis er auf eine Instanz stößt, der er vertraut. Im ungünstigsten Fall ist dies die TLCA, die als allgemein vertrauenswürdig angesehen wird. Bei der vertrauenswürdigen Instanz kann es sich beispielsweise um eine anerkannte Organisation handeln oder, für den Fall, dass er selbst auch Teilnehmer dieser PKI ist, einer Instanz, der er selbst untergeordnet ist.

In der Realität gibt es jedoch nicht eine einzige Zertifizierungshierarchie, sondern es existieren mehrere voneinander unabhängige Zertifizierungshierarchien. Zertifikate von der einen PKI können von anderen PKIs nicht anerkannt werden. Abhilfe hierfür können sogenannte *Cross - Zertifikate* schaffen. Mit Hilfe von Cross - Zertifikaten können sich zwei PKIs gegenseitig ein Zertifikat ausstellen.

3.3.2 Web of Trust

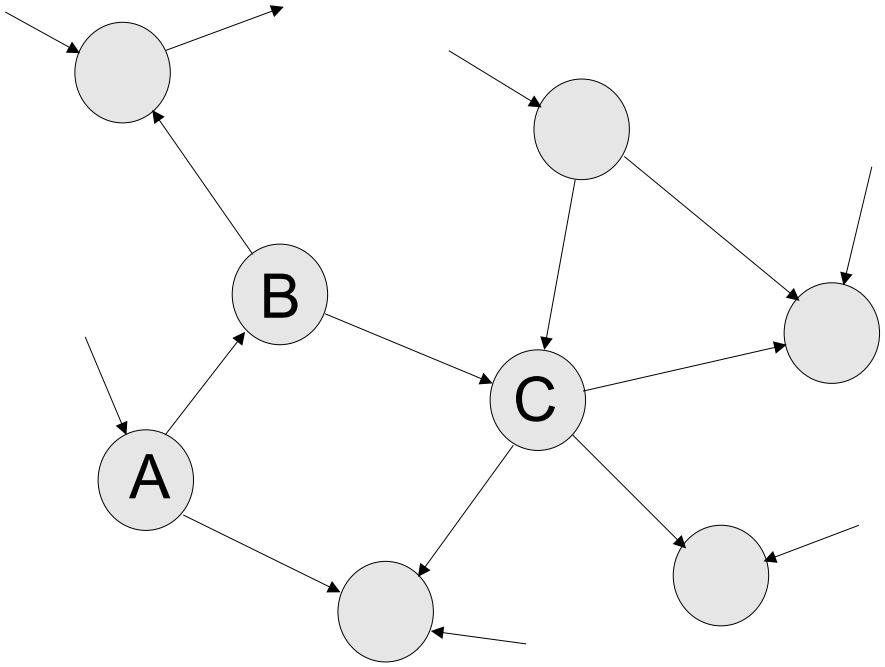

Abbildung 3.6: Web of Trust

Eine Alternative zu hierarchischen PKIs stellen die dezentralen PKIs dar, genannt *Web of Trust*. In einem *Web of Trust* kann jeder Benutzer Zertifikate für jeden anderen Benutzer ausstellen. Wenn ein Benutzer *A* einem Benutzer *B* vertraut, kann *A* nicht nur den öffentlichen Schlüssel

von B verwenden, sondern auch alle Schlüssel, die B zertifiziert hat. Da mit diesen Schlüsseln wiederum andere Zertifikate signiert wurden, kann A auch diesen vertrauen. Auf diese Weise bildet sich ein Netz von Zertifikaten, wie es in Abbildung 3.6 dargestellt ist. Ein Nachteil dieses dezentralen Ansatzes ist, dass es sehr schwierig ist, Richtlinien durchzusetzen, da es eben keine Instanz gibt, die für alle Teilnehmer gültige Regeln aufstellen kann. Genauso problematisch ist das Sperren von Zertifikaten. Möchte ein Benutzer sein Zertifikat sperren, muss er dies möglichst vielen anderen Benutzern mitteilen, und muss darauf hoffen, dass sich diese Nachricht im Netz verbreitet.

3.3.3 PKIs gemäß Signaturgesetz

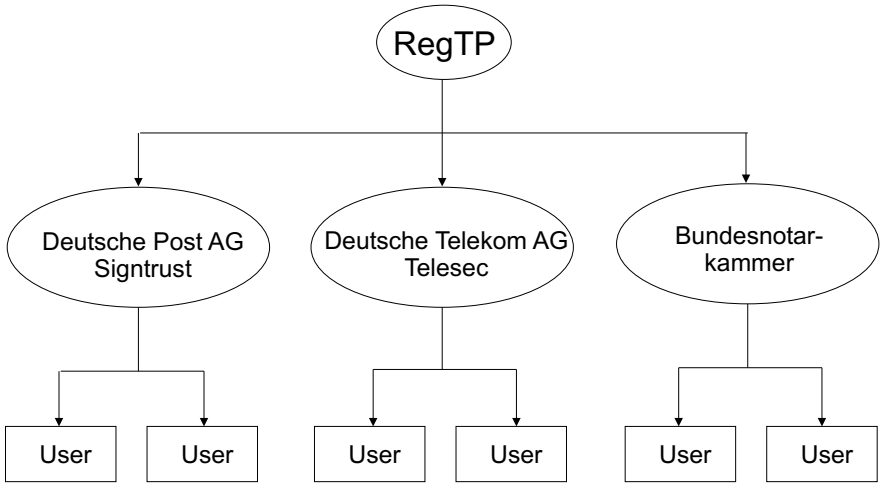

Abbildung 3.7: Zertifizierungshierarchie gemäß Signaturgesetz

Hier zu Lande legt das 1997 in Kraft getretene deutsche Signaturgesetz (*[SigG97]*) die Rahmenbedingungen für den Aufbau einer PKI fest. Das SigG sieht eine zweistufige hierarchische PKI vor (Abbildung 3.7). An der Spitze steht die Regulierungsbehörde für Telekommunikation und Post (RegTP). Ihr sind direkt die CAs untergeordnet. Bislang gibt es allerdings nur drei Zertifizierungsdienste, die die strengen Auflagen des SigG erfüllen, die der Telekom Tochter Telesec, die der Deutschen Post und das Trust Center der Bundesnotarkammer.

Zertifizierungsdienste haben einen umfangreichen Anforderungskatalog zu erfüllen, um die Genehmigung zum Betrieb zu erhalten. Die Anforderungen betreffen unter anderem auch die baulichen Beschaffenheiten der

Zertifizierungsdienste. Die hohen Kosten, die diese Anforderungen für den Betreiber verursachen, stellen einen Grund dar, warum sich digitale Signaturen bislang nicht auf dem Markt durchsetzen konnten.

Dies könnte sich jedoch mit dem neuen Signaturgesetz (*[SigG01]*) ändern, das Anfang März 2001 vom Bundesrat gebilligt wurde, und mit dem die entsprechende EU - Richtlinie umgesetzt wird. Das neue Signaturgesetz wird voraussichtlich Mitte 2001 in Kraft treten. Die Anforderungen an Zertifizierungsdienste werden mit dem neuen Gesetz deutlich gelockert. Insbesondere wird die Genehmigungspflicht durch eine Anzeigepflicht ersetzt. Eine Haftungsregelung soll sicherstellen, dass die Zertifizierungsdienste auf die Sicherheit achten. Sie müssen im Falle des Verschuldens für den Schaden aufkommen, den Dritte dadurch erleiden, dass sie auf ein ausgestelltes Zertifikat vertraut haben.

3.3.4 Formate für Zertifikate und Namen

Zertifikate liegen im allgemeinen im X.509 - Format vor. X.509 - Zertifikate enthalten die folgenden Informationen.

- Versionsnummer der verwendeten X.509 - Spezifikation
- Seriennummer des Zertifikats
- Signatur
- Name des Ausstellers (*Issuer*)
- Gültigkeitszeitraum
- Name des Eigentümers (*Subject*)
- Öffentlicher Schlüssel des Eigentümers, bestehend aus dem Bezeichner des verwendeten Algorithmus und des eigentlichen Schlüssels

Namen werden in Form von *Distinguished Names* (DN) gemäß der X.501 - Spezifikation angegeben. *Distinguished Names* haben folgendes Aussehen:

[C=Germany, L=Frankfurt/M., O=Uni Frankfurt, OU=Fachbereich Informatik, CN=Thomas Kunz]

Dabei steht C für das Land (*Country*), L für die Stadt (*Location*), O für die Organisation, OU für die *Organisation Unit* und CN für den Namen (*Common Name*). *[Rea96][SH00][NS01]*

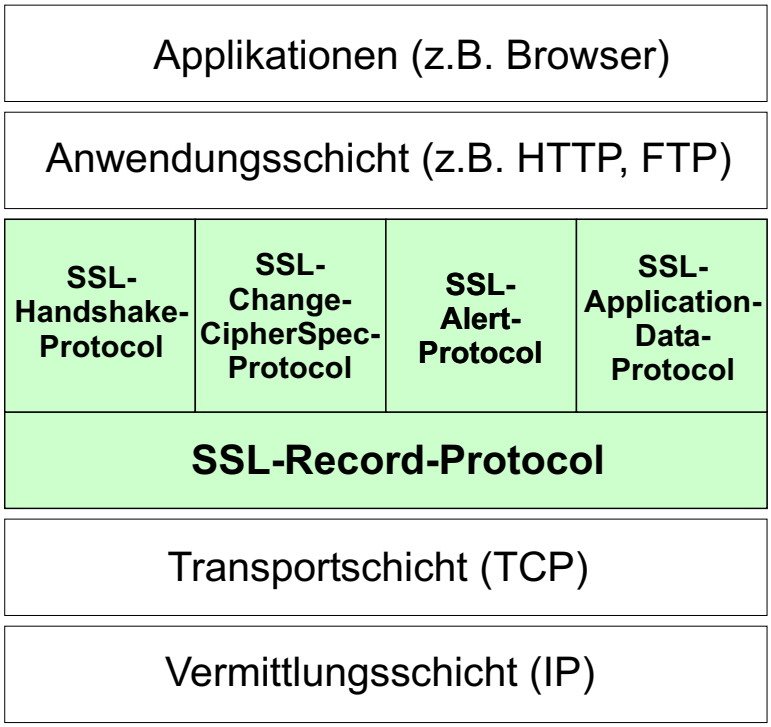

Abbildung 3.8: Das SSL - Protokoll im TCP/IP - Referenzmodell

3.4 SSL - Secure Socket Layer

Bei SSL (*Secure Socket Layer*) handelt es sich um ein Protokoll, das eine sichere Übertragung von Daten über das Internet ermöglicht. Zu diesem Zweck verschlüsselt SSL die Datenströme, die über den Kommunikationskanal übertragen werden. Es werden also nicht die Nachrichten direkt verschlüsselt, sondern die gesendeten Datenpakete. Damit ist es für einen Angreifer jedoch noch möglich, eine Nachricht vor der Übertragung zu manipulieren, SSL gewährleistet lediglich, dass die Nachricht während ihrer Übertragung nicht verändert werden kann. Zudem bietet SSL in der aktuellen Version 3 die Möglichkeit der Authentifizierung zwischen Client und Server.

Im TCP/IP - Referenzmodell[4] gliedert sich SSL zwischen der Transportschicht und der Anwendungsschicht ein (Abb. 3.8). In der Transportschicht kommt das TCP - Protokoll (*Transmission Control Protocol*) zum Einsatz, welches für die Übertragung einzelner Datenpakete zuständig ist. In der Anwendungsschicht arbeiten anwendungsorientierte Protokolle, wie

[4]wird z.B. in [Tan96] beschrieben

HTTP (Hypertext Transfer Protocol) zum Übertragen von Webseiten und FTP (*File Transfer Protocol*) zum Senden von Dateien. SSL selbst unterteilt sich in verschiedene Protokolle. Auf dem *SSL - Record - Protocol* setzen alle übrigen SSL - Protokolle auf. Seine wesentliche Aufgabe ist die Verschlüsselung der Daten. Das *SSL - Handshake - Protocol* dient zum Austausch der für die sichere Übertragung notwendigen Parameter. Dazu gehören die Authentifizierung, die Wahl der verwendeten Algorithmen und der Austausch der verwendeten Schlüssel. Mit Hilfe des *SSL - ChangeCipherSpec - Protocol* signalisieren die Kommunikationspartner, dass die zuvor mit dem Handshake - Protokoll festgelegten Parameter nun in Kraft treten. Das *SSL - Alert - Protocol* dient dem Senden von Fehlermeldungen, wie zum Beispiel der Zurückweisung eines Zertifikats, und dem regulären Beenden einer SSL - Verbindung. Das *SSL - ApplicationData - Protocol* schließlich hat die Aufgabe, die eigentlichen Daten mittels SSL zu übertragen.

Zur Verschlüsselung der Daten kommen bei SSL ausschließlich symmetrische Verschlüsselungsverfahren zum Einsatz. Bei digitalen Signaturen werden jedoch asymmetrische Verfahren verwendet. SSL unterstützt die folgenden Algorithmen.

- RSA, Diffie - Hellman, Fortezza (Asymmetrische Verfahren)
- DES, Triple - DES, IDEA, RC2, RC4 (Symmetrische Verfahren)
- MD5, SHA-1 (Hashalgorithmen)

Verschlüsselte Übertragung via SSL findet in der Regel zwischen Webservern und Browsern statt. SSL wird von allen gängigen Produkten, wie dem Apache Webserver oder dem Microsoft Information Server auf der Server - Seite und dem Microsoft Internet Explorer oder dem Netscape Communicator auf der Client - Seite, unterstützt. Für SSL - Übertragungen ist ein spezieller Port reserviert (Port 443). Übertragungen mittels SSL erkennt man im Browser durch die Darstellung eines geschlossenen Schlosses oder eines Schlüssels. *[FKK96]*

Kapitel 4

XML-Signaturen

Die elektronische Geschäftsabwicklung (E - Business), prägt mehr und mehr das moderne Wirtschaftsleben. Bereits heute erzielen Unternehmen wie Cisco oder Intel Umsätze in der Größenordnung von mehreren Milliarden Dollar allein im Online - Geschäft, und es ist abzusehen, dass dies erst der Anfang ist.

Unter E - Business versteht man die Unterstützung von Geschäftsprozessen durch elektronische Kommunikationsmittel. Damit sind vor allem Internetdienste, wie das World Wide Web oder E - Mail gemeint. Bislang fehlt es jedoch an einem einheitlichen und offenen Datenformat zur Darstellung dieser Geschäftsprozesse. Stattdessen gibt es viele unterschiedliche Standards, die sich oft nur schwer miteinander verbinden lassen.

An dieser Stelle soll insbesondere der elektronische Verkauf, der E - Commerce, hervorgehoben werden. Neben dem einfachen katalogbasierten Online - Shopping zählen dazu auch komplexere Szenarien wie Online Auktionen und elektronische Marktplätze. Die Notwendigkeit eines einheitlichen und leistungsfähigen Datenformats in diesen Geschäftsfeldern wird am Beispiel des Consumer - Bereichs deutlich. Der Kunde soll seine Geschäfte über das WWW tätigen[1] und er soll möglichst wenig zusätzliche Software neben seinem Browser installieren müssen. Bislang findet der Nachrichtenaustausch auf Basis von HTML statt. Für katalogbasiertes Online - Shopping mag das ausreichend sein. Wie schon in Kapitel 2 dargelegt, stößt man jedoch sehr schnell an die Grenzen von HTML, wenn es um die Weiterverarbeitung der übermittelten Daten geht. In komplexen Handelsszenarien, in denen Verhandlungen und rechtsgültige Vertragsabschlüsse notwendig sind, ist dies aber zwingend erforderlich.

[1] d.h., der Nachrichtenaustausch basiert auf dem HTTP - Protokoll

Die eXtensible Markup Language XML hat das Potential, sich in Zukunft zu dem wichtigsten Datenformat für den Nachrichtenaustausch in E - Commerce Szenarien entwickeln[2]. Zudem existieren bereits einige XML - Anwendungen, die zeigen, wie sich XML im E - Business einsetzen lässt. Dazu gehören beispielsweise OFX (*Open Financial Exchange*)[3], eine XML - Spezifikation zur Darstellung von Finanztransaktionen, und IOTP (*Internet Open Trading Protocol*, RFC 2801), eine XML - Anwendung zur Darstellung von Geschäftsprozessen. Im E - Business spielen jedoch Begriffe wie „Sicherheit“, „Nichtabstreitbarkeit“ und „Datenintegrität“ eine entscheidende Rolle[4]. Selbst das beste Datenformat für E - Business ist im Grunde genommen wertlos, wenn sich diese grundlegenden Anforderungen damit nicht realisieren lassen. Da es sich bei XML um einen offenen und beliebig erweiterbaren Standard handelt, sollte es grundsätzlich möglich sein, XML - Anwendungen zu schaffen, mit denen sich die genannten Anforderungen realisieren lassen.

In diesem Kapitel wird ein vielversprechender Ansatz in diese Richtung vorgestellt, eine Spezifikation zur Darstellung digitaler Signaturen in XML. *[Reb00]*

4.1 Überblick

Anfang 1999 bildete das W3C zusammen mit der *Internet Engineering Taskforce* IETF eine Arbeitsgruppe, die sich mit der Spezifikation einer XML-Syntax zur Darstellung digitaler Signaturen beschäftigt. Der Standard befindet sich noch in der Entwicklung, steht aber kurz vor der Verabschiedung.

Die vorliegende Arbeit beruht auf der *Candidate Recommendation* vom 31. Oktober 2000 (*[ERS00]*). Der darin beschriebene Standard für XML-Signaturen macht bereits einen sehr ausgereiften Eindruck und es ist davon auszugehen, dass bis zur endgültigen Verabschiedung keine gravierenden Änderungen mehr stattfinden werden.

Die Bezeichnung „XML - Signatur“ soll andeuten, dass die digitale Signatur in XML dargestellt wird. Das bedeutet jedoch nicht, dass ausschließlich XML - Daten gemäß dieser Spezifikation signiert werden können. Es können vielmehr beliebige binäre Daten signiert werden.

[2] vgl. Kapitel 2. Dort werden die Vorteile von XML beschrieben

[3] http://www.ofx.com

[4] Diese Begriffe werden in Kapitel 3 näher erläutert.

Mit Hilfe von XML - Signaturen ist es möglich, beliebig viele Daten zusammen zu fassen und zu signieren. Das heißt, statt für jedes Datenobjekt eine Signatur zu erzeugen, wird nur ein einziger Signaturwert berechnet. Dies stellt einen wichtigen Aspekt dar, da die Generierung digitaler Signaturen, insbesondere digitaler Signaturen auf Basis asymmetrischer kryptographischer Verfahren, eine sehr rechenintensive Operation darstellt. Erreicht wird dies durch eine Indirektion. Es werden nicht die Daten selbst signiert, sondern Referenzen auf diese Daten. Diese Referenzen wiederum enthalten, neben einigen anderen Informationen, Hashwerte der zu signierenden Daten.

Der XML - Signatur Standard unterscheidet zwischen drei Arten von Signaturen.

1. *Enveloping signatures*. Die Signatur und die Daten befinden sich im selben Dokument, und die XML - Signatur umschließt die Daten. Das bedeutet, die Signatur und die Daten stehen in einer Eltern - Kind - Beziehung zueinander.

2. *Enveloped signatures*. Auch hier liegen die Signatur und die Daten im selben Dokument. Allerdings umhüllen in diesem Fall die zu signierenden Daten die XML - Signatur. Auch hier stehen Signatur und Daten in einer Eltern - Kind - Beziehung zueinander, jedoch im umgekehrten Fall. Natürlich hat bei diesen Signaturen die jeweilige Applikation dafür zu sorgen, dass die XML - Signatur nicht versucht, „sich selbst zu signieren“. Entsprechendes gilt auch für die Validierung, bei der alle Daten, die zur XML - Signatur gehören, herausgenommen werden müssen, bevor der Hashwert über den signierten Daten berechnet wird.

3. *Detached signatures*. Diese Variante stellt eine Neuerung gegenüber der gewohnten Vorstellung von digitalen Signaturen dar. Die Signatur ist in diesem Fall von den Daten abgekoppelt. Es ist sogar möglich, mit einer URL eine Referenz auf eine beliebige WWW - Ressource zu bilden, die signierten Daten liegen also nicht in dem Dokument, in dem sich die XML - Signatur befindet. Liegen die XML - Signatur und die Daten im selben Dokument, bedeutet dies für *detached signatures*, dass sich Signatur und Daten in einer Geschwister - Beziehung zueinander befinden. *[ERS00]*

4.2 Beschreibung der wichtigsten Elemente

Die Abbildung 4.1 zeigt den grundlegenden Aufbau einer XML - Signatur.

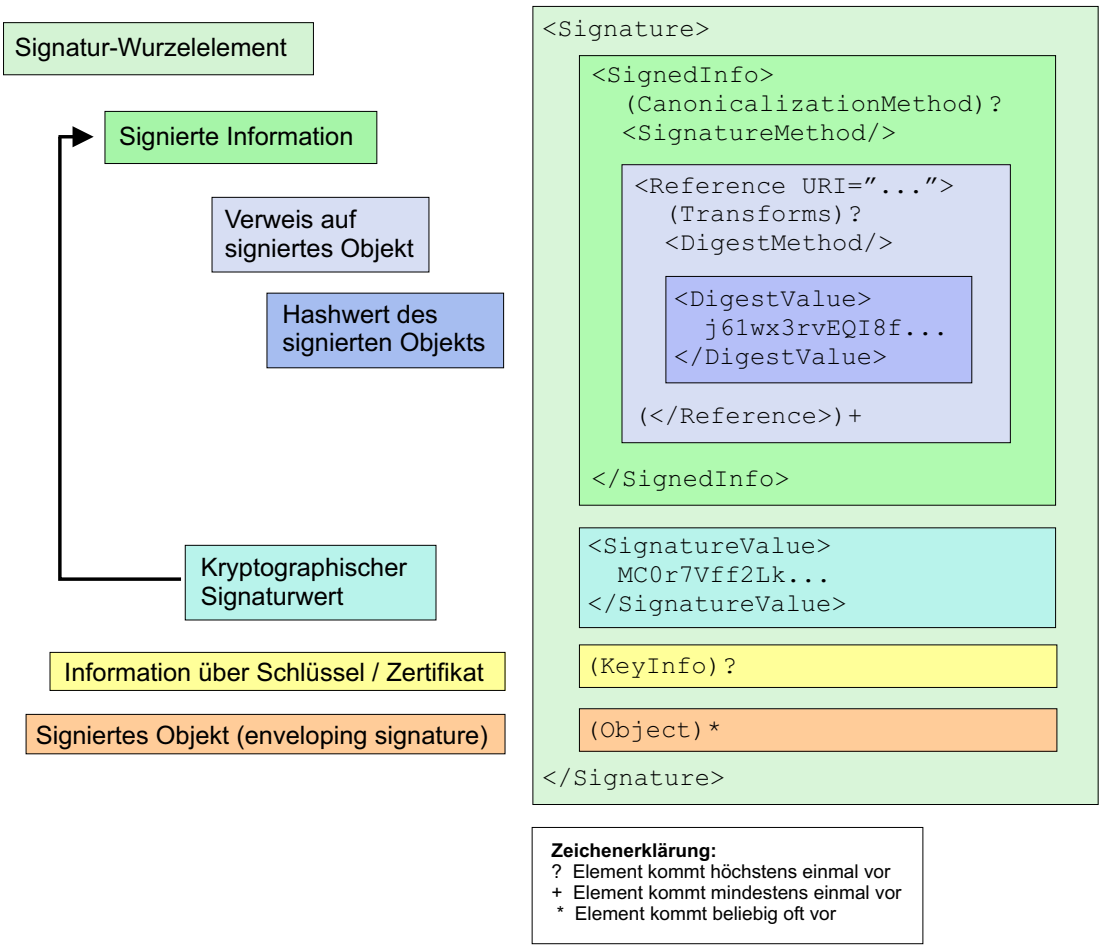

Abbildung 4.1: Grundstruktur einer XML - Signatur

4.2.1 Signature und SignedInfo

Jede XML - Signatur wird mit dem *Signature* - Element, dem Wurzelelement, eingeleitet. Das *SignedInfo* - Element stellt den Teil dar, der letztendlich signiert wird. Dieses Element enthält Informationen über die Signatur und die Daten, auf die sie sich bezieht.

Das *CanonicalizationMethod* - Element gibt den Algorithmus an, mit dem das *SignedInfo* - Element kanonisiert wird, bevor es signiert wird (s.a. Abschnitt 4.3).

Mit *SignatureMethod* wird der Algorithmus festgelegt, mit dem *SignedInfo* signiert wird. Es handelt sich hierbei um eine Kombination aus Verschlüsselungs - und Hashalgorithmus (z.B. RSA-SHA1).

4.2.2 Reference

Die *Reference* - Elemente stellen den Verweis auf die zu signierenden Daten dar. Der Bezug wird über das URI - Attribut hergestellt. Es kann sich dabei um eine komplette URL handeln, falls externe Daten signiert werden (*detached signatures*), oder den Wert eines ID - Attributs innerhalb des gleichen Dokuments. Aus den zu signierenden Daten wird ein Hashwert gebildet. Dieser wird Base64 - kodiert in dem *DigestValue* - Element abgelegt. *DigestMethod* spezifiziert den dazu verwendeten Algorithmus (in der Regel SHA1).

An dieser Stelle tritt die schon zuvor beschriebene Indirektion in Erscheinung. Die *Reference* - Elemente enthalten Hashwerte über den zu signierenden Daten. Die *Reference* - Elemente wiederum liegen innerhalb des *SignedInfo* - Elements, das schließlich signiert wird.

Bevor der Hashwert berechnet wird, können verschiedene Transformationsalgorithmen auf die Daten angewandt werden. Im Falle von XML - Daten ist es beispielsweise denkbar, die Daten zu kanonisieren, eine XSLT - oder eine XPath - Transformation durchzuführen. Bei beliebigen binären Daten ist eine Base64 - Kodierung sinnvoll. Die verwendeten Verfahren werden im *Transforms* - Element festgelegt.

Es ist prinzipiell möglich, mehrere Transformationsalgorithmen anzugeben. Die Verfahren werden dann nacheinander durchgeführt, so dass die Ausgabe der einen Transformation als Eingabe der nächsten dient.

Gerade bei der Verwendung mächtiger Transformationen wie XSLT oder XPath, mit deren Hilfe man die zu signierenden Daten in eine völlig andere Darstellungsform überführen kann oder man Teile der Daten von der Signatur ausschließen kann, ist jedoch Vorsicht geboten. Zum einen sind nur die Daten wirklich sicher, die auch tatsächlich signiert wurden, zum anderen sollte das und nur das signiert werden, was dem Benutzer auch präsentiert wird.

Ein Fall, bei dem eine XPath - Transformation sogar unerlässlich sein kann, sind *enveloped signatures*. Bei *enveloped signatures* stellt eine XPath-Transformation eine Möglichkeit dar, vor der Berechnung der Hashwerte die XML - Signatur herauszunehmen.

Ein Problem kann sich im Zusammenhang mit den URI - Attributen ergeben. Wie ist zu verfahren, wenn sich im Falle externer Daten der Ort der Daten ändert, man also nicht mehr über die angegebene URL auf sie zugreifen kann? Leider bietet die derzeitige Spezifilation keine Lösung für diesen Sachverhalt. In diesem Zusammenhang stellt sich grundsätzlich die Frage, wie streng die angegebenen URIs auszulegen sind. Legt man sie streng aus, verlangt man also, dass die URIs auch tatsächlich auf die Daten verweisen, würde das bedeuten, dass die Validierung der Referenz fehlschlägt, wenn sich der Ort der Daten ändert. Somit schlägt dann natürlich auch die Validierung der gesamten XML - Signatur fehl. Eine Lösung für obiges Problem wäre, die URIs lediglich als Hinweise zu interpretieren, von woher die Daten zu beziehen sind. Stehen dem Benutzer die signierten Daten sowieso zur Verfügung, muss er sich bei der Validierung nicht auf die URIs verlassen.

4.2.3 SignatureValue

Das *SignatureValue* - Element stellt die eigentliche Signatur dar, das heißt den verschlüsselten Hashwert über dem *SignedInfo* - Element (Base64 - kodiert).

4.2.4 KeyInfo

Über das optionale *KeyInfo* - Element besteht die Möglichkeit, Informationen über den verwendeten Schlüssel oder sogar den öffentlichen Schlüssel inklusive Zertifikat selbst in die Signatur zu integrieren und auf diesem Wege dem Empfänger des Dokuments zur Verfügung zu stellen.

Zu diesem Zweck sind in der Spezifikation verschiedene Elemente definiert. Beispiele sind *X509Data*, das ein X.509 - Zertifikat enthalten kann, oder *RetrievalMethod*, mit dem man eine URL angeben kann, unter der der Schlüssel zu finden ist.

4.2.5 Object

Das ebenfalls optionale *Object* - Element beinhaltet die zu signierenden Daten. Es wird verwendet, wenn sich die Daten innerhalb der Signatur

befinden (*enveloping signaturs*). Über ein ID - Attribut wird der Bezug zu dem dazugehörenden *Reference* - Element hergestellt.

4.2.6 Manifest

Manifest - Elemente befinden sich in der Regel in *Object* - Elementen. In *Manifest* - Elemente können beliebig viele *Reference* - Elemente integriert werden. Diese *Reference* - Elemente sind von der Validierung ausgeschlossen, da bei der Validierung nur die *Reference* - Elemente innerhalb von *SignedInfo* geprüft werden. Die Verwendung von *Manifest* - Elementen bietet sich an, wenn eine Applikation eine XML - Signatur als gültig betrachten soll, wenn der Signaturwert in Ordnung ist, unabhängig von der Validierung der Referenzen.

4.2.7 Verwendete Algorithmen

Die verschiedenen verwendeten Algorithmen werden anhand festgelegter URIs identifiziert. Zum berechnen der Hashwerte sieht die Spezifikation ausschließlich SHA1 vor. Zum Signieren wird zumindest DSA verlangt, es wird jedoch empfohlen, den RSA - Algorithmus zu verwenden.

Zum Transformieren sind URIs für XML - Kanonisierung, XSLT, XPath und Base64 - Kodierung definiert. Applikationen können zwar auch andere Algorithmen verwenden, es ist dann jedoch nicht mehr garantiert, dass diese dann bei der Validierung auch korrekt erkannt werden. *[ERS00]*

4.3 Canonical XML

Dieselbe Arbeitsgruppe, die die Spezifikation für XML - Signaturen entwickelt, beschäftigt sich auch mit kanonisierten XML - Dokumenten. Kanonisierte Formen von XML - Dokumenten bekommen eine große Bedeutung im Zusammenhang mit digitalen Signaturen. Digitale Signaturen funktionieren nur, wenn beim Verifizieren der Signatur exakt dieselbe Bitfolge zur Verfügung steht, auf deren Basis zuvor die Signatur erstellt wurde. Bei XML - Dokumenten ist dies jedoch nicht zwingenderweise der Fall. Zwei XML - Dokumente können sich durchaus in einigen Bytes unterscheiden, sind aber trotzdem noch als äquivalent zu betrachten.

Ein simples Beispiel hierfür sind unterschiedliche Zeilenende - Markierungen oder Leerzeichen bzw. Tabulatoren, die zum Einrücken verwendet werden. Würde man verlangen, dass sich ein XML - Dokument zwischen

Signieren und Validieren in keinem Byte ändern darf, würde man viel von der Flexibilität, die XML bietet, einbüßen. Das folgende Beispiel soll die Notwendigkeit von kanonisiertem XML veranschaulichen.

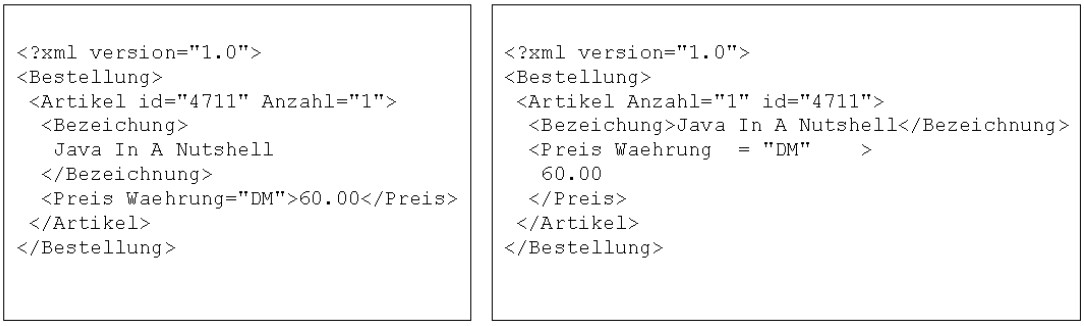

Abbildung 4.2: Darstellungsformen eines XML - Dokuments

Die Abbildung 4.2 zeigt zwei Repräsentationen ein und desselben XML - Dokuments. Im zweiten Dokument wurde die Reihenfolge der Attribute im *Artikel* - Element vertauscht, das *Bezeichung* - Element wird komplett in einer Zeile dargestellt und in dem *Preis* - Element sind ein paar Leerzeichen hinzugekommen. Es ist jedoch leicht einzusehen, dass diese beiden Dokumente als logisch äquivalent zu betrachten sind. Trotzdem würden sich beim Signieren zwei unterschiedliche Signaturwerte ergeben. Dieses Beispiel zeigt die Notwendigkeit einer einheitlichen, kanonisierten Darstellungsform für XML - Dokumente.

Die Arbeitsgruppe arbeitet zur Zeit an einer Methode zur Generierung kanonisierter XML - Dokumente, die vorliegende Arbeit bezieht sich auf die *Proposed Recommendation* vom 15. Januar 2001 (*[Boy01]*). Die wesentlichen Merkmale kanonisierter XML - Dokumente sind die Verwendung einheitlicher Zeilenende - Zeichen und die Entfernung überflüssiger Leerzeichen innerhalb von XML - Elementen. Desweiteren werden einige Sonderzeichen durch spezielle Zeichenfolgen ersetzt, beispielsweise das „&“ - Zeichen durch „&“ oder das „<“ - Zeichen durch „<“. Leere XML - Elemente werden durch Start - / Ende - Elemente ersetzt. Da sich gerade durch das Parsen von XML - Dokumenten oder durch die Darstellung in einer DOM - Struktur die Reihenfolge von Attributen ändern kann, werden in der kanonischen Form die Attribute in lexikographischer Reihenfolge angeordnet. Zur Zeichenkodierung wird UTF-8 verwendet. *[Boy01]*

4.4 Beschreibung der Java-API

4.4.1 Paketübersicht

Im Rahmen dieser Arbeit wurde eine API zur Generierung und Validierung von XML - Signaturen entwickelt. Die API verwendet zum Signieren die JCE (*Java Cryptography Extension*) Implementierung von *SECUDE*.

Die *Java Cryptography Extension* stellt ein Interface zu den kryptographischen Algorithmen dar, und enthält diverse abstrakte Klassen zur Beschreibung der API. Konkrete Implementierungen dieser Klassen werden von verschiedenen Anbietern zur Verfügung gestellt. Ermöglicht wird dies durch die Bereitstellung einer *Security Provider* Infrastruktur. Dadurch können die zur Laufzeit vorhandenen Implementierungen der Klassen benutzt werden, ohne den Programmcode ändern zu müssen. JCE stellt also eine konsistente API zur Verfügung, die unabhängig von dem verwendeten Provider benutzt werden kann.

Der Grund, warum die JCE - Implementierung von *SECUDE* gewählt wurde, ist die Unterstützung von Smartcards. Desweiteren ist der XML - Parser *Xerces* und der XSLT - Prozessor *Xalan*, beide von *Apache*, erforderlich. Die Anforderungen sind im folgenden nochmals zusammengefasst. *[Oak98]*

1. Java 1.2 (http://www.java.sun.com)
2. SECUDE JCA - Provider (http://www.secude.com)
3. Apache Xerces for Java 1.0.3 (http://xml.apache.org/xerces)
4. Apache Xalan for Java 1.0.1 (http://xml.apache.org/xalan)

Die API - Klassen wurden in drei Pakete unterteilt.

1. `de.gmd.xmlsig.signature` (zum Erzeugen von XML - Signaturen)
2. `de.gmd.xmlsig.validation` (zum Validieren von XML - Signaturen)
3. `de.gmd.xmlsig.util` (diverse Hilfsklassen)

Das Ziel bei der Entwicklung der Klassen zum Erzeugen der XML - Signaturen war, die wesentlichen Elemente einer XML - Signatur in einzelne Klassen zu unterteilen. Dadurch entstanden die folgenden Klassen.

1. `public class de.gmd.xmlsig.signature.Reference()`

2. `public class de.gmd.xmlsig.signature.Manifest()`

3. `public class de.gmd.xmlsig.signature.KeyInfo()`

4. `public class de.gmd.xmlsig.signature.XMLSignature()`

4.4.2 Beschreibung der Klassen

In dem folgenden Abschnitt werden die wichtigsten Methoden der einzelnen Klassen vorgestellt. Eine detaillierte Beschreibung ist auf der beigelegten CD-ROM in Form von *javadoc*-Dateien zu finden.

Die Klasse *KeyInfo* stellt das *KeyInfo* - Element dar. Sie enthält Methoden zum Setzen eines X.509 - Zertifikats und zum Setzen sonstiger Informationen über den verwendeten Schlüssel. Die Klasse generiert die DOM - Struktur des Elements.

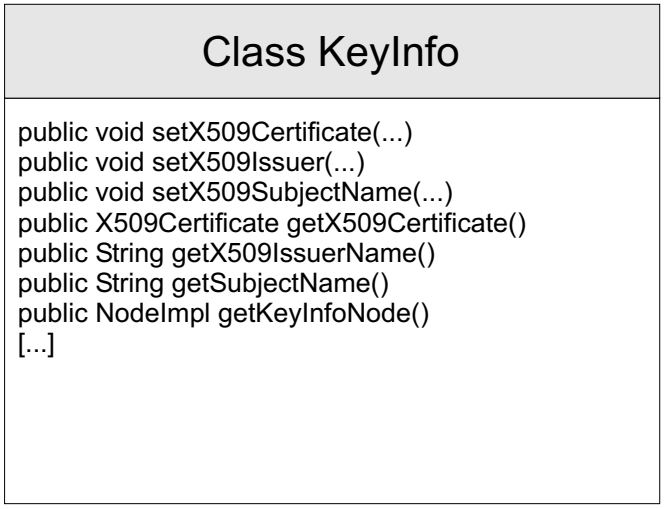

Abbildung 4.3: Klasse „KeyInfo“

Mit der Klasse *Reference* lassen sich Verweise auf die zu signierenden Daten realisieren. Die Klasse unterstützt alle drei Spielarten von XML - Signaturen, also *enveloping*, *enveloped* und *detached signatures*. Zu diesem Zweck werden verschiedene Methoden zur Verfügung gestellt, die die Art der Signatur festlegen und die zu signierenden Daten spezifizieren. Bei *detached signatures*, die sich auf externe Resourcen beziehen, werden die Daten über eine URL bestimmt. Bei den anderen Arten von XML - Signaturen

werden die Daten als DOM - Objekt übergeben, sofern es sich um XML - Daten handelt. Mit *enveloping signatures* können auch beliebige binäre Daten signiert werden, die von einem *InputStream* gelesen werden. Diese binären Daten werden dann Base64 - kodiert in einem *Object* - Element abgelegt. Neben den zu signierenden Daten können diverse Transformationsalgorithmen angegeben werden, wie XSLT - Transformation, XML - Kanonisierung oder Base64 - Kodierung. Als Hashalgorithmus steht nur SHA-1 zur Verfügung. Wurden alle nötigen Informationen angegeben, generiert die Klasse ein DOM - Objekt des kompletten *Reference* - Elements und gegebenenfalls ein DOM - Objekt des *Object* - Elements.

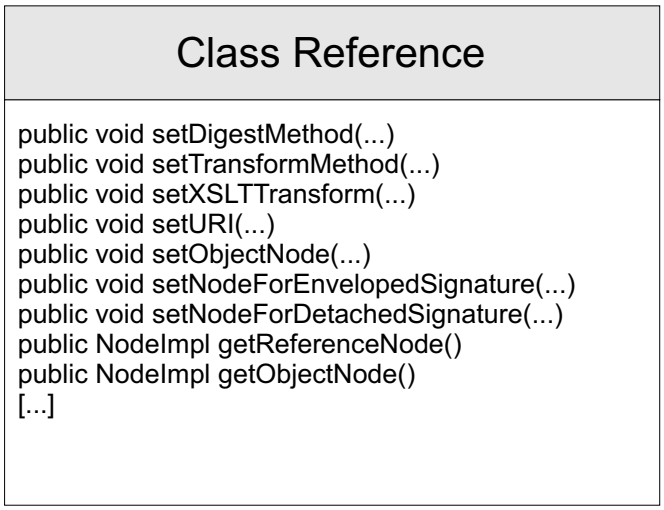

Abbildung 4.4: Klasse „Reference“

Mit der Klasse *Manifest* lassen sich *Manifest* - Elemente erzeugen. Die Klasse stellt Methoden bereit, mit denen sich beliebig viele *Reference* - Objekte zu dem Manifest hinzufügen und auch wieder entfernen lassen. Zu jedem *Manifest* - Element gehört ein *Reference* - Element, das auf das Manifest verweist und durch das das *Manifest* - Element signiert wird. Daher beinhaltet die Klasse auch Methoden, mit denen sich die nötigen Informationen, wie die zu verwendeten Algorithmen, für dieses *Reference* - Element angeben lassen. Die Klasse generiert also zum einen das *Manifest* - Element, zum anderen das dazugehörige *Reference* - Element.

Die Klasse *XMLSignature* repräsentiert das komplette *Signature* - Element. Die Klasse enthält Methoden zum Hinzufügen beliebig vieler *Reference* - und *Manifest* - Objekte, und zum Entfernen und Zurückliefern dieser Objekte. Daneben lässt sich ein optionales *KeyInfo* - Objekt in die XML -

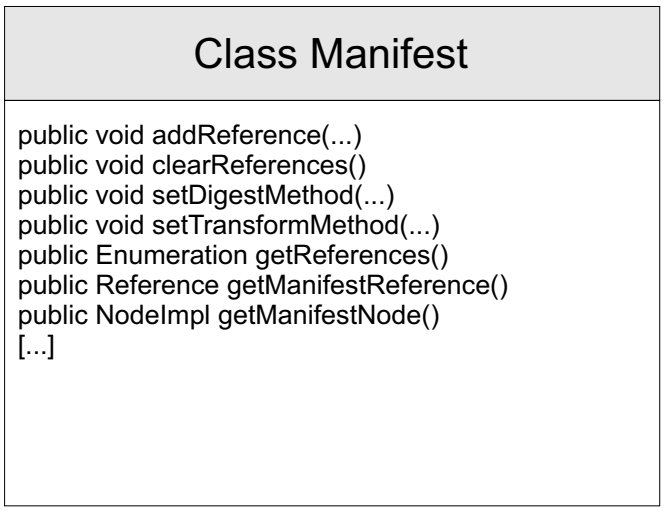

Abbildung 4.5: Klasse „Manifest“

Signatur integrieren. Natürlich lässt sich auch ein Kanonisierungs - und ein Signaturalgorithmus spezifizieren. Da als *Krypto - Engine* SECUDE verwendet wird, steht als Signaturalgorithmus nur RSA zur Verfügung. Analog zu den anderen Klassen generiert diese Klasse ein DOM - Objekt des *Signature* - Elements. Die generelle Vorgehensweise zum Erzeugen einer XML - Signatur ist also die folgende. Man erzeugt gegebenenfalls ein *KeyInfo* - Objekt und die benötigten *Reference* - und *Manifest* - Objekte. Dazu müssen natürlich die zu signierenden Daten zur Verfügung stehen. Diese ganzen Objekte werden dann an das *XMLSignature* - Objekt übergeben, das daraus das komplette *Signature* - Element bildet. Bei *enveloped signatures* und bei *detached signatures* auf XML - Daten innerhalb desselben Dokuments ist noch ein weiterer Schritt notwendig. Die jeweilige Anwendung hat dafür zu sorgen, dass das zuvor erzeugte *Signature* - Element an das XML - Dokument angehängt wird, das die signierten Daten enthält. Dabei ist bei *enveloped signatures* darauf zu achten, dass das *Signature* - Element auch tatsächlich ein Kind des signierten Elements ist.

Die Klasse *XMLSignatureValidation* hat im wesentlichen zwei Funktionen. Zum einen dient sie der Validierung von XML - Signaturen. Zum anderen lässt sich mit Hilfe dieser Klasse die Struktur einer XML - Signatur analysieren. Zum Validieren stehen verschiedene Methoden zur Verfügung, um entweder das komplette Dokument zu validieren oder um gezielt einzelne Signaturen oder sogar nur eine einzelne Referenz zu prüfen. Die Klasse erkennt und prüft auch in einander verschachtelte Signaturen. Bei der Validierung wird zwischen der sogenannten *core validati-*

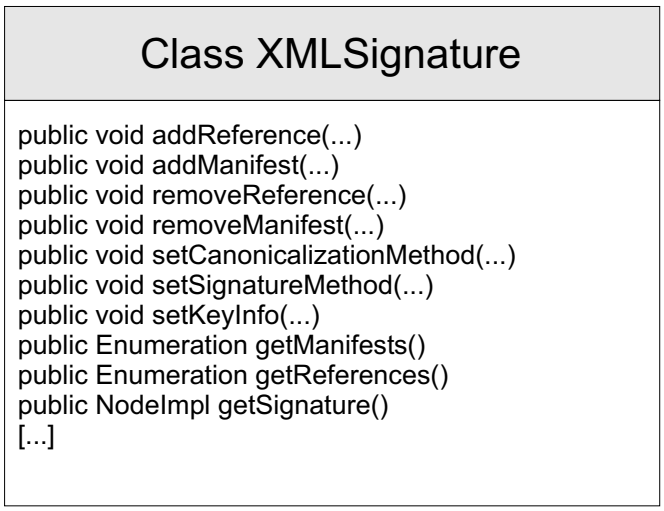

Abbildung 4.6: Klasse „XMLSignature“

on unterschieden, bei der Referenzen innerhalb von *Manifest* - Elementen nicht überprüft werden, und der kompletten Validierung, die auch die *Manifest* - Elemente einschließt. Die Methoden zur Validierung liefern *ValidationInfo* - Objekte zurück (s.u.), die das Ergebnis der Überprüfung enthalten. Dadurch lässt sich im Falle einer nicht erfolgreichen Validierung feststellen, welche Signatur oder welche Referenz nicht in Ordnung ist. Da zum Überprüfen einer Signatur der öffentliche Schlüssel benötigt wird, kann man an das *XMLSignatureValidation* - Objekt ein X.509 - Zertifikat übergeben, falls die XML - Signatur selbst kein Zertifikat enthält. Um die Struktur einer XML - Signatur zu analysieren, stellt die Klasse diverse Methoden zur Verfügung, die die Anzahl der Signaturen und der Referenzen und Manifeste innerhalb einer Signatur liefern, und auch Verweise auf diese Elemente liefern.

Die Klasse *ValidationInfo* beinhaltet das Ergebnis einer Validierung. Dazu stehen Methoden zum Setzen und Auslesen der Resultate einer Signatur - Überprüfung zur Verfügung. Die Ergebnisse einer Überprüfung werden in Form von *boolean* - Werten geliefert. Daneben dient die Klasse aber auch zur Darstellung des Validierungsergebnisses. Zu diesem Zweck generiert die Klasse ein XML - Dokument, das die entsprechenden Informationen enthält.

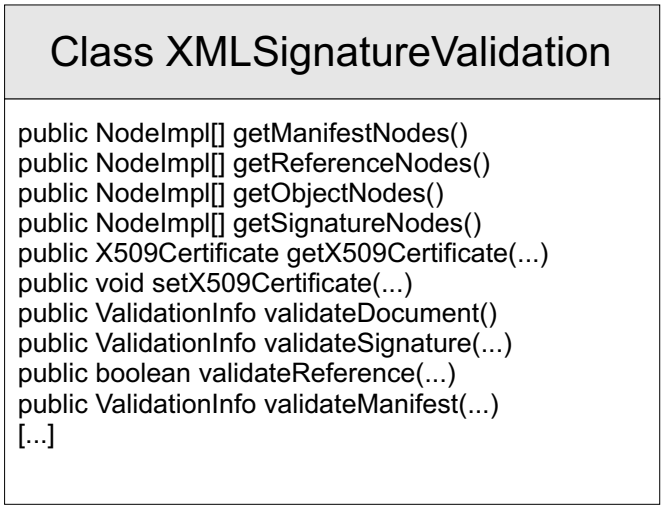

Abbildung 4.7: Klasse „XMLSignatureValidation“

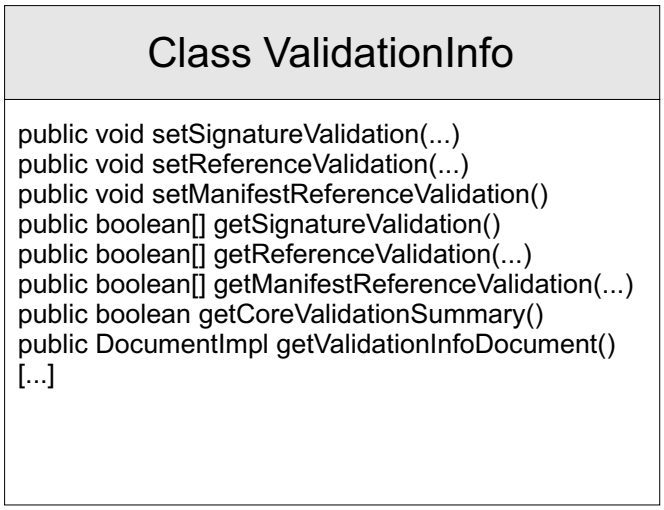

Abbildung 4.8: Klasse „ValidationInfo“

Kapitel 5

Das BAKO - Protokoll

BAKO (Basis Kommunikation) ist ein an der GMD entwickeltes Geschäftsprotokoll. Bei diesem Protokoll geht es darum, eine gegenseitige Übereinkunft zwischen zwei Handelspartnern zu erzielen, die sich nicht abstreiten lässt. Diese gegenseitige Übereinkunft kann beispielsweise ein Vertragsabschluss zwischen einem Händler und einem Kunden sein. Im folgenden sollen die Bezeichnungen *Händler* und *Kunde* für die beiden Parteien beibehalten werden. *Nichtabstreitbarkeit* bedeutet nun, dass jede der beiden Parteien weiß, dass der andere dem Vertrag zugestimmt hat und dies auch nicht mehr abstreiten kann. Dies lässt sich durch einen zweifach unterzeichneten Vertrag erreichen. Beide Parteien haben den Vertrag unterzeichnet, und jeder erhält eine Kopie dieses Vertrags.

Mit dem BAKO - Protokoll soll diese Nichtabstreitbarkeit mit Hilfe digitaler Signaturen in elektronischen Geschäftsprozessen erzielt werden. Der Ablauf des BAKO - Protokolls besteht aus drei Schritten (siehe Abbildung 5.1). Im ersten Schritt signiert der Händler den bis dahin noch unverbindlichen Vertrag. Erst durch die Signatur des Händlers entsteht daraus ein verbindliches Angebot. Er sendet dieses Angebot an den Kunden. Akzeptiert der Kunde das Angebot, setzt er seine Unterschrift unter das Angebot und sendet das Dokument zurück an den Händler. Dadurch entsteht eine verschachtelte Signatur, das heißt, der Kunde signiert mit seiner Unterschrift die Signatur des Händlers. Nun besitzen beide Parteien einen zweifach signierten Vertrag. Allerdings könnte der Händler nachträglich bestreiten, das Dokument mit der Signatur des Kunden erhalten zu haben, und sich daher weigern, seinen Teil des Vertrags zu erfüllen. Daher ist noch ein dritter Schritt notwendig. Der Händler signiert im dritten Schritt das Dokument mit den beiden Signaturen und sendet es zurück an den Kunden. Dieser Schritt dient der Sicherheit des Kunden, denn nun kann der Kunde beweisen, dass auch der Händler ein zweifach signiertes Dokument besitzt. Jede der beiden Parteien besitzt nach Ablauf des

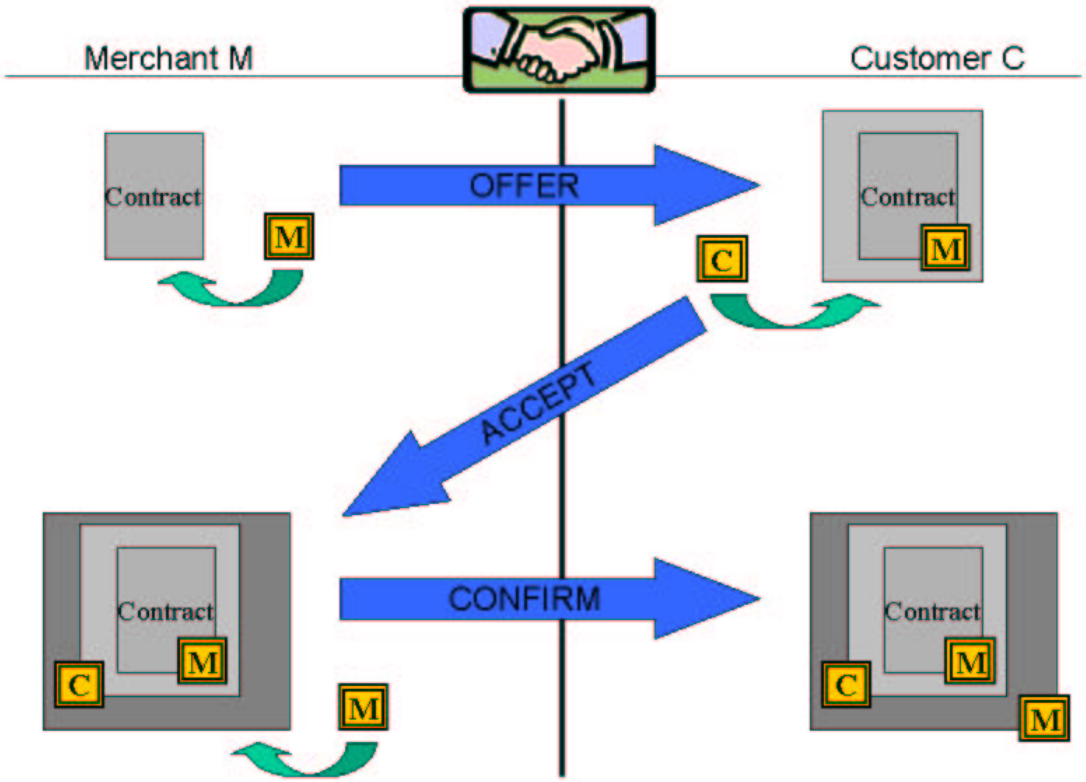

Abbildung 5.1: Schematische Darstellung des BAKO - Protokolls

Protokolls also ein Dokument mit mindestens zwei Signaturen, und kann nachweisen, dass die andere Partei ebenfalls ein solches Dokument besitzen muss. Der Kunde kann den Vertrag nicht abstreiten, da der Händler ein Dokument besitzt, das die Signatur des Kunden enthält, mit der er den Vertrag akzeptiert hat. Der Händler seinerseits kann den Vertrag ebenfalls nicht abstreiten, da der Kunde ein Dokument besitzt, das bestätigt, dass der Händler einen zweifach signierten Vertrag erhalten hat. Keine Partei kann also den Vertragsabschluss abstreiten. *[Kol96]*

5.1 BAKO mit XML - Signaturen

Im folgenden wird gezeigt, wie sich das BAKO - Szenario mit Hilfe von signierten XML - Dokumenten modellieren lässt. In diesem Zusammenhang wird auch dargelegt, wie XML - Signaturen generiert und validiert wer-

den. Bei den zu signierenden Daten handelt es sich um den Vertragstext, ein XML - Fragment, das kenntlich macht, dass es sich um ein BAKO - Dokument handelt und natürlich die XML - Signaturen selbst. Es wird davon ausgegangen, dass der Vertrag als externe Ressource vorliegt (*detached signature*). Dadurch kann der Vertrag problemlos in einem beliebigen binären Format abgelegt werden. Das BAKO - XML - Fragment, erkennbar an dem BAKO - Namensraum, enthält ein Element, das deutlich macht, um welches BAKO - Dokument es sich handelt. Daneben enthält es einen Zeitstempel. Der Zeitstempel gibt an, zu welchem Zeitpunkt das Dokument signiert wurde.

Im folgenden werden die XML - Dokumente der besseren Übersicht wegen schematisch dargestellt. Auf einige Details wurde bei der Darstellung verzichtet. Die kompletten XML - Dokumente befinden sich im Anhang A. In der ersten Phase des BAKO - Protokolls generiert der Händler das Angebot (*Offer* - Dokument, siehe Abbildung 5.2).

Zur Generierung dieses Dokuments sind folgende Schritte nötig:

1. Es wird der Transformationsalgorithmus auf den Vertrag angewendet. Liegt der Vertrag in Form eines beliebigen XML - Dokuments vor, kann der Kanonisierungsalgorithmus verwendet werden.

2. Es wird der Hashwert über dem kanonisierten Vertrag berechnet.

3. Es wird das *Reference* - Element für den Vertrag gebildet. Da es sich bei dem Vertrag um eine externe Ressource handelt, verweist eine URL auf den Vertrag. Der im zweiten Schritt berechnete Hashwert wird in dem *DigestValue* - Element abgelegt.

4. Es wird ein BAKO - Dokument erzeugt, das kenntlich macht, dass es sich um ein *Offer* - Dokument handelt. Dieses Dokument enthält auch den Zeitstempel (*Timestamp* - Element), der den Zeitpunkt des Signierens festhält. Das BAKO - Dokument wird in einem *Object* - Element abgelegt.

5. Über das *Object* - Element wird der Hashwert berechnet.

6. Analog zum dritten Schritt wird ein *Reference* - Element für das *Object* - Element gebildet.

7. Beide *Reference* - Elemente, sowie die *SignatureMethod* - und *CanonicalizationMethod* - Elemente werden in das *SignedInfo* - Element gepackt.

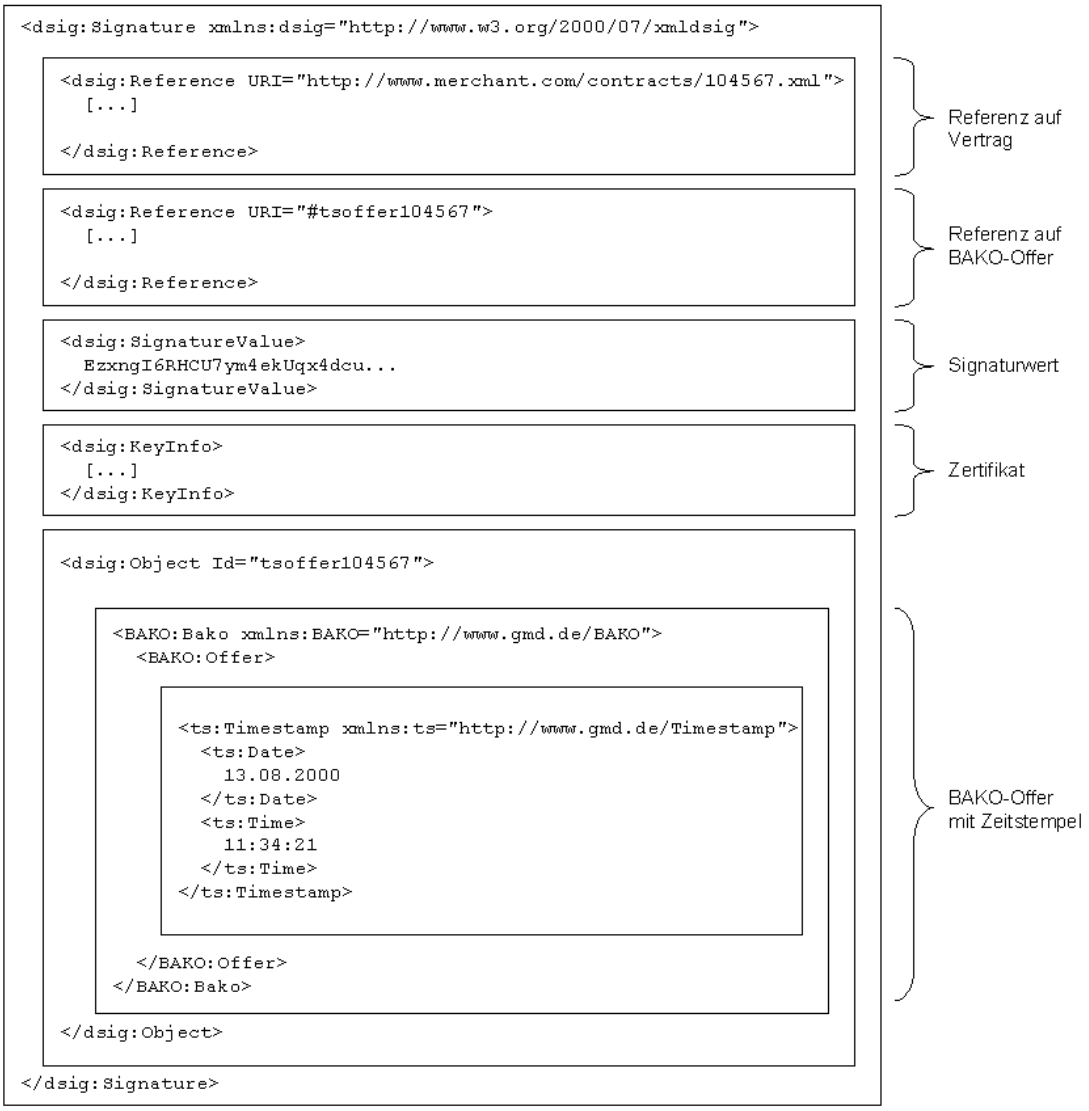

Abbildung 5.2: *Offer* - Dokument

8. *SignedInfo* wird mit dem angegebenen Algorithmus kanonisiert.

9. Der Signaturwert über dem kanonisierten *SignedInfo* - Element wird berechnet.

10. Das *KeyInfo* - Element, das das X.509 - Zertifikat der Händlers enthält, wird gebildet.

11. Das *Signature* - Element wird gebildet.

Das auf diese Weise erzeugte Angebot wird zusammen mit dem Vertrag an den Kunden gesendet. Wichtig ist, dass der Kunde sowohl das Dokument, das den Vertragstext enthält, als auch das *Offer* - Dokument mit der Signatur erhält. Nur beide Dokumente zusammen bilden ein verbindliches Angebot. Würde der Kunde nur den Vertragstext bekommen, wäre das Angebot wegen der fehlenden Signatur nicht verbindlich. Das *Offer* - Dokument allein nützt ihm nichts, da er in der Regel den Vertrag einsehen möchte, bevor er ihn akzeptiert.

Eine andere Möglichkeit wäre es, den Vertragstext ähnlich wie den Zeitstempel in das *Offer* - Dokument zu integrieren. Auf diese Weise würde der Kunde nur ein Dokument erhalten. Der Nachteil dieser Methode ist jedoch, dass in den folgenden beiden Phasen des BAKO - Protokolls jeweils der komplette Vertragstext mit geführt würde. Da Verträge unter Umständen sehr umfangreich sein können, würden die Dokumente unnötig groß werden.

Wurden dem Kunden die beiden Dokumente zugesendet, beginnt die zweite Phase des BAKO - Protokolls. Er wird zunächst die Signatur des *Offer* - Dokumentes prüfen. Dazu sind folgende Schritte auszuführen:

1. Es wird der Hashwert des Vertrags geprüft. Dazu wird auf den Vertrag der angegebene Transformationsalgorithmus angewendet und über dem transformierten Vertrag wird der Hashwert berechnet.

2. Der resultierende Hashwert wird mit dem Hashwert in dem *Reference* - Element verglichen.

3. Auf die gleiche Art und Weise wird der Hashwert des Zeitstempels geprüft.

4. Um die Signatur zu prüfen, wird das Public Key - Zertifikat aus dem *KeyInfo* - Element ausgelesen und auf seine Gültigkeit überprüft.

5. Das *SignedInfo*-Element wird mit dem angegebenen Algorithmus kanonisiert.

6. Mit dem gegebenen Signaturalgorithmus und dem öffentlichen Schlüssel des Händlers kann nun der Signaturwert überprüft werden.

7. Die Signatur ist in Ordnung, wenn sowohl alle Hashwerte als auch der Signaturwert erfolgreich validiert werden konnten.

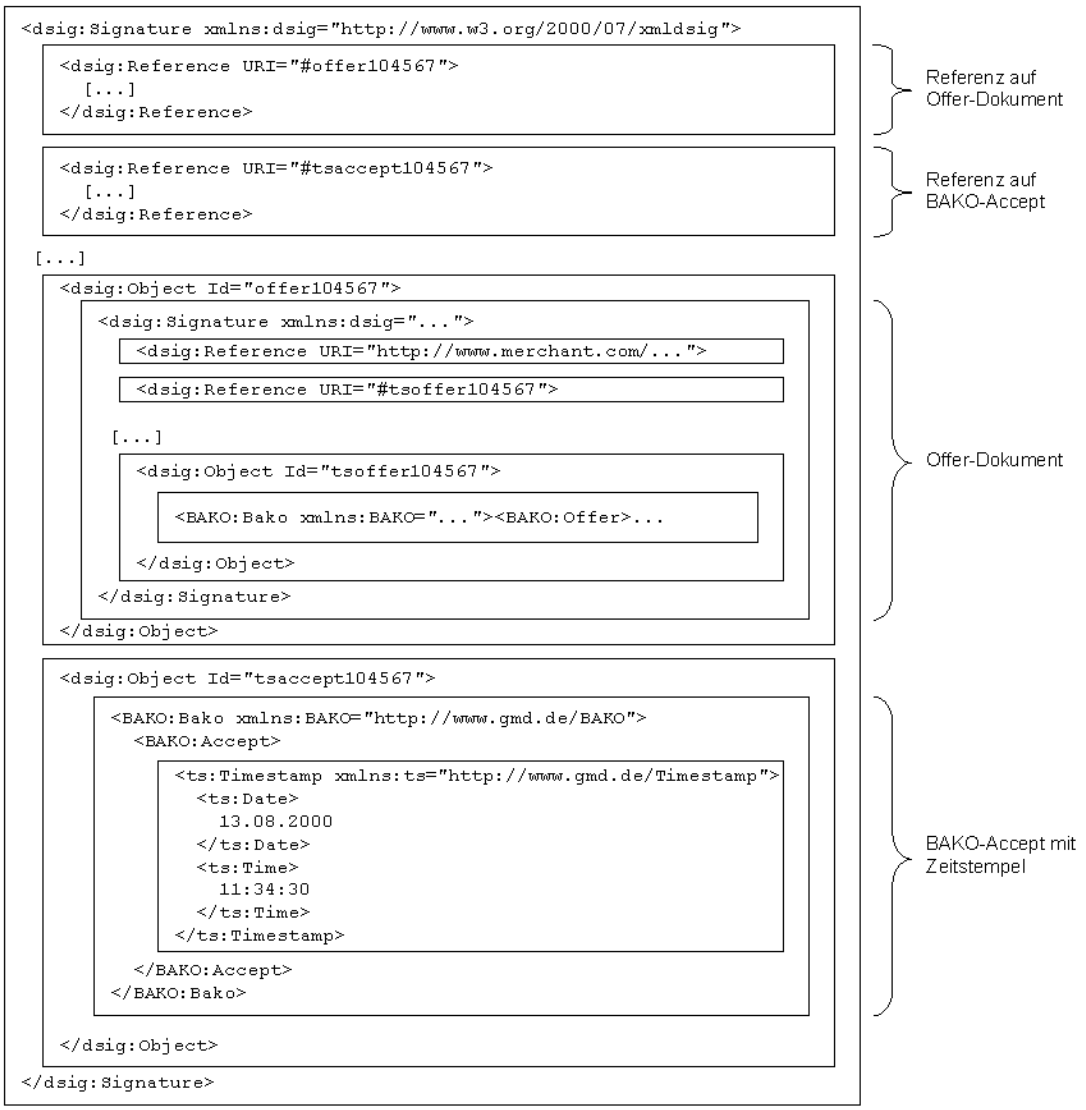

Abbildung 5.3: *Accept* - Dokument

Ist die Signatur in Ordnung, wird der Kunde die Integrität des Vertragstextes prüfen, bevor er das Angebot akzeptiert. Sofern er das Angebot akzeptiert, erzeugt der Kunde das *Accept* - Dokument (Abbildung 5.3) und sendet es an den Händler.

Interessant an diesem Dokument ist, dass es das komplette *Offer* - Dokument enthält. Die Schritte zum Generieren des *Accept* - Dokumentes unterscheiden sich nur unwesentlich von denen zum Erzeugen des *Offer* - Dokumentes, weswegen sie nicht nochmal im einzelnen aufgeführt werden.

Der einzige Unterschied ist, dass statt der Referenz auf den Vertrag eine Referenz auf das *Offer* - Dokument gebildet wird.

Die dritte und letzte Phase des BAKO - Protokolls beginnt damit, dass der Händler das *Accept* - Dokument prüft. Dies geschieht analog zur Prüfung des *Offer* - Dokuments. Wichtig ist, dass der Händler mit der oben beschriebenen Methode sowohl die Signatur des *Accept* - Dokuments, als auch die Signatur des darin enthaltenen *Offer* - Dokuments prüft, und dass er sich vergewissert, dass es sich bei dem *Offer* - Dokument um exakt dasselbe handelt, das er zuvor an den Kunden gesendet hat. Damit ist sichergestellt, dass der Kunde nicht das Angebot verändert hat. Würde der Händler nur die Signatur des *Accept* - Dokuments prüfen, wäre es für den Kunden möglich, ein von ihm verändertes *Offer* - Dokument in das *Accept* - Dokument zu integrieren, ohne dass es der Händler bemerken würde.

Ist das *Accept* - Dokument in Ordnung, erzeugt der Händler als letzten Schritt des BAKO - Protokolls die Bestätigung (Abbildung 5.4) und übermittelt sie an den Kunden.

Dieses *Confirm* - Dokument wird auf die gleiche Art und Weise gebildet, wie das *Accept* - Dokument. Anstatt des *Offer* - Dokuments wird das komplette *Accept* - Dokument integriert. Wenn der Kunde das *Confirm* - Dokument prüft, muss er alle drei Signaturen prüfen. Außerdem sollte er überprüfen, ob es sich bei dem enthaltenen *Accept* - Dokument um das von ihm generierte handelt.

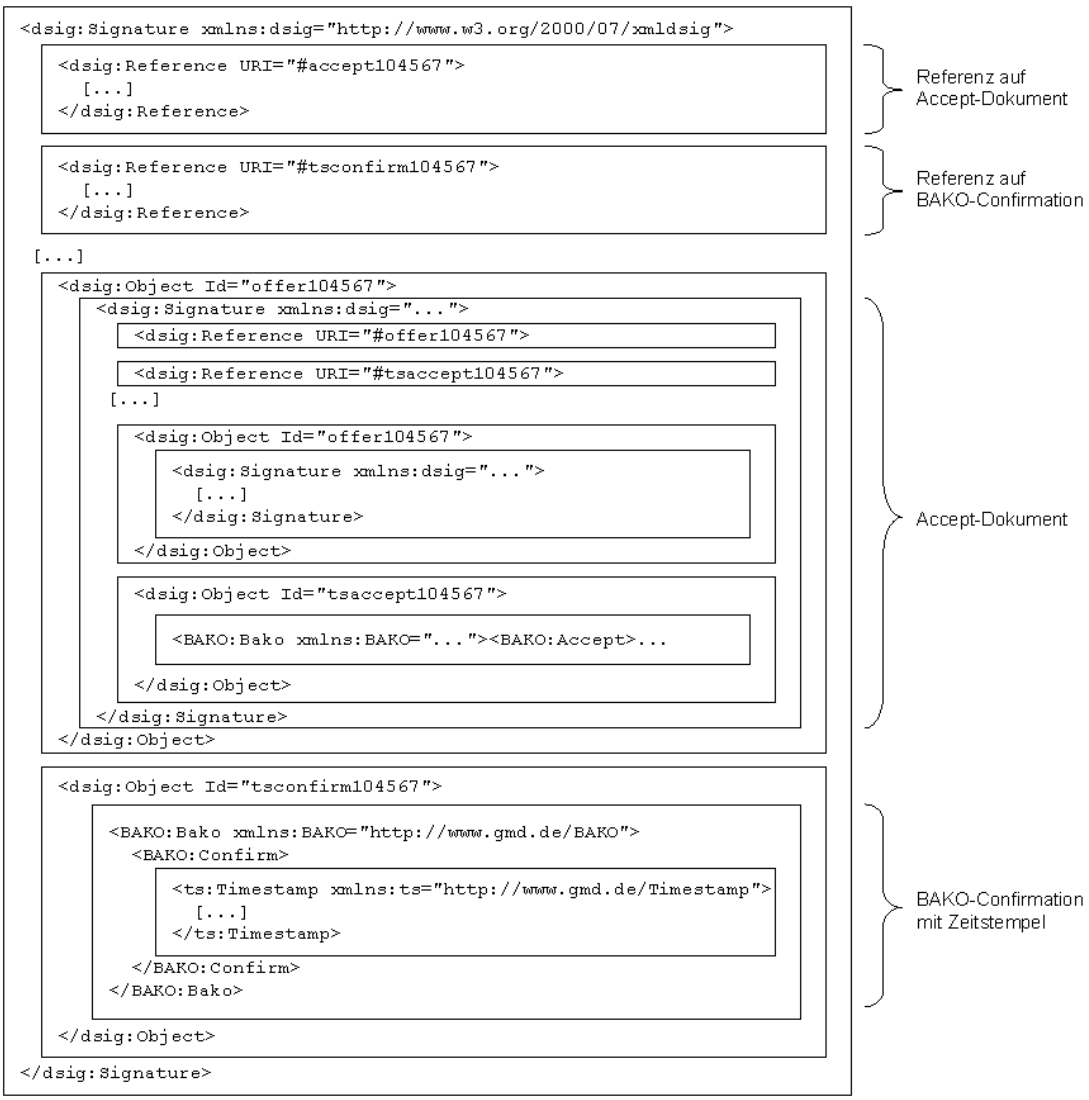

Abbildung 5.4: *Confirm* - Dokument

Kapitel 6

Darstellung von digitalen Signaturen

Digitale Signaturen sollen das Äquivalent zu einer handschriftlichen Unterschrift bezogen auf digitale Daten darstellen. Tatsächlich aber gibt es einige erhebliche Unterschiede zwischen digitalen Signaturen und Unterschriften auf Papier.

Einerseits leisten digitale Signaturen mehr als handschriftliche Unterschriften. Digitale Signaturen garantieren die Unveränderbarkeit der Daten. Ändert sich nämlich auch nur ein Bit der Daten, bricht die Signatur. Bei Unterschriften auf Papier ist es jedoch möglich, nach der Unterzeichnung Teile des Textes beispielsweise auszuradieren.

Andererseits ergeben sich jedoch Probleme bezüglich der Darstellung digitaler Signaturen gegenüber einer Unterschrift auf einem Papierdokument. Die Unterschrift auf dem Papier befindet sich bereits in einer für den Menschen lesbaren und wahrnehmbaren Form. Bei der digitalen Signatur handelt es sich jedoch nur um signierte Folgen von Bits. Wie diese signierten Bitfolgen und auch die digitale Signatur dem Anwender präsentiert werden sollen, ist zunächst noch völlig offen und zudem auch sehr variabel.

> *Erst eine Präsentation der signierten Rohdaten erzeugt, was bei einem unterschriebenen Papierdokument per se gegeben ist: den Wahrnehmungsgegenstand, d.h. die durch Menschen wahrzunehmende und zu interpretierende Willenserklärung. Und nicht die signierten Daten selbst sind das Beweismittel, sondern durch Präsentation werden aus den signierten Daten Beweismittel erzeugt.* (Pordesch, *[Por00]*)

Da stellt sich die Frage, ob digitale Signaturen überhaupt dieselbe Beweiskraft besitzen können, wie unterzeichnete Papierdokumente.

6.1 Präsentationsproblem

Das Präsentationsproblem bezeichnet den Fall, dass zwei Präsentationen ein und derselben signierten Daten so voneinander abweichen, so dass sie von Menschen unterschiedlich interpretiert werden. Die Variabilität der Darstellung signierter Daten kann verschiedene Ursachen haben.

Zunächst einmal kann die verwendete Hardware die Präsentation beeinflussen. Die Auflösung und die Größe des Monitors können Auswirkungen auf die Darstellung der Daten haben. Durch unterschiedliche Farb - oder Helligkeitseinstellungen können sich auf verschiedenen Rechnern unterschiedliche Darstellungen ein und desselben Dokuments ergeben. Durch ungünstige Farbkombinationen können Teile des Dokuments verdeckt bleiben. Sogar die zur Verfügung stehenden Eingabegeräte können Einfluss auf die Präsentation nehmen. Steht beispielsweise nur eine Tastatur zur Verfügung, kann es passieren, dass gerade bei Hypertext - Dokumenten Teile gar nicht oder nur sehr umständlich zugänglich sind.

Aber auch durch die Software kann es zu Problemen bei der Darstellung signierter Daten kommen. Viele Datenformate sind an spezielle Programme zur Präsentation bzw. Wiedergabe gekoppelt. Verwendet man ein ungeeignetes Programm, kommt es zu einer falschen und meist sogar unsinnigen Darstellung. Bei Word-Dokumenten können sich durch die Verwendung unterschiedlicher Formatvorlagen unterschiedliche Darstellungen ergeben. Durch die Verwendung von Stylesheets ist es möglich, absichtlich oder unabsichtlich, Teile eines Dokumentes zu verbergen. So wird zum Beispiel das „Kleingedruckte“eines Vertrages beim Kunden nicht angezeigt, aber er signiert es unwissentlich mit.

Signierte Daten können bei der Präsentation auf weitere, nicht signierte Daten, wie Zeichensätze, zugreifen. Werden diese unsignierten Daten manipuliert, wird die Ausgabe verfälscht. *[Por00]*

6.2 Darstellung von XML - Signaturen

Natürlich stellt sich das Präsentationsproblem auch bei XML - Signaturen. Anhand der in Kapitel 5.1 vorgestellten BAKO - Dokumente soll nun un-

tersucht werden, welchen Beitrag XML - Signaturen zur Lösung des Präsentationsproblems leisten können.

Bei der näheren Betrachtung der BAKO - Dokumente fallen zunächst die unterschiedlichen Namensräume auf. Das bedeutet, die BAKO - Dokumente vereinen mehrere XML - Anwendungen. Die eigentlichen XML - Signaturen sind an dem „dsig“ - Namesraum erkennbar. Daneben gibt es den „BAKO“ - Namensraum, durch den deutlich wird, dass das Dokument ein Teil des BAKO - Protokolls ist. Schließlich gibt es noch den Namensraum für den Zeitstempel („ts“), der eine eigene XML - Anwendung darstellt, die auch in anderen Zusammenhängen benutzt werden kann. Beide Namensräume, der „BAKO“ - Namensraum und der Zeitstempel - Namensraum, stellen einen Teil der signierten Daten dar. Handelt es sich bei dem Vertragstext ebenfalls um ein XML - Dokument, kommt mit Sicherheit noch wenigstens ein weiterer Namensraum hinzu. Durch die Verwendung der Namensräume werden verschiedene Anwendungskontexte voneinander getrennt, die Namensräume erfüllen somit also auch eine semantische Funktion.

Bei den XML - Signaturen ist noch eine weitere Auffälligkeit festzustellen. Die signierten Daten sind bei XML - Signaturen auf drei Arten von der Signatur getrennt.

- Syntaktische Trennung durch die Verwendung eines eigenen Namensraums und einer eigenen DTD.
- Eventuell eine physische Trennung durch die Verwendung externer Signaturen (*detached signatures*).
- Technische Trennung, da nur der Inhalt des *SignedInfo* - Elements tatsächlich signiert wird.

Bei XML - Signaturen lässt sich also zum einen die Syntax von der Darstellung trennen, zum anderen kann der Anwendungskontext vom Signaturkontext unterschieden werden. Somit ergeben sich vier klar trennbare, für XML - Signaturen relevante Komponenten.

- Die DTD der Signatur, vom W3C standardisiert, legt die Syntax der XML - Signatur fest.
- Die DTDs für die Anwendung, also die signierten Daten, legen dem entsprechend die Syntax der Anwendung fest.
- Ein Signatur - Stylesheet kann dazu dienen, die Signatur und die in ihr enthaltenen Informationen darzustellen.

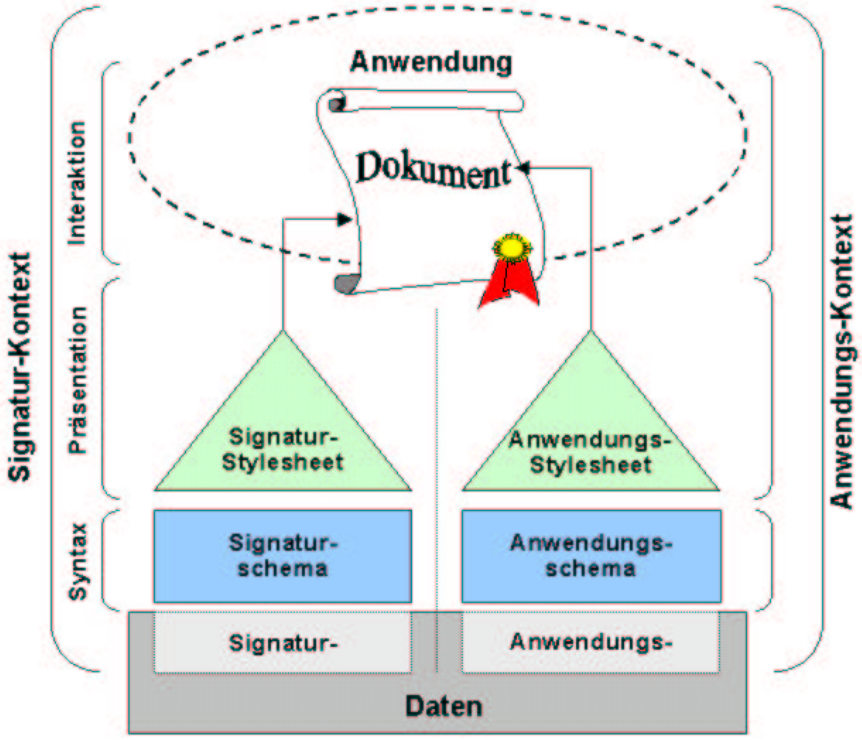

Abbildung 6.1: Kontext eines signierten XML-Dokuments

- Ein Stylesheet für die Anwendung kann dasselbe für die jeweilige Anwendung leisten.

Die verschiedenen Kontexte eines signierten XML - Dokuments sind in Abbildung 6.1[1] dargestellt. Diese strikte Trennung von Signatur und Anwendung macht Sinn. Denn grundsätzlich gilt, dass der Signierer für den Signaturkontext keine Verantwortung tragen kann. Die XML - Signaturen wurden von einer unabhängigen Arbeitsgruppe des W3C definiert, und stellen ein allgemein gültiges Signaturmedium dar. Dieses Signaturmedium verwendet der Unterzeichner zwar, ähnlich einem Blatt Papier, auf dem er mit einem Stift unterschreiben würde, aber er kann für die korrekte Funktionsweise dieses Mediums nur bedingt Verantwortung übernehmen. Anders sieht es dagegen beim Anwendungskontext aus. Die Anwendung kann vom Unterzeichner geschaffen werden, wodurch sich auch eine Verantwortung des Unterzeichners ergibt.

[1] übernommen aus: [Sch00]

Natürlich muss gewährleistet werden, dass das verwendete Signaturwerkzeug die vier Komponenten einer XML - Signatur auch tatsächlich verwendet, und nicht etwa auf eigene Stylesheets zurückgreift. Dazu muss zum einen sicher gestellt werden, dass die Komponenten einzeln zur Verfügung stehen. Zum anderen ist eine eindeutige Bindung der Komponenten an das jeweilige Dokument erforderlich, so dass aus dem Dokument ersichtlich ist, welche Komponenten zu verwenden sind. In XML lässt sich eine solche Bindung über sogenannte *Processing Instructions* (PI) ermöglichen. *Processing Instructions* sind spezielle Komponenten eines XML - Dokuments, die nicht zu den eigentlichen Elementen zählen, und Instruktionen für die Anwendungen, die das Dokument verarbeiten, enthalten. Daneben trägt diese explizite Benennung der Komponenten zu einer zusätzlichen Transparenz für den Anwender bei.

Eine weitere Anforderung ist, dass in der DTD keine Elemente definiert werden, die nicht von dem dazugehörigen Stylesheet dargestellt werden. Ist diese Anforderung erfüllt, spricht man von der Vollständigkeit der Darstellung. Der umgekehrte Fall sollte genauso gelten. Das Stylesheet sollte nicht viel mehr Bestandteile enthalten als es unterschiedliche Elemente in der DTD gibt. Damit wird eine Darstellungstreue erzielt.

Durch die Bindung der Signaturkomponenten an das Dokument wird, wie schon erwähnt, ein Bezug auf ein Signaturmedium hergestellt, für das der Unterzeichner keine Verantwortung übernehmen kann. Es sollte daher auch nicht mit signiert werden, sondern nur eindeutig benannt werden. Vertrauen in die Signaturkomponenten kann durch eine Standardisierung durch unabhängige, vertrauenswürdige Instanzen erzeugt werden. Für die DTD ist dies bereits durch das W3C erfüllt, für die Stylesheets müssen solche Standardisierungen noch geschaffen werden. Die Integrität dieser Komponenten kann durch Zertifikate gesichert werden.

Anders sieht es bei den Anwendungskomponenten aus. Da der Signierer durchaus Verantwortung für diese Komponenten tragen kann, würde es sich anbieten, sie nicht nur über eine PI zu benennen, sondern sie auch tatsächlich mit zu signieren. Standardisierte Komponenten wie für den Signaturkontext sind schon aufgrund der ungeheuren Vielfalt möglicher XML - Anwendungen ungeeignet. Eine Standardisierung würde die Möglichkeiten von XML zu stark einschränken. Denkbar sind jedoch zusätzliche standardisierte Stylesheets für eine möglichst große Zahl von Anwendungsbereichen, die gegenüber der vollen Darstellung eine vereinfachte, vollständige Präsentation bieten.

Abbildung 6.2: Darstellung einer XML - Signatur (Screenshot)

Vieles an dem vorgestellten Konzept der Trennung von Signatur und Anwendung ist noch hypothetisch. Insbesondere gibt es weder standardisierte Darstellungskomponenten für die Signatur, noch gibt es sie für die Anwendung. Das Konzept zeigt aber, dass es prinzipiell möglich ist, eine rechtliche Beweiskraft digitaler Signaturen mit Hilfe von XML zu erzielen.

Im Rahmen dieser Arbeit entstand ein auf XSLT und HTML basierter Lösungsansatz zur Darstellung von XML - Signaturen. Mit Hilfe von XSLT sollen die signierten BAKO - Dokumente zur Darstellung in HTML transformiert werden. Hierbei soll untersucht werden, in wie weit sich die beschriebene strikte Trennung von Signatur - und Anwendungskontext realisieren lässt.

Bei der Darstellung der signierten BAKO - Dokumente sollte folgendes beachtet werden.

- Die Darstellung sollte in einer Form erfolgen, die auch technisch ungeschulte Benutzer verstehen.
- Die Darstellung sollte der intuitiven Vorstellung einer Unterschrift mit Datum auf Papier folgen.
- Da in einem BAKO - Dokument bis zu drei Signaturen enthalten sind, muss klar erkennbar sein, auf welche Daten sich die Signaturen beziehen, und in welcher Beziehung die Signaturen zueinander stehen.

Da es sich bei den BAKO - Dokumenten um ineinander verschachtelte XML - Signaturen handelt, macht es Sinn, diese Verschachtelung auch auf dem Bildschirm darzustellen, zum Beispiel durch Einrückungen und durch nach innen kleiner werdende Schriftarten. Dies führt jedoch dazu, dass

das Signatur - Stylesheet nicht mehr universell einsetzbar ist, denn die Hervorhebung der Verschachtelungstiefen durch unterschiedliche Schriftgrößen und Farben muss explizit im Stylesheet angegeben werden. Das Stylesheet kann also nur noch Signaturen bis zu einer bestimmten Verschachtelungstiefe darstellen.

Bei der verschachtelten Darstellung der Signaturen sollte auch für einen Benutzer, der mit dem BAKO - Protokoll nicht vertraut ist, ersichtlich sein, zu welcher Phase des BAKO - Protokolls die einzelnen Signaturen gehören. Um dies zu erreichen, ist es erforderlich, den „BAKO“ - Namensraum mit in das Signatur - Stylesheet hinein zu ziehen. Um schließlich auch die intuitive Vorstellung einer Unterschrift mit Datum auf dem Bildschirm zu realisieren, muss der Zeitstempel bei der dazugehörenden Signatur dargestellt werden. Dadurch wird auch die Zeitstempel - Anwendung in das Signatur - Stylesheet hineingezogen.

Somit ist es nur schwer möglich, die klare Abtrennung des Signaturkontextes von der Anwendung bei der Darstellung auch tatsächlich umzusetzen. Insbesondere ergeben sich dadurch keine universell einsetzbaren Signatur - Stylesheets. Andererseits stellt die Einbeziehung der BAKO - und Zeitstempel - Anwendung in den Signaturkontext keine große Verletzung des Konzepts der klaren Trennung von Signatur und Anwendung dar. Der Hintergrund dieser Trennung ist in erster Linie die Verteilung von Verantwortlichkeiten. Die BAKO - und die Zeitstempel - Anwendung stellen genauso wie die XML - Signatur lediglich ein Medium dar, das der Unterzeichner zwar anwendet, das er aber nicht definiert hat. Somit kann er auch für diese beiden Anwendungen keine Verantwortung übernehmen. Diese beiden Anwendungen lassen sich wie die Komponenten der XML - Signatur von unabhängigen Instanzen definieren. Das BAKO - Protokoll und der Zeitstempel stellen nicht die eigentlichen Daten dar, die der Unterzeichner signieren möchte, sondern sie sind für ihn ein Mittel, um gegenseitige Nichtabstreitbarkeit zu erzielen. Die eigentlichen Daten, die er signieren möchte, sind in dem Vertragstext enthalten, und der lässt sich von den übrigen Anwendungen trennen.

Ein weiteres Problem bei der Darstellung von XML - Signaturen ist die Forderung, alle signierten Daten und alle in der Signatur enthaltenen Informationen darzustellen. Wie in Kapitel 4 beschrieben, bezieht sich der Signaturwert auf das in der XML - Signatur enthaltene *SignedInfo* - Element. Streng genommen müsste also auch und gerade dieses Element dargestellt werden. Es ist allerdings fraglich, ob es einem durchschnittlichen

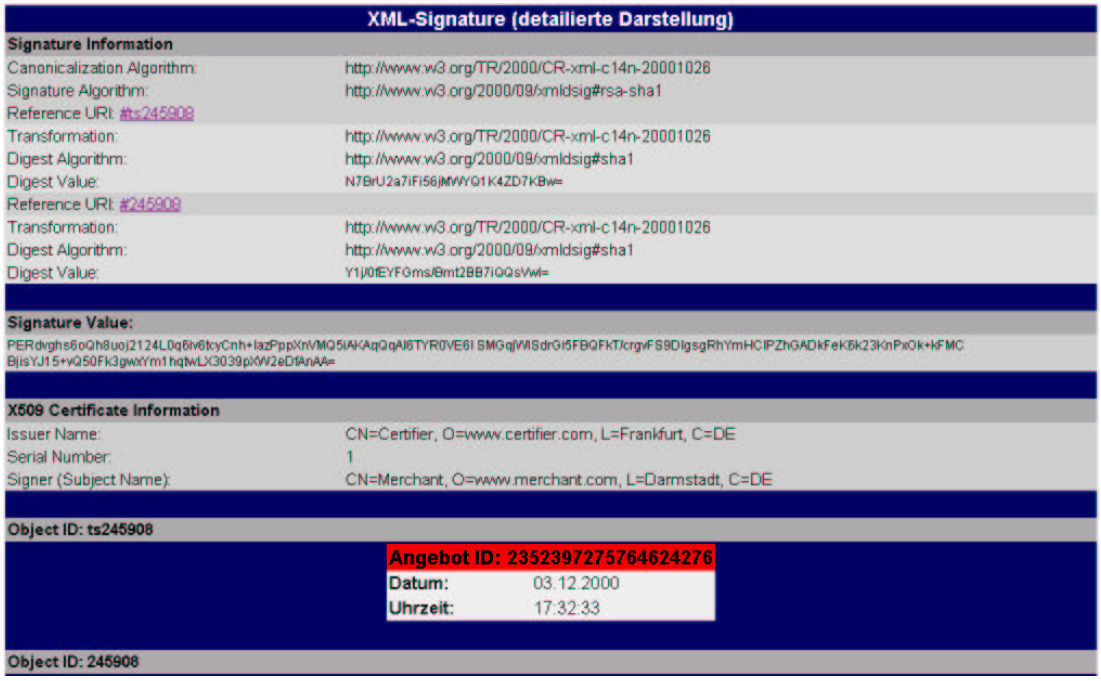

Abbildung 6.3: Detaillierte Darstellung (Screenshot)

Benutzer einen Nutzen bringt, die Bezeichnungen verschiedener verwendeter Algorithmen, die er wahrscheinlich nicht einmal kennt, angezeigt zu bekommen, oder ob es ihn nicht sogar eher verwirrt.

Eine Lösung hierzu, die in dieser Arbeit verfolgt wurde, ist die Bereitstellung zweier Stylesheets. Das eine Stylesheet stellt die komplette XML - Signatur mit allen Informationen dar (Abbildung 6.3). Das andere Stylesheet dagegen präsentiert nur die für den Benutzer relevanten Daten, das Datum und den Namen des Unterzeichners, was der Unterschrift auf Papier am nächsten kommt (Abbildung 6.2). Der Benutzer hat die Möglichkeit, zwischen diesen beiden Darstellungsformen hin - und herzuschalten. Im übrigen demonstriert dies auch die Mächtigkeit der XSLT - Transformationen. Ein und dasselbe XML - Dokument wird auf zwei völlig verschiedene Arten dargestellt, insbesondere was die Menge der dargestellten Informationen betrifft. *[Sch00]*

6.3 Bewertung gemäß Signaturgesetz

1997 traten das deutsche Signaturgesetz (SigG, *[SigG97]*) und die dazugehörige Signaturverordnung (SigV, *[SigV97]*) in Kraft. Das Signaturgesetz

zusammen mit der Signaturverordnung stellen ein sehr ambitioniertes Regelwerk dar, das sehr hohe Anforderungen an Zertifizierungsstellen, aber auch beispielsweise an Hersteller von Chipkarten - Lesegeräte stellt. Dabei nutzt der Gesetzgeber seine Regulierungsmöglichkeiten voll aus. Es gibt sogar Stimmen, die sagen, dass gerade diese strengen Regulierungen und Auflagen, die der Gesetzgeber stellt, mit ein Grund dafür sind, warum der Durchbruch in der Verwendung digitaler Signaturen bis heute ausgeblieben ist.

Natürlich bietet das Signaturgesetz mit seinen hohen Auflagen ein hohes Sicherheitsversprechen. Wer digitale Signaturen gemäß Signaturgesetz nutzt, hat nach § 1 Abs.1 SigG die „Sicherheitsvermutung“ für sich. Das bedeutet, im Falle eines Rechtsstreits kommt er in den Genuss einer Beweiserleichterung, vergleichbar mit der Rechtssicherheit, die herkömmliche Schrifturkunden bieten.

Jedoch beschäftigt sich das Signaturgesetz überwiegend mit Auflagen, die Zertifizierungsstellen zu erfüllen haben, und Anforderungen an Public Key Zertifikate. Anforderungen an die Darstellung digitaler Signaturen werden nur am Rande gestellt. Die Darstellung digitaler Signaturen wird lediglich in § 14 Abs.2 SigG und § 16 Abs.2 SigV angesprochen.

§ 14 (2) SigG:

> *Für die Darstellung zu signierender Daten sind technische Komponenten mit Sicherheitsvorkehrungen erforderlich, die die Erzeugung einer digitalen Signatur vorher eindeutig anzeigen und feststellen lassen, auf welche Daten die digitale Signatur sich bezieht. Für die Überprüfung signierter Daten sind technische Komponenten mit Sicherheitsvorkehrungen erforderlich, die feststellen lassen, ob die signierten Daten unverändert sind, auf welche Daten die digitale Signatur sich bezieht und welchem Signaturschlüssel - Inhaber die digitale Signatur zuzuordnen ist.*

§ 16 (3) SigV:

> *Die zum Darstellen zu signierender Daten erforderlichen technischen Komponenten müssen so beschaffen sein, dass die signierende Person die Daten, auf die sich die Signatur erstrecken soll, eindeutig bestimmen kann, eine digitale Signatur nur auf ihre Veranlassung erfolgt und diese vorher eindeutig angezeigt wird. Die zum Prüfen signierter Daten erforderlichen technischen Komponenten müssen so beschaffen sein, dass die prüfende Person die*

> *Daten, auf die sich die digitale Signatur erstreckt, sowie den Signaturschlüssel - Inhaber eindeutig feststellen kann und die Korrektheit der digitalen Signatur zuverlässig geprüft und zutreffend angezeigt wird. Die technischen Komponenten zum Nachprüfen von Zertifikaten müssen eindeutig erkennen lassen, ob die nachgeprüften Zertifikate im Verzeichnis der Zertifikate zu einem angegebenen Zeitpunkt vorhanden und nicht gesperrt waren. Die technischen Komponenten müssen nach Bedarf den Inhalt der zu signierenden oder signierten Daten hinreichend erkennen lassen. Werden technische Komponenten nach den Sätzen 1 bis 4 geschäftsmäßig Dritten zur Nutzung angeboten, muss die eindeutige Interpretation der Daten sichergestellt sein und müssen die technischen Komponenten bei Benutzung automatisch auf ihre Echtheit überprüft werden. Sicherheitstechnische Veränderungen an den technischen Komponenten müssen für den Nutzer erkennbar werden.*

Man sieht, eine ausdrückliche Regelung der Präsentation ist nicht gegeben. Zudem werfen das Signaturgesetz und die Signaturverordnung einige Fragen auf. Es fehlt eine Abgrenzung der Begriffe „Daten“ und „Inhalt von Daten“ . Zudem fehlt eine klare Definition des Begriffs „Daten“. Unter diesem Begriff kann man alles verstehen, von einem Dateinamen oder einer URL bis hin zu einer kompletten Auflistung der Daten. Warum muss sich der Inhalt der zu signierenden Daten nur bei Bedarf hinreichend erkennen lassen? Und wieso muss eine eindeutige Interpretation der Daten nur bei geschäftsmäßiger Nutzung durch Dritte sichergestellt sein?

Trotzdem soll im folgenden versucht werden, die bereits vorgestellten Stylesheets zur Darstellung von XML - Signaturen an den Anforderungen der Signaturgesetzes zu messen.

Die im Gesetzestext erwähnten *technischen Komponenten* zur Darstellung sind in diesem Fall zunächst einmal die XSLT - Stylesheets. Aber auch der zur Darstellung verwendete Browser und letztendlich auch die bereitgestellte Hardware sind dazu zuzählen.

Durch die wie in Abbildung 6.2 zu sehende intuitive Darstellung der Signatur und der Daten ist eine eindeutige Interpretation der Daten gegeben. Da sowohl die Signatur als auch die signierten Daten in einem Fenster angezeigt werden, ist für den Benutzer ersichtlich, auf welche Daten sich die Signatur erstreckt. Zudem werden in der detaillierten Darstellung alle Referenzen auf signierte Daten klar aufgeschlüsselt. Allerdings ist

diese Darstellungsform auf Grund ihrer relativ technischen Präsentation für den Laien weniger geeignet. Es ist zudem zu bedenken, dass bei der Verwendung sehr vieler Referenzen oder auch bei der Verwendung von *detached signatures* eine übersichtliche Darstellung von Signatur und Daten in einem Fenster unter Umständen nicht mehr möglich ist. Ein ähnliches Problem kann sich ergeben, wenn beliebige binäre Daten signiert wurden, für deren Darstellung besondere Software zur Verfügung stehen muss.

Der Inhaber des Schlüssels wird von dem Stylesheet angezeigt. Da sich die Darstellung an der gewohnten Darstellung einer Unterschrift auf Papier orientiert, ist der Name des Schlüssel - Inhabers für den Benutzer auch sofort als solcher zu erkennen. Daneben bietet die detaillierte Ansicht zusätzliche Informationen über des Zertifikat des Schlüssel - Inhabers.

Das Ergebnis der Überprüfung der Signatur wird intuitiv durch ein Ampelsymbol dargestellt. Ein grünes Ampelsymbol zeigt an, dass die Signatur in Ordnung ist, ein rotes Ampelsymbol signalisiert, dass die Signatur nicht korrekt ist. In der detaillierten Ansicht werden zudem die Teilergebnisse der Überprüfung im einzelnen aufgeschlüsselt.

Die automatische Überprüfung der Echtheit der verwendeten Stylesheets lässt sich durch die Einbeziehung einer vertrauenswürdigen Instanz, von der die Stylesheets bezogen werden, lösen. Die Stylesheets werden von dieser Instanz zertifiziert und signiert, so dass sich ihre Echtheit überprüfen lässt.

Damit ist gezeigt, dass XML als universelles Datenformat und XSLT - Stylesheets als Darstellungskomponente prinzipiell dazu geeignet sind, die Auflagen des Signaturgesetzes zur Darstellung digitaler Signaturen zu erfüllen. Das Präsentationsproblem ist damit jedoch noch nicht vollständig gelöst, auch wenn man der Lösung ein ganzes Stück näher gekommen ist. Das Problem besteht nach wie vor darin, dass neben den Stylesheets noch weitere Faktoren die Darstellung beeinflussen. Zur Darstellung sind Browser erforderlich. Die Browser von Microsoft und Netscape weisen jedoch erheblich Unterschiede bezüglich der Darstellung auf, man denke nur an die Probleme bei der Darstellung von Tabellen. Streng genommen müsste man, genauso wie man zertifizierte Stylesheets verwendet, auch spezielle zertifizierte Viewer entwickeln. Und selbst dann bestehen immer noch die Probleme, die die verwendete Hardware verursachen kann. *[Sch00][SH00]*

Kapitel 7

Elektronische Auktionen

Die Zahl elektronischer Märkte im Internet ist in den letzten Jahren stark angestiegen. Neben katalogbasiertem Handel spielen dabei in immer größerem Maße auch andere Handelsformen, insbesondere Auktionen, eine Rolle. Neben dem Online - Shopping stellen Auktionen auch heute schon die am weitesten verbreitete Handelsform im Internet dar. Es wird erwartet, dass bis zum Jahr 2002 mit elektronischen Auktionen Umsätze von über 52 Milliarden US - Dollar erzielt werden[1]. Elektronische Auktionen finden im *Business to Business* Bereich (B2B), im *Business to Consumer* Bereich (B2C), als auch im *Consumer to Consumer* Bereich (C2C) Anwendung.

Ein Beispiel für elektronische Auktionen im B2B - Bereich sind die im letzten Jahr in Europa durchgeführten Versteigerungen der UMTS - Lizenzen. Stellvertretend für den Erfolg von Auktionen im Consumer - Bereich steht das 1995 gegründete und damit älteste amerikanische Online - Auktionshaus *eBay*[2]. Die Zahl der registrierten Nutzer von eBay stieg bis Ende 2000 auf 22,5 Millionen. Allein im vierten Quartal 2000 fanden bei eBay über 79 Millionen Auktionen statt. Der Umsatz betrug im Jahr 2000 431,4 Millionen US - Dollar, was einer Steigerung gegenüber dem Vorjahr von 92 Prozent entspricht[3].

Einer Studie von Forrester Research zufolge machen gegenwärtig die Geschäfte im C2C - Bereich rund zwei Drittel des Umsatzes der Online - Auktionen aus. Dies könnte sich jedoch in Zukunft ändern. Schon bald könnten die Transaktionen im B2B - Bereich den Hauptanteil am Umsatz der Online - Auktionen ausmachen. Insbesondere die kleineren und mittleren

[1]Quelle: Forrester Research, 1999 (http://www.forrester.com)
[2]http://www.ebay.com
[3]Quelle: Heise News-Ticker (http://www.heise.de/newsticker/data/mbb-19.01.01-000/)

Unternehmen stellen laut der Studie die wichtigste Zielgruppe der Zukunft für die Auktionshäuser dar. Für Produkte wie Büroartikel oder Firmenwagen könnten Online - Auktionen eine alternative Beschaffungsquelle für diese Unternehmen darstellen. Durch zusätzliche Dienstleistungen wie das Einräumen von Krediten könnten die Auktionshäuser ihre Attraktivität noch erhöhen[4]. *[DAP01]*

7.1 Eigenschaften

Die folgenden Betrachtungen beziehen sich hauptsächlich auf Online - Auktionen als besondere Form elektronischer Auktionen, also Auktionen, die im Internet angeboten werden. Online - Auktionen stellen eine Erweiterung traditioneller Auktionen dar, die sich die besonderen Eigenschaften des Internets, wie etwa die weltweite Erreichbarkeit, zu Nutze machen. Daneben bringt das Internet neue Arten und Einsatzbereiche für Auktionen hervor.

Die Vorteile von Online - Auktionen gegenüber traditionellen Offline - Auktionen liegen auf der Hand. Online - Auktionen können wesentlich kostengünstiger durchgeführt werden, da keine besonderen Räume für die Durchführung einer Auktion angemietet werden müssen, und die Auktionen in der Regel nicht im großen Rahmen angekündigt werden müssen.

Aber auch für die Kunden bieten Online - Auktionen Kostenvorteile. Da Gebote über das Internet abgegeben werden, entstehen beispielsweise keine Anfahrtskosten zu einem Auktionshaus. Trotzdem entstehen für den Kunden Kosten durch die Teilnahme an einer Online - Auktion, auch wenn diese deutlich geringer ausfallen, als bei klassischen Offline - Auktionen. Es entstehen Kosten, wenn der Kunde den Verlauf einer Auktion beobachtet (*Monitoring Costs*). Wenn er auf die Versteigerung eines bestimmten Produktes wartet, können Kosten entstehen, da er in regelmäßigen Abständen die Kataloge verschiedener Auktionshäuser abfragen wird, bis er das gesuchte Produkt gefunden hat. Aber auch wenn der Kunde kein bestimmtes Produkt sucht, wird er die Kataloge der Auktionshäuser durchstöbern. Eventuell wird er nach alternativen Anbietern suchen müssen, falls er überboten wurde, wodurch auch Kosten entstehen (*Delay and Search Costs*). Zudem entstehen in der Regel Kosten durch die Lieferung der Waren, während bei Offline - Auktionen die Waren möglicherweise selbst abgeholt werden können.

[4]Quelle: Heise News-Ticker (http://www.heise.de/newsticker/data/pmz/14.02.01-001/)

Für Verkäufer ist es einfacher, Auktionen anzubieten, da es inzwischen eine Vielzahl von Online - Auktionsplattformen gibt, die für jedermann offen sind. Auf Grund der spezifischen Eigenschaften des Internets erreichen Online - Auktionen eine wesentlich größere Zahl potentieller Käufer. Für Hersteller eröffnen sich durch Online - Auktionen neue, zusätzliche Vertriebswege. Da Online - Auktionen in der Regel über mehrere Tage hinweg durchgeführt werden, haben Käufer länger Zeit, Entscheidungen zu treffen, bevor sie ein Gebot abgeben. Traditionelle Auktionen sind dagegen oft schon nach wenigen Minuten beendet. Dies führt dazu, dass Käufer entweder kein Gebot abgeben, oder aber ohne nachzudenken bieten und am Ende einen zu hohen Preis zahlen.

Aber natürlich haben Online - Auktionen nicht nur Vorteile. Online - Auktionen bieten oft nur unzureichende Informationen über die angebotenen Produkte. Der Kunde kann sich die Objekte nicht live ansehen, sondern muss sich auf die Beschreibung im Netz verlassen. Die Atmosphäre, die traditionelle Auktionen bieten, und Aspekte wie Spaß am Bieten und am Treffen schneller Entscheidungen können nur begrenzt auf Online - Auktionen übertragen werden. Der größte Nachteil von Online - Auktionen sind jedoch die hohen Sicherheitsanforderungen, die von bestehenden Online - Auktionen oft nur in Ansätzen erfüllt werden. Gerade der Aspekt der Rechtssicherheit wird gegenwärtig von Online - Auktionen stark vernachlässigt. Stattdessen beruhen Online - Auktionen auf Vertrauen gegenüber den beteiligten Parteien, was allerdings bei der per se gegebenen Anonymität von Online - Auktionen eher bedenklich ist.

Es zeigt sich, dass Online - Auktionen besonders gut für Produkte mit bestimmten Eigenschaften geeignet sind. Die angebotenen Produkte sollten sich leicht mit den Möglichkeiten, die das Internet, insbesondere das WWW, bietet, beschreiben lassen. Online - Auktionen bieten sich an, wenn der Wert eines Produkts so niedrig ist, dass sich die Durchführung einer kostspieligen traditionellen Offline - Auktion nicht lohnt. Auch für die Versteigerung von Informationen sind Online - Auktionen geeignet. Bei Versteigerungen von normalen Gütern dagegen können unter Umständen erhebliche Transportkosten entstehen, da Auktionator und Kunde weit voneinander entfernt sein können. *[Tur97][Wri97][Klm00]*

7.2 Klassifikationen

Auktionen können verschiedenen Zwecken dienen. Auktionen bilden einen Marktplatz für Waren, die nicht auf üblichen Wegen gehandelt werden kön-

nen, wie zum Beispiel seltene Kunstwerke oder auch Telekommunikationsfrequenzen.

Desweiteren können Auktionen einen zusätzlichen Distributionskanal neben herkömmlichen Vertriebswegen darstellen. Diese Art von Auktionen richten sich häufig an „Schnäppchenjäger“. Sie dienen auch dazu, den Bekanntheitsgrad der Produkte zu steigern, und neue Kunden zu gewinnen. Beispiele hierfür sind die Versteigerungen von Flugtickets bei Fluggesellschaften wie etwa der Lufthansa[5], oder die Versteigerung von Auslaufmodellen, wie sie gerne von Computerherstellern angeboten werden[6].

Man verwendet Auktionen, um knappe Ressourcen anzubieten. Durch die Auktion wird dynamisch der Preis ermittelt, der dann gewissermaßen als Maß für die Knappheit der Ressource angesehen werden kann.

Außerdem kommen Auktionen zum Einsatz, wenn Waren verkauft werden sollen, die auf Grund ihrer begrenzten Haltbarkeit nicht lange gelagert werden können. Diese Güter müssen schnell verkauft werden, auch zu niedrigeren Preisen. Als Beispiele hierfür dienen Fischauktionen oder die holländischen Blumenauktionen.

Auktionen werden hauptsächlich auf Grund der verwendeten Regeln zur Preisbestimmung unterschieden. Die wichtigsten Arten von Auktionen, die auch im Offline - Bereich eingesetzt werden, sind *Englische Auktionen*, *Holländische Auktionen* und *Versiegelte Auktionen*. Daneben gibt es weitere Arten und Variationen, die zum Teil speziell für elektronische Auktionen entwickelt wurden.

Englische Auktion. Die sogenannte Englische Auktion (*English Auction*) ist zweifellos der bekannteste Auktionstyp. Bei diesem Typ wird vom Verkäufer ein Startgebot festgelegt. Die Bieter geben nun Gebote ab, die über diesem Startgebot liegen, und sorgen dafür, dass der Preis steigt. Bei manchen Auktionen können die Bieter die Höhe ihres Gebots frei bestimmen, in der Regel jedoch erhöhen sich die Gebote in festgelegten Schritten. Die abgegebenen Gebote werden veröffentlicht. Dadurch haben die Bieter die Möglichkeit, den Auktionsverlauf zu beobachten und weitere Gebote abzugeben, falls sie überboten wurden. Offline - Auktionen enden meist dann, wenn über einen zuvor festgelegten Zeitraum hinweg kein weiteres

[5] http://www.lufthansa.com
[6] Beispiel: Hewlett-Packard (http://www.hewlett-packard.com)

Gebot abgegeben wurde. Bei Online - Auktionen wird in der Regel ein Zeitpunkt festgelegt, zu dem die Auktion beendet wird. Nach der Beendigung der Auktion wird der Gewinner ermittelt. Dies ist der Bieter, der das höchste Gebot abgegeben hat. Beispiele für Englische Auktionen, die zu einem festgelegten Zeitpunkt enden, sind die Auktionen, die bei Ebay angeboten werden. Das Online - Auktionshaus Ricardo[7] bietet zudem Live - Auktionen nach demselben Prinzip an, die nur wenige Minuten dauern, und die noch stärker die Atmosphäre von Offline - Auktionen nachbilden sollen. Diese Auktionen werden als besonderes Event angekündigt und werden oft von bekannten Moderatoren im Chat moderiert.

Holländische Auktion. Holländische Auktionen (*Dutch Auctions*) starten bei einem zuvor festgesetzten Höchstpreis. Dieser Preis wird in festgelegten Zeitintervallen schrittweise erniedrigt. Dies geschieht solange, bis ein Bieter den aktuellen Preis akzeptiert und damit die Auktion abbricht. Dieser Bieter erhält den Zuschlag und gewinnt die Auktion. Während der gesamten Auktion wird also nur ein einziges Gebot abgegeben. Holländische Auktionen werden eher selten im Consumer - Bereich angewendet. Ein Beispiel hierfür ist das Online - Auktionshaus Klik - Klok[8]. Ihre Hauptanwendung haben diese Auktionen im Business - Bereich bei der Versteigerung leicht vergänglicher Waren, wie etwa Fischauktionen. Seinen Ursprung hat dieser Auktionstyp in den Niederlanden, wo nach diesem Prinzip Blumen versteigert werden (daher auch die Bezeichnung *Holländische Auktion*).

Versiegelte Auktion. Bei Versiegelten Auktionen (*Sealed Bid Auctions*) werden, ausgehend von einem zu Beginn vom Verkäufer festgelegten Mindestpreis, die Gebote der Interessenten verdeckt übermittelt. Die Auktion endet zu einem festgesetzten Zeitpunkt. Erst dann öffnet der Auktionator die eingegangenen Gebote und bestimmt den Gewinner. Das bedeutet, während der Auktion werden die Gebote nicht geöffnet und daher auch nicht veröffentlicht. Die Bieter haben also keine Möglichkeit, den Verlauf der Auktion zu beobachten. Dem zu Folge gibt jeder Bieter auch nur ein Gebot ab, er kann sein Gebot nicht nachbessern, falls er überboten wurde. Der Gewinner der Auktion ist derjenige Bieter, der das höchste Gebot abgegeben hat, und wird nach Beendigung der Auktion bekannt gegeben. Sollten mehrere Bieter dasselbe Höchstgebot abgegeben haben, gewinnt das Gebot, das zuerst abgegeben wurde. Versiegelte Auktionen werden überwiegend im Business - Bereich eingesetzt, beispielsweise bei der Versteigerung von Grundstücken.

[7] http://www.ricardo.de
[8] http://www.klik-klok.com

Double Auction. Bei *Double Auctions* geben sowohl Käufer, als auch Verkäufer Gebote ab. Mehrere Verkäufer geben Verkaufsangbote zu einem Produkt ab, die Käufer geben Kaufgebote ab. Zueinander passende Kauf- und Verkaufangebote werden zugeordnet, wodurch Geschäftsabschlüsse zwischen einem Käufer und einem Verkäufer entstehen. *Double Auctions* finden beispielsweise im Aktienhandel oder auf Rohstoffmärkten Anwendung.

Japanese Auction. *Japanese Auctions* sind eng mit den Englischen Auktionen verwandt. Auch hier wird zuvor ein Mindestpreis festgelegt. Interessenten können sich zur Auktion anmelden. Nach Auktionsstart steigt der Verkaufspreis automatisch schrittweise an. Die Teilnehmer der Auktion steigen nach und nach aus, wenn ihnen der Preis zu hoch erscheint. Die Auktion läuft solange, bis nur noch ein Teilnehmer übrig bleibt. Dieser Teilnehmer gewinnt die Auktion. *Japanese Auctions* sind im Grunde eine Variante der Englischen Auktionen, die speziell für den Online - Bereich entwickelt wurden. Sie lassen sich in noch stärkerem Masse automatisiert durchführen und sind in der Regel schneller als Englische Auktionen.

Quick - Win Auction. Bei *Quick - Win Auctions* bietet der Verkäufer ein Produkt zur Versteigerung an, indem er einen minimalen Preis festsetzt, zu dem er den Verkauf akzeptiert. Wenn sich ein Käufer findet, der bereit ist, die Ware zu diesem Preis zu kaufen, erhält er auch sofort den Zuschlag. Das Produkt ist also verkauft und die Auktion beendet. Allerdings fehlt diesem Auktionstyp die Dynamik, die Auktionen im allgemeinen auszeichnet, und es stellt sich die Frage, in wie weit man bei diesem Typ noch von einer Auktion reden kann. Ebenso wie *Japanese Auctions* sind *Quick - Win Auctions* speziell für den Online - Bereich entstanden.

Eine Untersuchung ergab, dass es sich bei 42 Prozent aller angebotenen Online - Auktionen um Englische Auktionen handelt. Holländische Auktionen erreichen immerhin noch einen Anteil von gut 11 Prozent, alle anderen Arten von Auktionen liegen unter fünf Prozent. Englische Auktionen kommen sowohl auf consumer - orientierten Sites, als auch bei reinen B2B - Auktionen zum Einsatz, der Schwerpunkt liegt jedoch auf dem Consumer - Bereich. *Japanese Auctions* und *Quick - Win Auctions* sind reine Consumer - Auktionen, die zudem speziell für den Online - Markt entwickelt wurden. Im Business - Bereich spielen diese Auktionstypen keine Rolle. Holländische Auktionen, Versiegelte Auktionen und *Double Auctions* dagegen stellen fast ausschließlich Geschäftsmodelle für den B2B - Bereich dar.

Ein Grund für diese Verteilung ist der Faktor „Unterhaltung“. Bei consumer - orientierten Auktionen spielt Unterhaltung eine große Rolle, was schon am Design dieser Sites deutlich wird. Englische Auktionen bieten im Vergleich zu Holländischen Auktionen oder Versiegelten Auktionen mehr Spaß und Unterhaltung, da man sich direkter und aktiver mit anderen Bietern im Wettbewerb befindet und man den Verlauf der Auktion mitverfolgen kann. *Japanese* und *Quick - Win Auctions* eignen sich für den Consumer - Bereich, da sie sehr schnell sind, und daher Online - Zeit bei dem Kunden sparen. *[DAP01][Kle00][HZ00][Rec97]*

Kapitel 8

Implementierung einer Online - Auktion

In jüngster Zeit entstehen im Internet immer mehr Online - Auktionen. Gerade im Consumer - Bereich stellen Auktionen das beliebteste Geschäftsmodell hinter dem katalogbasierten Online - Shopping dar, was nicht zuletzt an dem Unterhaltungswert liegt, den Online - Auktionen bieten. Wie schon in Kapitel 7 ausführlich dargelegt, bieten Online - Auktionen daneben noch weitere Vorteile für Kunden und Verkäufer, wie etwa Kostenersparnisse und weltweite Erreichbarkeit. Online - Auktionen stellen jedoch hohe Anforderungen an die Sicherheit, wenn es darum geht, Rechtssicherheit für die beteiligten Parteien zu erzielen, und die Auktion vor Angriffen und Manipulationen zu schützen. Derzeitige Online - Auktionen erfüllen diese Anforderungen allerdings nur in Ansätzen.

Vor diesem Hintergrund wurde vom Verfasser dieser Diplomarbeit die Software *sintaX* (***S**ecure **Int**ernet **A**uction with **X**ML*), ein Prototyp einer sicheren Online - Auktion, entwickelt. Es handelt sich bei *sintaX* um eine Englische Auktion, das heißt, die Teilnehmer der Auktion überbieten sich gegenseitig und steigern dadurch den Verkaufspreis. Die Auktionen enden zu einem vorher festgesetzten Zeitpunkt, es gewinnt das höchste Gebot. Die Online - Auktion macht extensiven Gebrauch von signierten XML - Dokumenten, wodurch auch die Vorzüge der Spezifikation für XML - Signaturen demonstriert werden.

Im folgenden wird zunächst ausführlicher auf die Sicherheitsanforderungen einer Online - Auktion eingegangen. Im Anschluss wird gezeigt, wie sich diese Anforderungen umsetzen lassen, und es wird auf die Architektur der entwickelten Online - Auktion eingegangen.

8.1 Sicherheitsanforderungen

Die Sicherheitskonzepte von bestehenden Online - Auktionen können nicht als ausreichend bezeichnet werden. Zwar bieten diese Auktionen oftmals eine einfache Authentifizierung mittels Benutzername und Passwort und eine verschlüsselte Übertragung mittels SSL. Wichtige Aspekte wie Nichtabstreitbarkeit werden dagegen nicht erfüllt. Stattdessen beruhen diese Auktionen auf Vertrauen gegenüber den beteiligten Parteien. Um dieses Vertrauen zu schaffen, müssen die Teilnehmer einer Auktion häufig gegenseitig private Informationen preisgeben. Bei Ebay soll das Vertrauen dadurch gesteigert werden, dass die Teilnehmer die Möglichkeit haben, sich gegenseitig zu bewerten. Diese Bewertungen sind für alle Teilnehmer von der Webseite abrufbar. Zudem behält sich das Auktionshaus das Recht vor, Teilnehmer mit schlechten Bewertungen von der Auktion auszuschließen. Auf der Webseite von Ebay ist bei den Regeln und Nutzungsbedingungen der Auktionen zum Thema Vertrauen folgendes lesen.

1. *Wir glauben, dass die Menschen gut sind.*
2. *Wir glauben, dass eine offene Gemeinschaft das Beste in den Menschen hervorbringt.*
3. *Wir schätzen jeden als einzigartige Person.*
4. *Wir fordern jeden dazu auf, sich anderen gegenüber so zu verhalten, wie er von ihnen behandelt werden möchte.*
5. *Wir sehen im Vertrauen der Mitglieder untereinander das wertvollste Gut von eBay. Jeder soll deshalb die Bewertung seiner Handelspartner sehr ernst nehmen.*

(...)

Gerade in einem offenen und anonymen Medium wie dem Internet kann Vertrauen allein keine ausreichende Sicherheit bieten. Die beteiligten Parteien ändern sich ständig, für die Anzahl der beteiligten Parteien gibt es nach oben keine Grenzen. Es gibt keinen plausiblen Grund, warum ein Kunde einem Verkäufer trauen soll, von dem er außer einem Pseudonym nichts weiß, und der vielleicht Tausende von Kilometern entfernt lebt. Natürlich sieht es umgekehrt für den Verkäufer kaum besser aus. Auch er geht ein erhebliches Risiko ein, wenn er seine Ware an den Käufer versendet, bevor er sein Geld bekommen hat. Gewinner dieser Lage sind Treuhand - Unternehmen, die als Vermittler zwischen Käufer und Verkäufer auftreten. Der Käufer kann bei diesen Unternehmen den Betrag, zu dem

er eine Ware ersteigert hat, solange hinterlegen, bis er die Ware erhalten hat.

Wie anfällig Online - Auktionen für Betrug und Missbrauch sind, zeigt ein Bericht[1] des *Internet Fraud Complaint Center* (IFCC), einer Meldestelle für Internet - Betrug. Diesem Bericht zufolge sind in den USA Online - Auktionen der mit Abstand häufigste Anlass für Betrugsmeldungen. Von den mehr als 20.000 Beschwerden, die das IFCC in der zweiten Hälfte 2000 erreichten, betrafen 64 Prozent Online - Auktionen.

Dass derzeit existierende Online - Auktionen keine ausreichende Rechtssicherheit bieten, zeigt auch ein Fall aus dem Jahr 1999 (vgl. *[BG00]*). Ein Kunde ersteigerte bei dem Online - Auktionshaus Ricardo einen Neuwagen zu einem Preis, der deutlich unter dem Wert des Wagens lag. Der Anbieter weigerte sich nach Beendigung der Auktion, den Wagen auszuliefern, weil ihm der Preis zu niedrig war. Der Fall landete vor Gericht. Der Kläger führte an, dass es sich bei der Versteigerung um einen rechtsgültigen Vertrag handelt, und der Anbieter den Wagen daher auszuliefern habe. Erstaunlicherweise entschied das Gericht jedoch zu Gunsten des Verkäufers. In der Urteilsbegründung hieß es, der Käufer hätte erkennen müssen, dass ein Angebot, in dem ein Wagen zu einem Mindestpreis von 10 DM angeboten wird, offensichtlich nicht ernst gemeint sein kann. Weiter hieß es, die gegenüber dem Auktionshaus abgegebene Erklärung zum Verkauf des Wagens sei zu abstrakt und pauschal, und kann somit nicht als verbindliche Verkaufsverpflichtung im rechtlichen Sinne gelten. Zudem verglich das Gericht das Zeitlimit bei Online - Auktionen mit Glücksspielen und stellte damit die Seriosität von Online - Auktionen in Frage.

Prinzipiell lassen sich daraus die folgenden Anforderungen an eine sichere Online - Auktion ableiten.

- *Vertraulichkeit*. Die Nachrichten, die zwischen den beteiligten Parteien Auktionator und Käufer ausgetauscht werden, sind vertraulich zu behandeln. Für unautorisierte Personen soll es nicht möglich sein, diese Nachrichten zu lesen.

- *Integrität der Nachrichten*. Es darf nicht möglich sein, Nachrichten unerlaubt zu manipulieren oder zu verändern. Solche Manipulationen müssen sich aufdecken lassen.

[1] Quelle: Heise News - Ticker (http://www.heise.de/newsticker/data/fro-08.03.01-000/)

- *Authentifizierung*. Käufer und Auktionator sollen sich beim Nachrichtenaustausch gegenseitig identifizieren, um zu vermeiden, dass sich unautorisierte Personen unter falschem Namen ausgeben.
- *Authentizität der Nachrichten*. Die Einzigartigkeit der Nachrichten soll sicher gestellt sein.
- *Nichtabstreitbarkeit*. Alle ausgetauschten Nachrichten sind verbindlich. Der Sender einer Nachricht kann später nicht abstreiten, die Nachricht gesendet zu haben.
- *Anonymität*. Die Bieter einer Auktion bleiben gegenüber ihren Konkurrenten und gegenüber dem Anbieter einer Auktion anonym. Bieter und Anbieter lernen sich nur dann kennen, wenn es zum Zuschlag kommt.
- *Fairness*. Der Auktionator soll sich den Bietern gegenüber fair verhalten. Die Regeln einer Auktion sollen verbindlich und frei zugänglich sein. Die Bieter sollen den korrekten Verlauf der Auktion verifizieren können. Der Auktionator sollte nicht in der Lage sein, einzelne Bieter zu diskriminieren und andere zu bevorzugen. *[TSF01][KF98][FR96]*

8.2 Umsetzung

In diesem Abschnitt wird gezeigt, wie die zuvor genannten Sicherheitsanforderungen bei der Implementation von *sintaX* umgesetzt wurden. Dabei liegt der Schwerpunkt auf der Realisierung der Gebotsphase und der Ermittlung des Gewinners einer Auktion.

8.2.1 Allgemeines

Um Vertraulichkeit zu erreichen, bietet es sich an, die gesendeten Nachrichten zu verschlüsseln. Alle vertraulichen Nachrichten werden auf Transportebene via SSL (*Secure Socket Layer*) verschlüsselt[2]. Um die Integrität der Nachrichten sicher zu stellen, werden grundsätzlich alle Nachrichten digital signiert. Dadurch ist es nicht möglich, Nachrichten nachträglich zu manipulieren, da solche Manipulationen beim Verifizieren der Signatur entdeckt werden. Durch die Verwendung asymmetrischer kryptographischer Verfahren im Zusammenspiel mit Zertifikaten können sich das Auktionshaus und die Kunden gegenseitig identifizieren. Zudem können nur registrierte Personen an den Auktionen teilnehmen. Die Authentifizierung beim Betreten des Online - Auktionshauses erfolgt gegenwärtig

[2] z.Z. noch nicht implementiert

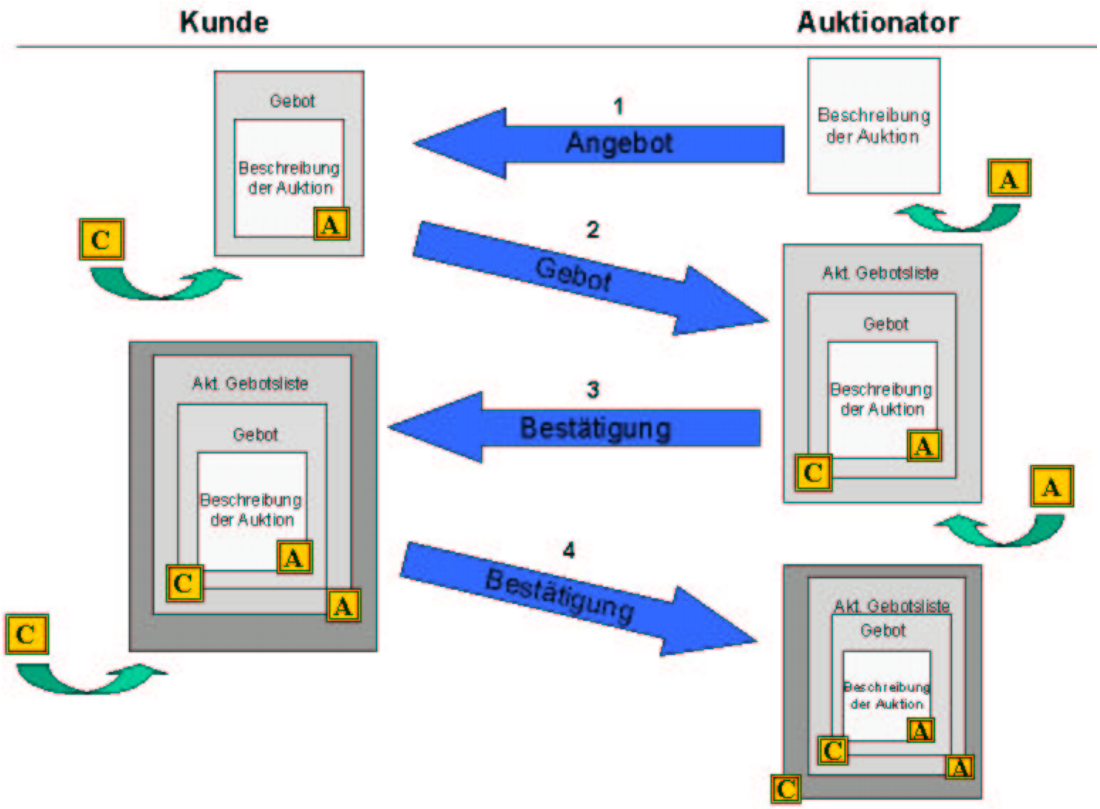

Abbildung 8.1: Protokoll der Gebotsabgabe

mittels schwacher Authentifizierung durch die Eingabe eines Benutzernamens und eines Passwortes. Zukünftige Versionen von *sintaX* werden starke Zwei - Wege Authentifizierung auf Basis von *Public Key* Kryptographie unterstützen. Um die Einzigartigkeit der Nachrichten zu gewährleisten, werden alle signierten Nachrichten zusätzlich mit einem Zeitstempel versehen, der den Zeitpunkt der Signierung angibt. Zudem enthalten alle signierten Angebote des Auktionshauses eine eindeutige ID. Auf diese Weise lässt sich das Gebot eines Kunden auf eindeutige Weise dem dazugehörigen Angebot zuordnen.

8.2.2 Protokoll der Gebotsabgabe

Die gegenseitige Nichtabstreitbarkeit spielt während der Gebotsphase eine wichtige Rolle. Durch die Abgabe eines Gebots entsteht ein verbindlicher Vertrag zwischen dem Kunden und dem Auktionshaus. Der Kunde verpflichtet sich, den von ihm gebotenen Preis zu zahlen, falls er die Auktion gewinnen sollte. Das Auktionshaus verpflichtet sich, das Gebot des Kun-

den in die Auktion aufzunehmen, das heißt, sein Gebot bei der Ermittlung des Gewinners zu berücksichtigen und, sofern er gewinnt, die ersteigerte Ware an ihn auszuliefern. Um diese gegenseitige Nichtabstreitbarkeit zu erzielen, beruht die Abgabe eines Gebots auf dem bereits in Kapitel 5 vorgestellten BAKO - Protokoll. Abbildung 8.1 zeigt schematisch das Protokoll für die Gebotsabgabe. Im Gegensatz zu dem in Kapitel 5 beschriebenen Protokoll sind zur Abgabe eines Gebots nicht drei sondern vier Schritte notwendig. Wie später noch gezeigt wird, handelt es sich bei diesem Protokoll jedoch nicht um eine Erweiterung des BAKO - Protokolls, sondern vielmehr um zwei sich überlagernde BAKO - Sitzungen.

Im ersten Schritt des Protokolls macht der Auktionator A einem Bieter B_i ein verbindliches Angebot (Abbildung 8.2). Mit diesem Angebot bietet der Auktionator an, ein bestimmtes Produkt zu den genannten Bedingungen und zu einem bestimmten Mindestpreis zu ersteigern. Das Angebot des Auktionators hat den folgenden Aufbau.

$$offer_i = product_descr, rules, history, offer_id_i, timestamp, cert_A, Sig_{A_i}(product_descr, rules, history, offer_id_i)$$

Dabei bezeichnet *product_descr* die Beschreibung des angebotenen Produkts. *rules* beschreibt alle für die Auktion relevanten Bedingungen. Dazu gehören die Zeitpunkte, zu denen die Versteigerung startet und beendet wird und die Regeln, nach denen nach Ablauf der Auktion der Gewinner ermittelt wird. *history* enthält die aktuelle Gebotsliste, anhand derer der Bieter den bisherigen Verlauf der Auktion nachvollziehen kann. *offer_id* bezeichnet eine einmalige, nur für dieses spezielle Angebot vergebene *ID*, die eine eindeutige Zuordnung von Angebot und Gebot ermöglicht. $Sig_{A_i}(data)$ kennzeichnet die digitale Signatur über den angegebenen Daten (*data*). Es werden die Produktbeschreibung, die Regeln, die Gebotsliste, die *ID* und der Zeitpunkt der Signierung (*timestamp*) signiert. Zum Signieren wird der RSA - Algorithmus verwendet. $cert_A$ bezeichnet das *Public Key* Zertifikat des Auktionators. Dieses Dokument erfüllt alle genannten Anforderungen. Anhand des Zertifikats kann der Bieter die Identität des Auktionators überprüfen. Durch die digitale Signatur ist die Integrität der Daten gewährleistet und das Angebot ist verbindlich. Durch die Angabe eines Zeitstempels und einer *ID* ist die Einmaligkeit des Dokuments sicher gestellt.

Im zweiten Schritt gibt der Bieter B_i seinerseits ein Gebot ab. Durch die Abgabe eines Gebots akzeptiert der Bieter das Angebot des Auktionators aus Schritt eins. Er bietet dem Auktionator an, das beschriebene Produkt zu einem von ihm festgelegten Preis zu kaufen, und stimmt den in dem Angebot enthaltenen Regeln zu. Das Gebot sieht wie folgt aus.

Abbildung 8.2: Angebot des Auktionators (Screenshot)

$$bid_i = bid_info_i, offer_i, offer_id_i, timestamp, cert_{B_i}, Sig_{B_i}(bid_info_i, offer_i, offer_id_i)$$

Hierbei steht bid_info_i für das eigentliche Gebot. Es enthält alle für das Gebot relevanten Daten, wie etwa den gebotenen Betrag, den Benutzernamen des Bieters und die Angabe des Produkts (*ID*). $offer_i$ enthält das komplet-

te Angebot, das der Auktionator im ersten Schritt an den Bieter gesendet hat. Wichtig ist, dass in diesem Schritt nicht nur die Informationen über das abgegebene Gebot signiert werden, sondern auch das erhaltene Angebot. Durch die Signatur des Gebots entsteht Verbindlichkeit, der Bieter kann nicht mehr bestreiten, das Gebot abgegeben zu haben. Durch die Signatur des Angebots signalisiert der Bieter, dass er das Angebot erhalten hat, und dass er die darin enthaltenen Bedingungen zur Ersteigerung des Produktes zur Kenntnis genommen hat und akzeptiert. Daneben enthält auch dieses Dokument einen Zeitstempel (*timestamp*) und das *Public Key* Zertifikat, in diesem Fall natürlich das des Bieters. Durch die $offer_id_i$ entsteht nun eine eindeutige Beziehung zwischen Angebot ($offer_i$) und Gebot (bid_i), wodurch sogenannte *replay attacks* (wiederholtes Senden ein und desselben Dokuments) unmöglich gemacht werden.

Im dritten Schritt bestätigt der Auktionator den Erhalt des Gebotes. Diese Bestätigung hat den folgenden Aufbau.

$$confirmation_{A_i} = bid_i,\ history_upd_i,\ timestamp,\ cert_A,\ Sig_{A_i}(bid_i,\ history_upd_i)$$

Die Bestätigung des Auktionators enthält zum einen das komplette Gebot bid_i, das wiederum das Angebot aus dem ersten Schritt beinhaltet. Dadurch, dass der Auktionator das Gebot signiert, kann er nicht abstreiten, das Gebot erhalten zu haben. Interessant an dieser Bestätigung ist, dass sie daneben auch die um dieses Gebot aktualisierte Gebotsliste enthält ($history_upd_i$), und diese Gebotsliste auch signiert wurde. Dadurch bestätigt der Auktionator dem Bieter explizit, dass er das Gebot nicht nur erhalten hat, sondern dass er das Gebot auch tatsächlich in die Versteigerung aufgenommen hat. Dadurch kann sich der Bieter vom ordnungsgemäßen Verlauf der Auktion überzeugen. Diese Bestätigung dient alles in allem der Sicherheit des Bieters, da der Auktionator nun nicht mehr bestreiten kann, das Gebot in die Auktion aufgenommen zu haben.

Im vierten und letzten Schritt bestätigt der Bieter seinerseits den Erhalt der aktualisierten Gebotsliste.

$$confirmation_{B_i} = confirmation_{A_i},\ timestamp,\ cert_{B_i},\ Sig_{B_i}(confirmation_{A_i})$$

Dieses Dokument ist recht einfach aufgebaut und enthält im wesentlichen nur die Bestätigung des Auktionators aus Schritt drei ($confirmation_{A_i}$) und die Signatur des Bieters. Diese Bestätigung des Bieters dient dem Schutz des Auktionator. Ohne diesen Schritt könnte der Bieter sich, falls er die Auktion gewinnt, weigern, das ersteigerte Produkt zu kaufen, indem er

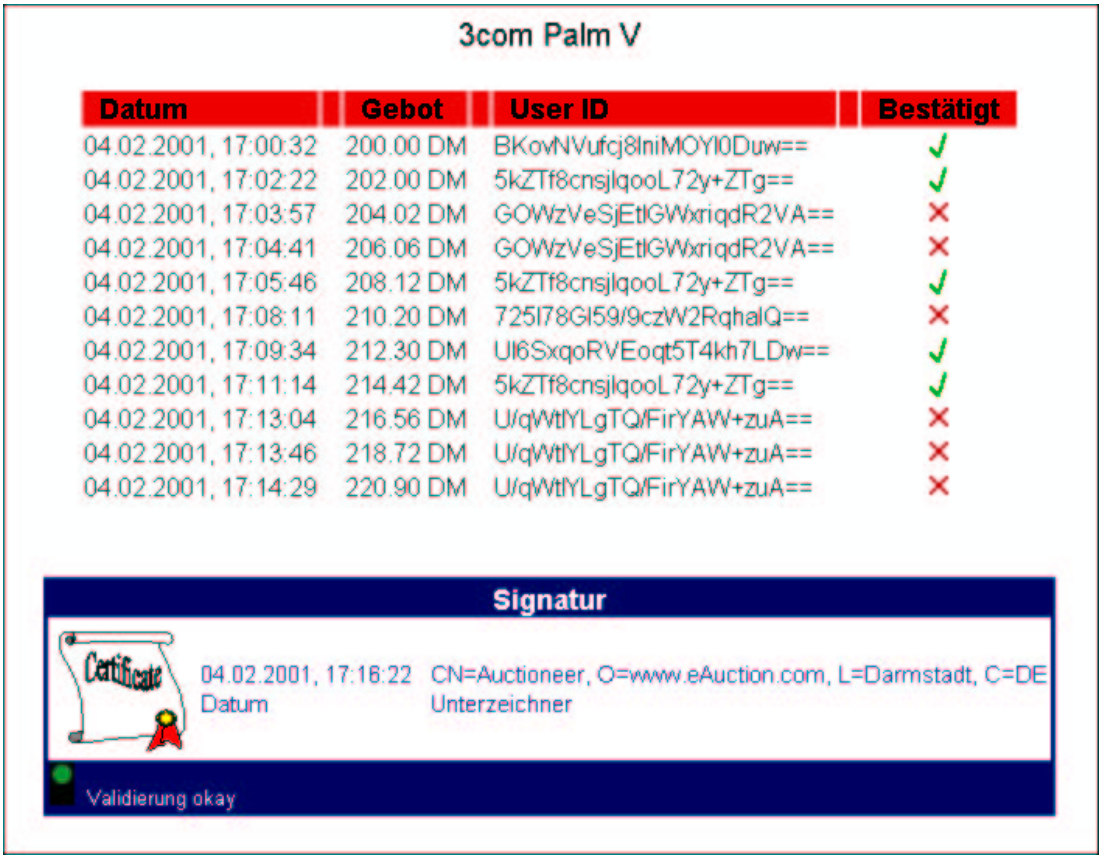

Abbildung 8.3: Gebotsliste (Screenshot)

behauptet, nie eine Bestätigung darüber erhalten zu haben, dass sein Gebot auch tatsächlich angenommen wurde. Er macht somit also in diesem Schritt deutlich, dass er sich davon überzeugt hat, dass sein Gebot in die Auktion aufgenommen wurde. Dies bedeutet aber gleichzeitig auch, dass das Gebot im dritten Schritt von dem Auktionator nur unter Vorbehalt in die Auktion aufgenommen wurde. Endgültig aufgenommen wird das Gebot erst, wenn der Auktionator die Bestätigung des Bieters erhalten hat. Gebote, die nach Ablauf der Auktion unbestätigt sind, werden, wie später noch gezeigt wird, nicht bei der Gewinnerermittlung berücksichtigt.

Nach Ablauf dieser vier Schritte ist die Abgabe eines Gebotes beendet. Auf diese Weise abgegebene Gebote sind endgültig, es besteht die aus dem BAKO - Protokoll bekannte gegenseitige Nichtabstreitbarkeit. Einmal abgegebene Gebote können dadurch auch nicht mehr zurückgezogen werden. Bei der Beschreibung der vier Schritte wurde stets vorausgesetzt, dass eine ausreichende Validierung der erhaltenen Dokumente erfolgt. Das bedeu-

tet, es werden alle in einem Dokument enthaltenen Signaturen geprüft und es wird in den Schritten zwei bis vier sicher gestellt, dass es sich bei den in den Dokumenten enthaltenen signierten Dokumenten um diejenigen aus den jeweiligen Schritten davor handelt.

Wie schon angedeutet, handelt es sich bei diesen vier Schritten nicht um eine Erweiterung des BAKO - Protokolls, sondern es handelt sich um zwei sich überlagernde BAKO - Sitzungen. Die erste BAKO - Sitzung beginnt im ersten Schritt und endet im dritten Schritt. Im ersten Schritt gibt der Auktionator ein verbindliches Angebot zur Ersteigerung eines Produkts ab. Der Bieter akzeptiert das Angebot im zweiten Schritt, indem er seine Unterschrift unter das Angebot setzt. Im dritten Schritt bestätigt der Auktionator den Erhalt dieses Dokuments. Indem der Bieter im zweiten Schritt das Angebot des Auktionators akzeptiert, unterbreitet er gleichzeitig aber auch dem Auktionator ein Angebot. Er bietet ihm nämlich an, das Produkt zu einem bestimmten Betrag zu kaufen. Damit beginnt also eine zweite BAKO - Sitzung. Wenn der Auktionator im dritten Schritt das Dokument des Bieters bestätigt, akzeptiert er damit auch das Angebot des Bieters. Dies macht er deutlich, indem er das Gebot in die Gebotsliste aufnimmt. Der vierte Schritt stellt dann den letzten Schritt der zweiten BAKO - Sitzung dar. Der Bieter bestätigt den Erhalt der aktualisierten Gebotsliste, wodurch der Vertrag (die Abgabe eines Gebots) endgültig abgeschlossen wird.

8.2.3 Anonymität

Im folgenden wird gezeigt, wie die Anonymität gegenüber den konkurrierenden Bietern und dem Anbieter erreicht wird. Alle abgegebenen Gebote enthalten neben dem Betrag und dem Benutzername einen Hashwert über einem von dem Bieter gewählten Geheimnis. Das Element bid_info_i aus dem zweiten Schritt hat also den folgenden Aufbau.

$$bid_info_i \quad = \quad amount_i, user_name_i, h(secret_i), product_id$$

In der Gebotsliste erscheint, wie in Abbildung 8.3 zu sehen ist, statt des Benutzernamens nur der Hashwert über dem Geheimnis ($h(secret_i)$). Dies hat zwei Dinge zur Folge. Zum einen kann der Bieter seine eigenen Gebote eindeutig in der Liste identifizieren, da er den Hashwert seines Geheimnisses kennt. Gleichzeitig bleibt er jedoch gegenüber den anderen Bietern anonym, da sich auf Grund des Hashwertes keine Rückschlüsse auf die Identität des Bieters ziehen lassen. Da der Bieter sein Geheimnis beliebig oft ändern kann, ist es prinzipiell möglich, dass in jedem seiner Gebote ein anderer Hashwert erscheint. Dadurch lässt sich für einen Beobachter der

Auktion nicht einmal eindeutig feststellen, wieviele Personen sich an einer Auktion beteiligen. Wenn jedoch ein anderer Bieter B_j behauptet, das Gebot des Bieters B_i wäre seines, muss er beweisen, wie man den zu dem Gebot gehörenden Hashwert bildet. Dies kann jedoch nur der Bieter B_i, da nur er das Geheimnis kennt.

Das Auktionshaus dagegen kennt die Bieter, schon allein deswegen, da nur registrierte Personen zugelassen sind. Zudem enthalten die Gebote neben dem Hashwert auch den Benutzername, und sie wurden von dem Bieter digital signiert, so dass der Auktionator die abgegebenen Gebote seinen Kunden zuordnen kann. Es stellt sich hierbei die Frage, ob dies wirklich notwendig ist, oder ob sich nicht auch eine Anonymität der Kunden gegenüber dem Auktionshaus erreichen lässt. Dadurch, dass der Auktionator die Gebote seinen Kunden zuordnen kann, könnten sich Manipulationsmöglichkeiten durch den Auktionator ergeben. Sind die Kunden dagegen anonym, kann der Auktionator beispielsweise nicht den Gewinner benachrichtigen. Es zeigt sich, dass dies ein bislang ungelöstes Problem zumindest bei Englischen Auktionen ist. Anders sieht es bei versiegelten Auktionen aus. Bei diesen Auktionen öffnet der Auktionator die Gebote erst nach Ablauf der Auktion und kann sie erst dann den Kunden zuordnen.

8.2.4 Gewinnerermittlung

Abbildung 8.3 zeigt den beispielhaften Verlauf einer Auktion, bei der einige Gebote nicht von dem jeweiligen Bieter bestätigt wurden, erkennbar an dem roten Kreuz in der Spalte „Bestätigt“. Bei der Ermittlung des Gewinners der Auktion werden jedoch nur die bestätigten Gebote berücksichtigt. Der Gewinner ist demnach der Bieter, der das höchste und damit das letzte bestätigte Gebot abgegeben hat. Nun muss jedoch noch der Preis bestimmt werden, den der Gewinner zu zahlen hat. Würde man einfach den Preis ansetzen, den der Gewinner geboten hat, ergibt sich eine eklatante Sicherheitslücke. Ein Betrüger könnte ungefährdet die Auktion manipulieren und den Preis in die Höhe treiben, ohne befürchten zu müssen, die Auktion zu gewinnen. Er kann nämlich, so wie in dem Beispiel in Abbildung 8.3 geschehen, Gebote abgeben, ohne sie anschließend zu bestätigen. So könnte etwa der Verkäufer des Produktes vereinzelt unbestätigte Gebote einstreuen, um einen höheren Verkaufspreis zu erzielen.

Es ist daher notwendig, die nicht bestätigten Gebote aus dem Verkaufspreis herauszurechnen, indem man den Preis der bestätigten Gebote entsprechend zurücksetzt, wenn zuvor unbestätigte Gebote abgegeben wurden. Dies lässt sich mit einem *FIFO* - Speicher (*First in first out*) realisie-

ren. Im folgenden wird der verwendete Algorithmus kurz in einer java - ähnlichen Notation vorgestellt.

```
n = index_of_highest_confirmed_bid;
for(i = 0; i < n; i++) {
   put(bid[i]);
   if(bid[i] is confirmed) {
      bid = get();
      winner_price = price(bid);
   }
}
```

Hierbei bezeichnen *get()* und *put()* die beiden Operationen auf dem *FIFO* - Speicher, *put()* schreibt ein Element an das Ende des Speichers, *get()* liefert und entfernt das Element am Anfang des Speichers. Die Gebote werden sukzessive in den Speicher geschrieben. Wenn es sich um ein bestätigtes Gebot handelt, wird das Gebot, das am Anfang des *FIFO* - Speichers steht, weggenommen. Dessen Höhe stellt den vorläufigen Preis dar, den der Gewinner zu zahlen hat. Nach Beendigung des Algorithmus steht in der Variablen *winner_price* der endgültig ermittelte Preis, den der Gewinner zahlen muss. Da die Schleife nur bis zu dem höchsten bestätigten Gebot, also dem Gebot des Gewinners, läuft, kann der ermittelte Verkaufspreis nie höher sein, als der von dem Gewinner gebotene Preis. Nach diesem Algorithmus müsste der Gewinner aus dem Beispiel in Abbildung 8.3 208,12 DM zahlen.

Mit dieser Form der Ermittlung des Gewinners bietet die Auktion jedoch noch immer einen Angriffspunkt. Zur Verdeutlichlung dient die Gebotsliste in Tabelle 8.1. In diesem Beispiel wurden vier bestätigte Gebote abge-

Gebot	Betrag	Bestätigung
1.	1,00 DM	bestätigt
2.	2,00 DM	bestätigt
3.	3,00 DM	bestätigt
4.	4,00 DM	bestätigt
5.	1.000,00 DM	nicht bestätigt

Tabelle 8.1: Gebotsliste (Beispiel)

geben, der fünfte Bieter hat jedoch ein dem Anschein nach unverhältnismäßig hohes Gebot abgegeben und dies nicht bestätigt. Nach diesem Gebot war kein weiterer Bieter mehr bereit, ein weiteres Gebot abzugeben, denn ein weiteres Gebot müsste höher sein, als das fünfte Gebot, und würde bei

der Gewinnermittlung bestenfalls auf 1.000,00 DM zurückfallen. Der Bieter des fünften Gebots hat somit dafür gesorgt, dass die Versteigerung mit seinem Gebot endet, da sich kein weiterer Bieter findet, ohne aber jemals Gefahr zu laufen, seinen hohen Betrag auch tatsächlich zahlen zu müssen. Denkbar wäre beispielsweise, dass der Bieter des vierten Gebots im Anschluss ein extrem hohes, unbestätigtes Gebot abgibt, und sich auf diese Weise den Gewinn sichert.

Diese Art der Manipulation wird verhindert, indem die Höhe eines Gebots von dem Auktionshaus vorgegeben wird. Die einzelnen Bieter können also den Betrag ihres Gebots nicht frei wählen. Sie müssen den vorgegebenen Betrag akzeptieren oder auf das Gebot verzichten. Die Gebote werden in Ein - Prozent - Schritten erhöht, die Mindestschrittweite beträgt jedoch 1,00 DM.

8.2.5 Fairness

Der Aspekt der Fairness bezieht sich auf das Verhalten des Auktionators. Er sollte sich seinen Kunden gegenüber fair verhalten. Bei der Umsetzung der Online - Auktion wurde versucht, einen möglichst hohen Grad an Fairness zu gewährleisten. Dazu gehört zum einen, dass die Regeln der Auktion für jedermann zugänglich sind. Dies wird dadurch erreicht, dass sie von der Webseite des Auktionshauses abrufbar sind. Sie wurden von dem Auktionator digital signiert und sind damit verbindlich. Außerdem sind die Regeln fest an die Angebote gebunden, so dass sicher gestellt ist, dass ein Bieter die Regeln erhält, bevor er ein Gebot abgibt. Entsprechend verhält es sich mit den Gebotslisten zu den Auktionen. Auch sie werden auf der Webseite veröffentlicht, damit jedermann den Verlauf der Auktionen verfolgen kann. Zusätzlich sind sie aber auch Teil der Angebote.

Der Auktionator hat nicht die Möglichkeit, einzelne Bieter zu diskriminieren. Der Auktionator macht den ersten Schritt während der Gebotsabgabe. Er macht im ersten Schritt dem Kunden ein verbindliches Angebot zur Ersteigerung eines Produkts, und damit zur Abgabe eines Gebots. Er macht dieses Angebot also bevor der Kunde signalisiert, dass er ein Gebot abgeben möchte. Da dieses Angebot verbindlich ist und als Beweismittel dienen kann, kann der Auktionator später ein abgegebenes Gebot nicht diskriminieren. Zudem kann der Auktionator eine Auktion nicht manipulieren, indem er bereits abgegebene Gebote verändert. Es würde nicht genügen, die Höhe des Gebots in der Gebotsliste zu verändern, er müsste diese Änderung auch in seiner Bestätigung, die er dem Bieter sendet, vornehmen. Dies würde jedoch dazu führen, dass die Signatur in

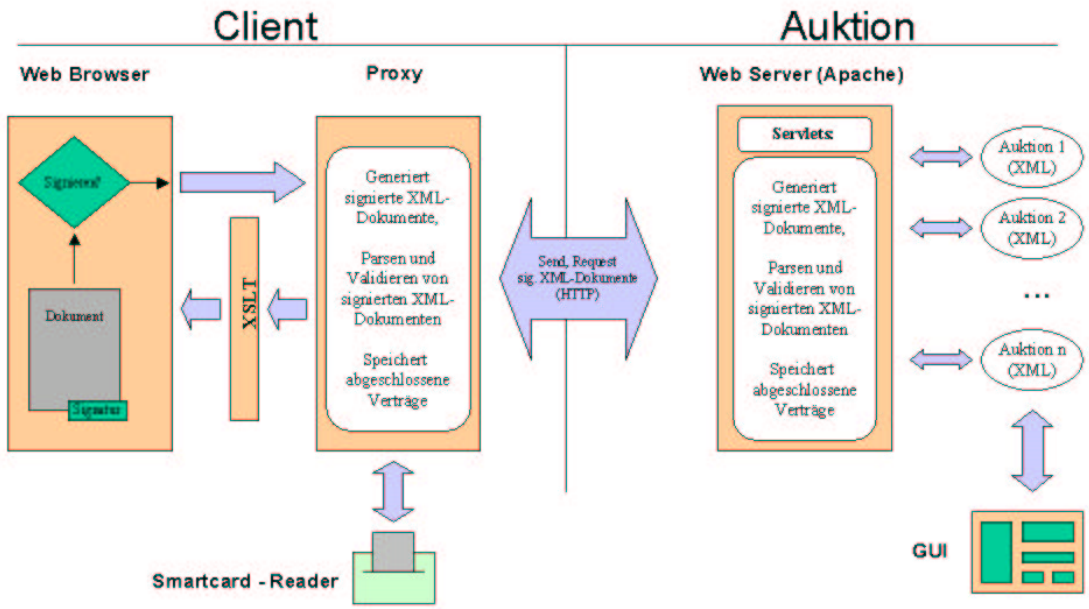

Abbildung 8.4: Architektur von *sintaX*

dem Gebot, das in der Bestätigung enthalten ist, bricht. Das Dokument lässt sich nicht validieren, die Manipulation würde sofort entdeckt werden. *[KF98][TSF01][SS99][Cac99][NPS99]*

8.3 Architektur

In diesem Abschnitt werden einige Details der Architektur von *sintaX* beschrieben. Ein Ziel bei der Entwicklung von *sintaX* war es, die Bedienung für den Benutzer möglichst einfach und intuitiv zu gestalten. Gerade der Umgang mit digitalen Signaturen und den dazu nötigen Schlüsseln sollte den Benutzer nicht überfordern. Die Bedienung sollte zudem in erster Linie über einen Web - Browser erfolgen. Die komplette Anwendung, das heißt, der Client als auch der Server, sollten sowohl auf einem Einzelplatzrechner lauffähig sein, als auch auf mehrere über das Internet verbundene Rechner verteilt werden können. Zudem sollten die entwickelten Softwarekomponenten nach Möglichkeit plattformunabhängig sein, das bedeutet, sie sollten zumindest unter *Microsoft Windows* und unter *Unix / Linux* laufen. Um eine einfache Portierung auf verschiedene Betriebssysteme zu gewährleisten, bietet sich *Java* als verwendete Programmiersprache an. Die Plattformunabhängigkeit von Java wird dadurch erreicht, dass Java - Programme in einer plattformabhängigen virtuellen Maschine (*Java Virtual Machine*) laufen. Der Nachteil dieser Lösung ist die vergleichsweise gerin-

ge Geschwindigkeit. Diese *Performance* - Probleme machen sich besonders bei der Verwendung mit *Java Swing* erstellter graphischer Benutzeroberflächen und der Verwendung in Java implementierter kryptographischer Algorithmen bemerkbar. Daher wird für den optimalen Einsatz von *sintaX* ein moderner, leistungsfähiger Rechner vorausgesetzt.

Bei der Umsetzung von *sintaX* werden verschiedene kryptographische Algorithmen verwendet. Um diese Algorithmen nicht neu implementieren zu müssen, bietet es sich an, auf bereits implementierte Klassenbibliotheken zurückzugreifen. Auf Grund der Exportrestriktionen der Vereinigten Staaten sollten keine amerikanischen Implementierungen verwendet werden, da sie nur eine schwache Verschlüsselung bieten und einige Algorithmen wie RSA gänzlich fehlen. Für die Implementierung von *sintaX* standen im wesentlichen zwei Produkte zur Auswahl, die JCE - Implementierung (*Java Cryptography Extension*) vom IAIK[3] und die JCE - Implementierung von *SECUDE*. Der Vorteil des IAIK - Produkts ist, dass es zumindest für wissenschaftliche Zwecke frei verfügbar ist. Es wurde jedoch die *SECUDE* - Implementierung gewählt, da sie im Gegensatz zum IAIK - JCE Chipkarten (Smartcards) unterstützt. Die vom Rechner getrennte Aufbewahrung des zum Signieren benötigten Schlüsselpaares auf einer *Smartcard* bietet, einmal abgesehen von biometrischen Verfahren, das höchste Maß an Sicherheit. Alternativen dazu sind die Verwendung von Disketten als Schlüsselspeicher oder die direkte Ablegung des Schlüssels auf dem Dateisystem des Rechners. Beide Varianten bieten jedoch keinen ausreichenden Schutz vor unbefugtem Zugriff auf den privaten Schlüssel. Entsprechende Chipkartenlesegeräte sind mittlerweile recht günstig zu haben, sind für verschiedene Anschlüsse - serielle Schnittstelle, USB und PCMCIA - erhältlich und werden von *SECUDE* unterstützt. Probleme kann es jedoch bei der Verfügbarkeit der benötigten Treiber geben, wenn andere Betriebssysteme außer Microsoft Windows zum Einsatz kommen sollen.

Da *sintaX* starken Gebrauch von XML als Datenformat und XSLT-Transformationen macht, werden auch entsprechende in Java implementierte XML - Parser und XSLT - Prozessoren benötigt. Auch hier bietet es sich an, fertige Produkte zu verwenden, statt einen eigenen Parser zu entwickeln. Die Wahl fiel hierbei auf die Produkte von Apache, *Xerces* und *Xalan*, da sie sehr stark verbreitet sind und ständig weiterentwickelt werden.

[3]Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie an der TU Graz

Zur Darstellung und Bedienung kann prinzipiell sowohl der *Microsoft Internet Explorer* ab Version 4 als auch der *Netscape Communicator* verwendet werden. Zur optimalen Darstellung wird jedoch der Internet Explorer empfohlen. Da dieser nicht für Unix / Linux erhältlich ist, sollte somit Windows als zugrunde liegendes Betriebssystem vorgezogen werden. Es gibt im wesentlichen zwei Gründe für die Bevorzugung des Internet Explorers. Zur Darstellung wurden Cascading Stylesheets (CSS) verwendet, die allerdings in der aktuellen Netscape Version noch fehlerhaft implementiert sind, sodass es zu fehlerhaften Darstellungen kommt. Zum anderen kommen HTML - Seiten mit stark ineinander verschachtelten HTML - Tabellen zum Einsatz. Aufgrund eines *Bugs* im Netscape Communicator braucht dieser Browser extrem lange, um solche HTML - Seiten zu rendern und letztendlich darzustellen. Auf der Server - Seite wird ein Apache - Webserver mit *Servlet* - Unterstützung benötigt. Theoretisch könnte wohl auch ein anderer Webserver verwendet werden, sofern er *Servlets* unterstützt, der Apache Server wurde jedoch wegen seiner freien Verfügbarkeit verwendet.

Zusammenfassend stellt *sintaX* somit folgende Anforderungen.

- Microsoft Windows NT oder Windows 2000 (getestet), prinzipiell auch Unix / Linux (jedoch nicht getestet)
- Microsoft Internet Explorer ab Version 4 (empfohlen) oder Netscape Communicator
- Apache Web - Server mit Servlet - Unterstützung
- Java Development Kit (JDK) 1.2 von Sun
- *SECUDE - JCE*
- Apache *Xerces 1.0.3* und *Xalan 1.0.1*
- Leistungsfähiger Rechner und Chipkartenlesegerät

Die Architektur von *sintaX* ist in Abbildung 8.4 dargestellt. Die wesentlichen Komponenten auf der *Server* - Seite sind ein Webserver mit *Servlet* - Unterstützung und eine graphische Benutzerschnittstelle zum Starten und Verwalten der Auktionen. Auf der *Client* - Seite dient ein Browser zur Darstellung der Auktionsseiten und somit zur Bedienung der Auktion. Zusätzlich läuft auf der *Client* - Seite ein Proxy, der als Signierwerkzeug des Kunden dient, und für die Transformation von XML - Dokumenten in HTML - Dokumente zuständig ist. Die Kommunikation zwischen Client und Server erfolgt durch *HTTP - Requests*. *sintaX* macht starken Gebrauch

von XML als Datenformat. Ziel war es, möglichst viel mit XML zu beschreiben, um das Potential von XML zu demonstrieren. So werden etwa die Auktion, Gebote und Angebote, aber auch Konfigurationsdokumente mit Hilfe von XML dargestellt. Dabei finden insbesondere in großem Umfang XML - Signaturen gemäß der W3C - Spezifikation Anwendung. Damit dürfte es sich bei *sintaX* um eine der ersten praktischen Anwendungen dieser W3C - Spezifikation handeln. Zur Generierung der XML - Signaturen wird die in Abschnitt 4.4 vorgestellte API verwendet. Im folgenden werden die einzelnen Komponenten näher vorgestellt.

8.3.1 Server

Auf der *Server* - Seite läuft ein *Apache* Webserver mit *Servlet* - Unterstützung. Die Servlets[4] übernehmen im wesentlichen die Steuerung der einzelnen Aktionen der Online - Auktion. Das bedeutet, es gibt Servlets für die Registrierung und Anmeldung der Benutzer, für das dynamische Generieren der signierten Dokumente, wie Angebote, Auktionsregeln und Gebotslisten, und für die Abwicklung der Gebotsphase. Die Durchführung der Versteigerungen und das Signieren der Dokumente erfolgen somit automatisiert im Hintergrund.

Jede Auktion wird durch ein XML - Dokument beschrieben. Das Dokument enthält alle nötigen Informationen, wie Produktbeschreibung, Bedingungen und Regeln sowie die Liste der eingegangenen Gebote. Ein Beispiel für solch ein Dokument ist in Anhang B zu sehen. Eine Auktion wird eindeutig durch eine ID (Element *ProductID*) definiert. Zur Produktbeschreibung zählt darüber hinaus ein Name und eine Beschreibung. Die Bedingungen einer Auktion setzen sich aus einem Startgebot (Element *StartBid*), dem Betrag des zuletzt abgegebenen Gebots (Element *LastBid*), sowie dem Start und dem Ende der Auktion (*AuctionStart*, *AuctionEnd*) und dem Status zusammen. Start und Ende einer Auktion werden durch ein Datum und eine Uhrzeit beschrieben. Die Angaben werden in dem Dokument sowohl in einem Textformat zur Darstellung, als auch durch die Angabe der Millisekunden abgelegt. Letzteres dient der einfacheren Verarbeitung der Daten. Der Status gibt an, ob die Auktion erst in Zukunft gestartet wird (Status 0), ob sie gerade läuft (Status 1) oder ob sie bereits beendet wurde (Status 2). Der Status findet in erster Linie bei der XSLT - Transformation in ein HTML - Dokument zur Darstellung eines Angebots Anwendung. Der Status entscheidet dabei beispielsweise, ob ein Button zur Abgabe eines Gebots eingeblendet wird. Das Element *Rules* enthält beliebigen Text, der die Regeln der Auktion beschreibt. Die Gebotsliste (Element *BidList*)

[4]Eine gute Einführung in Servlets bietet z.B. [HC98]

enthält beliebig viele *Bid* - Elemente zur Beschreibung der eingegangenen Gebote. Ein Gebot wird durch eine eindeutige ID (*ContractID*), dem Benutzername und dem Hashwert des Benutzers, dem Betrag und der Angabe, ob das Gebot bestätigt wurde, beschrieben.

Diese Dokumente stellen in dieser Form noch kein Angebot des Auktionators dar. Allerdings werden aus diesen Dokumenten die Angebote generiert. Um ein Angebot zu erzeugen, werden ein Zeitstempel und eine eindeutige ID gebildet. Es wird eine XML - Signatur generiert, in die die Beschreibung der Auktion, der Zeitstempel und die ID integriert werden. Diese XML - Signatur stellt dann ein verbindliches Angebot dar, das an einen Kunden gesendet werden kann.

Für jede Auktion wird ein *Thread* zur Überwachung des Auktionsendes gestartet. Der *Thread* berechnet den Zeitraum, bis die Auktion beendet wird, und legt sich entsprechend lang schlafen. Nach Beendigung der Auktion ermittelt der Thread mit Hilfe des in Abschnitt 8.2.4 vorgestellten Algorithmus den Gewinner der Auktion und aktualisiert die Beschreibung der Auktion. Für den Gewinner der Auktion wird eine Nachricht erzeugt, die der Gewinner über die Webseite des Auktionshauses abrufen kann. Diese Nachricht liegt in Form eines signierten XML - Dokuments vor und enthält neben einem erklärenden Text die ID und die Bezeichnung des ersteigerten Produkts und den ermittelten Betrag, zu dem das Produkt ersteigert wurde.

Nachrichten dieser Art werden nicht nur nach Beendigung einer Auktion generiert, sondern auch während der Gebotsphase. Nach Abgabe eines Gebots, das heißt, nachdem der Bieter sein Gebot bestätigt hat, wird für den Bieter eine entsprechende Nachricht erzeugt. Diese Nachricht informiert ihn darüber, dass sein Gebot führt. Gleichzeitig wird auch für denjenigen Bieter, dessen Gebot nun nicht mehr führt, eine entsprechende Nachricht generiert.

Zur Steuerung der Auktionen steht dem Auktionator eine graphische Benutzerschnittstelle (*Graphical User Interface - GUI*, Abbildung 8.5) zur Verfügung. Sie zeigt dem Auktionator alle bestehenden Auktionen an, und hat im wesentlichen zwei Aufgaben. Zum einen werden über diese Benutzerschnittstelle die Auktionen gestartet und können auch jederzeit abgebrochen werden. Zum anderen können mit Hilfe der Benutzerschnittstelle Auktionen editiert werden und es können neue Auktionen angelegt werden.

Abbildung 8.5: Benutzerschnittstelle für den Server (Screenshot)

8.3.2 Client

Bei der Entwicklung der *Client* - Seite stand im Vordergrund, dass der Kunde seine Gebote über seinen Browser abgeben kann. Das heißt, der Kunde besucht die Webseite des Auktionshauses, kann dort die Auktionen beobachten und gegebenenfalls durch Klicken auf einen entsprechenden Link ein Gebot abgeben. Der Nachrichtenaustausch zwischen Client und Server erfolgt bei *sintaX* auf Basis signierter XML - Dokumente. Browser stellen jedoch weder Funktionen zum Prüfen von Signaturen, insbesondere XML - Signaturen, bereit, noch können sie XML - Dokumente darstellen oder gar generieren. Es musste also ein Signierwerkzeug für den Kunden entwickelt werden, das mit XML - Signaturen umgehen kann und möglichst im Hintergrund arbeitet. Dabei sollte trotz allem nicht auf den Browser als Ausgabemedium verzichtet werden.

Das Problem wurde mit Hilfe eines *Proxys* gelöst, der lokal auf dem Rechner des Kunden installiert ist, und der im wesentlichen dazu dient, XML - Signaturen zu validieren und zu erzeugen, und XSLT - Transformationen durchzuführen. Als Alternativen hätten sich beispielsweise ein Browser - *Plug-In* oder ein *Java - Applet* angeboten. Diese Alternativen haben jedoch entscheidende Nachteile gegenüber einem Proxy. Ein Browser - *Plug-In* erfordert einen wesentlich höheren Entwicklungsaufwand, da es nicht plattformunabhängig ist. Ein solches *Plug - In* müsste nicht nur für verschiedene Betriebssysteme, sondern auch für verschiedene Browsertypen entwickelt werden. Der Proxy dagegen wurde in Java entwickelt, und ist somit plattformunabhängig. Ein *Java - Applet*, also ein Java - Programm, das in einen Browser geladen wird und dort läuft, hat den Nachteil, dass es von einer Web - Seite geladen werden muss. Da es in diesem Zusammenhang benötigt wird, wenn sich der Kunde auf den Webseiten des Auktionshauses befindet, würde das Applet von dem Auktionshaus geliefert werden. Dies würde jedoch bedeuten, dass das Auktionshaus nicht nur signierte Dokumente sendet, sondern gleichzeitig auch ein Werkzeug, mit denen sich die von ihm signierten Dokumente validieren lassen. Dies stellt natürlich eine gravierende Sicherheitslücke dar, da der Kunde keine Möglichkeit hat, sich von der korrekten Funktionsweise des Applets zu überzeugen. Es könnte etwa grundsätzlich alle Dokumente, die von dem Auktionshaus kommen, als korrekt geprüft deklarieren. Zudem würde das Applet auch selbst Dokumente für den Kunden signieren. Auch hier kann der Kunde einen Missbrauch nicht ausschließen. Den *Proxy* dagegen könnte der Kunde von einer unabhängigen und vertrauenswürdigen dritten Instanz beziehen. Ein weiterer Nachteil von Applets, durch den eine Implementierung der benötigten Funktionalität erschwert, wenn nicht gar unmöglich gemacht wird, sind die eingeschränkten Rechte von Applets. So können Applets beispielsweise nicht auf Dateiebene oder auf Speicherbereiche außerhalb des für die reservierten Bereichs zugreifen, und sie können keine nativen Methoden aufrufen. Dieses Problem ließe sich zwar durch die Verwendung signierter Applets umgehen, da signierte Applets mit zusätzlichen Rechten ausgestattet werden können, jedoch werden signierte Applets nicht von allen Browsern unterstützt.

Der *Proxy* empfängt signierte XML - Dokumente von dem Auktionshaus und validiert diese. Anschließend führt er eine XSLT - Transformation durch, und wandelt die XML - Dokumente in HTML - Dokumente um, die er an den Browser weiter sendet. Im Browser werden nicht nur die Dokumente selbst, sondern auch das Ergebnis der Validierung dargestellt. Beispiele für diese Darstellung zeigen die Abbildungen 8.2 und 8.3. Bei der Transformation kommen im Prinzip die in Abschnitt 6.2 vorgestellten Sty-

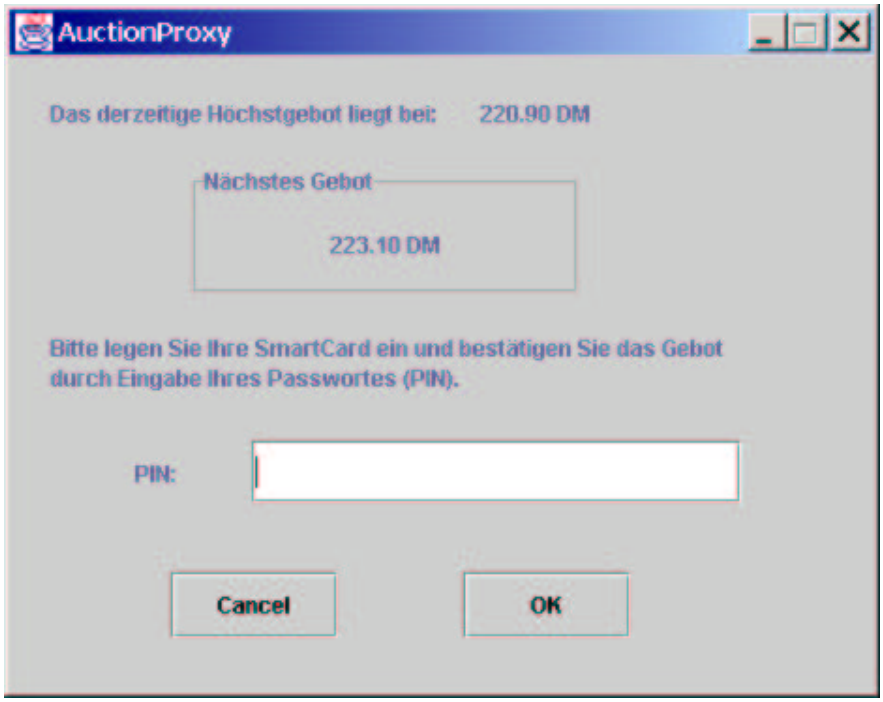

Abbildung 8.6: Dialogbox zur Eingabe des Passworts (Screenshot)

lesheets zum Einsatz. Diese Stylesheets können, genauso wie der *Proxy* selbst, von einer vertrauenswürdigen Instanz geliefert werden. Der Benutzer hat die Möglichkeit, zwischen zwei Darstellungsarten für XML - Signaturen zu wählen, einer minimalen Ansicht, die nur die nötigsten Informationen zeigt, und einer detaillierten Ansicht, die die komplette Struktur des XML - Dokuments anzeigt. Normale HTML - Dokumente, die der *Proxy* von dem Auktionshaus empfängt, werden nicht weiterverarbeitet, sondern direkt an den Browser weitergeleitet.

Bei der Abgabe eines Gebots ist es notwendig, dass der *Proxy* für den Kunden signierte XML - Dokumente generiert. Er muss im zweiten Schritt der Gebotsabgabe das Gebot erzeugen und im letzten Schritt die Bestätigung. Bevor der *Proxy* ein Dokument signiert, wird der Benutzer aufgefordert, sein Passwort zum Zugriff auf die *Smartcard* einzugeben. Durch die Verwendung von *SECUDE* als zugrunde liegende *Crypto - Engine* war es möglich, eine *Smartcard* - Anbindung zu schaffen. Das bedeutet, das Schlüsselpaar des Benutzers wird auf einer *Smartcard* gespeichert. Die Eingabe des Passworts erfolgt nicht über ein Formular, das im Browser dargestellt wird. Stattdessen öffnet der *Proxy* selbst eine Dialogbox zur Eingabe des Passworts. Dadurch ist das Passwort zusätzlich gesichert, da

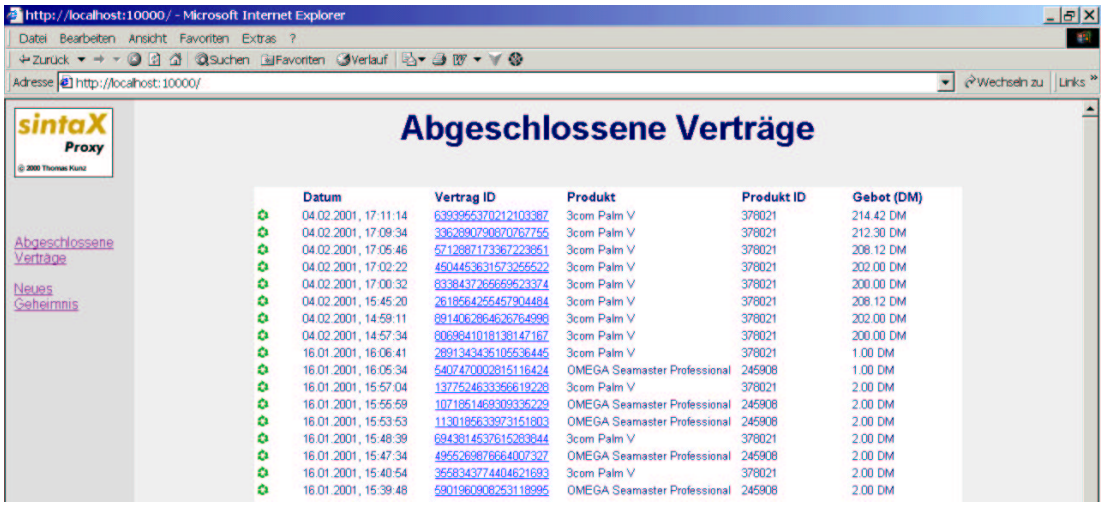

Abbildung 8.7: Index der abgegebenen Gebote (Screenshot)

es nicht vom Browser zum *Proxy* übertragen werden muss. Abbildung 8.6 zeigt die Dialogbox vor der Generierung des Gebots. Es wird auch die Höhe des Gebots angezeigt, sodass der Benutzer durch die Eingabe seines Passworts auch den Betrag bestätigt.

Nachdem der *Proxy* ein signiertes XML - Dokument erzeugt hat, sendet er es jedoch nicht sofort an das Auktionshaus. Zunächst findet eine XSLT - Transformation statt, und das Dokument wird dem Benutzer im Browser präsentiert. Dieser Zwischenschritt ist wichtig, damit der Benutzer sieht, was er signiert hat. Er kann das Dokument in Ruhe prüfen und hat die Gelegenheit, gegebenenfalls den gesamten Vorgang abzubrechen. Erst wenn der Benutzer explizit auf einen Button klickt, gibt er das Dokument frei, und es wird an das Auktionshaus gesendet. Dieser Schritt stellt zugleich auch eine wichtige Anforderung des Signaturgesetzes dar.

Eine weitere wichtige Funktion, die der Proxy erfüllt, ist das automatische Abspeichern der erfolgreich abgegebenen Gebote. Ein Gebot wurde erfolgreich abgegeben, wenn der Benutzer die Bestätigung im letzten Schritt des Protokolls gesendet hat. Diese Bestätigung enthält alle relevanten Informationen und wird nach dem Senden automatisch lokal beim Benutzer abgelegt. Zudem wird ein Index in Form eines HTML - Dokuments generiert, über den der Benutzer bequem über den Browser alle in der Vergangenheit abgegebenen Gebote öffnen kann (Abbildung 8.7).

Abbildung 8.8: Benutzerschnittstelle für den Proxy (Screenshot)

Die Steuerung des *Proxys* erfolgt über die Parameter der angeforderten URL in einem *HTTP - Requests*. Der *Proxy* wertet für jeden ankommenden *Request* die Liste der Parameter aus, und erkennt daran, welche Aktionen durchzuführen sind. Ein typische URL könnte beispielsweise wie folgt aussehen.

http://localhost/servlets/auction.servlet.AuctionServletBid
?userid=kunz&product=378021&action=makebid&
lastbid=220.90&nextbid=223.10

Der für den Proxy in erster Linie relevante Parameter ist *action*. Er besagt, welche Aktion durchzuführen ist, der Wert *makebid* beispielsweise kennzeichnet, das der Benutzer ein Gebot abgeben möchte und der Proxy daher ein entsprechendes Dokument zu generieren hat. Die Parameter *lastbid* und *nextbid* enthalten die Höhe des zuletzt abgegebenen Gebots und des von dem Benutzer abzugebenden Gebots.

Zum Starten des Proxys wurde eine graphische Benutzeroberfläche entwickelt, die zudem der Ausgabe diverser Meldungen (Debug - und Fehlermeldungen) dient (Abbildung 8.8).

Der derzeitige Entwicklungsstand von *sintaX* dient in erster Linie als Demonstrator. Er eignet sich besonders zum Einsatz auf einem Einzelplatzrechner, wobei jedoch an die bereits erwähnten Hardware - Anforderungen zu denken ist. Mit *sintaX* lässt sich demonstrieren, wie sich eine sichere Online - Auktion über das Internet realisieren lässt, die zudem sehr einfach und intuitiv zu bedienen ist. Zudem zeigt der Demonstrator, welches Potential XML und digitale Signaturen in E - Commerce - Szenarien bieten.

Kapitel 9

Fazit und Ausblick

In dieser Arbeit wurde mit der Spezifikation für XML - Signaturen eine vielversprechende Erweiterung von XML vorgestellt. Mit dieser Spezifikation ist es möglich, die hohen Sicherheitsanforderungen, die an ein Datenformat für den Einsatz in E - Business - Szenarien gestellt werden, zu lösen. Damit wird das große Potential, das XML als universelles Datenformat hat, eindrucksvoll unterstrichen. Der Vorteil dieser Spezifikation ist ihre universelle Anwendbarkeit. Sie ist nicht an bestimmte Szenarien, wie etwa Finanztransaktionen, gebunden, sondern lässt sich in beliebigen Szenarien einsetzen.

Insbesondere wurde gezeigt, wie sich das BAKO - Protokoll zum Erreichen einer gegenseitigen Nichtabstreitbarkeit mit Hilfe von XML - Signaturen umsetzen lässt. Somit stellt die eXtensible Markup Language XML ein geeignetes Datenformat zur Darstellung verbindlicher Verträge dar.

Wie in Kapitel 6 dargelegt, stellt das Problem der Präsentation digitaler Signaturen einen wichtigen Aspekt im Zusammenhang mit digitalen Signaturen dar. Es wurde eine mögliche Lösung zur Darstellung von XML - Signaturen präsentiert, die auf der XSLT - Transformation beruht. XSLT - Transformationen lösen das Präsentationsproblem zwar nicht hundertprozentig, können aber zumindest als wichtige Schritte in die richtige Richtung gelten. Es stellt sich zudem die Frage, ob es überhaupt möglich sein wird, das Präsentationsproblem jemals vollständig zu lösen, da man beispielsweise immer auf bestimmte Hardware - Eigenschaften angewiesen sein wird.

Mit der Implementierung der Online - Auktion *sintaX* wurde eine Anwendung für XML - Signaturen geschaffen, die zeigt, dass XML - Signaturen auch für den praktischen Einsatz bestens geeignet sind. *sintaX* de-

monstriert außerdem, wie eine Online - Auktion sicher gestaltet werden kann. Das derzeitige Sicherheitskonzept von *sintaX* kann jedoch noch erweitert werden. Daten, die zwischen dem Benutzer und dem Auktionshaus ausgetauscht werden, sollten mit Hilfe von SSL verschlüsselt werden. Die Anmeldung an das Auktionshaus kann durch eine Zwei - Wege - Authentifizierung mit digitalen Signaturen noch sicherer gestaltet werden. Zudem findet in dem derzeit implementierten Prototyp keine Prüfung der Zertifikate statt. Es werden stattdessen selbstsignierte Zertifikate verwendet, da keine PKI aufgebaut wurde. Auf der Seite des Benutzers würde eine Authentifizierung dem Proxy gegenüber für zusätzliche Sicherheit sorgen, da ansonsten unbefugte Personen auf die von dem Benutzer abgegebenen Gebote zugreifen können. Zusätzlich sollten die lokal gespeicherten Gebote verschlüsselt abgelegt werden, um sie vor unbefugtem Zugriff zu schützen. Dies würde es zudem ermöglichen, dass ein Proxy von mehreren Benutzern verwendet wird.

Prinzipiell stellt die Verwendung eines Proxys als Signierwerkzeug jedoch einen vielversprechenden und richtungsweisenden Ansatz zur Etablierung digitaler Signaturen im World Wide Web dar. Die „Proxy - Lösung“ ist gerade im Zusammenspiel mit XML / XSLT ein guter Schritt in Richtung einer für den Benutzer einfachen und intuitiven Anwendung und Darstellung von digitalen Signaturen. Diese Lösung verdeutlicht aber auch, dass die Bedienung durch den Einsatz digitaler Signaturen eher komplizierter wird, und zusätzliche Aktionen von dem Benutzer erfordert, wenn die nötigen Sicherheitsanforderungen im Umgang mit digitalen Signaturen erfüllt sein sollen. Deutlich wird dies an dem vorgestellten Protokoll zur Abgabe von Geboten. Die vier notwendigen Schritte zur Erzielung einer gegenseitigen Verbindlichkeit plus der Zwischenschritte zur Darstellung der vom Benutzer erzeugten Dokumente und der damit verbundenen Bestätigungen und Passworteingaben erscheinen doch etwas umständlich.

Der Nachteil dieser Lösung ist, dass sich der Benutzer zusätzliche Software installieren muss. Durch die Benutzung der Software können beispielsweise zusätzliche Anschaffungskosten für den Benutzer entstehen. Die gegenwärtige Implementierung des Proxys ist stark auf das Auktionsszenario zugeschnitten. Noch ist fraglich, wie universell solch ein Proxy gestaltet werden kann, sodass er mit verschiedenen E - Commerce - Anbietern, deren Nachrichtenaustausch auf XML - Signaturen basiert, kommunizieren kann. Es wäre jedenfalls für den Benutzer nicht zumutbar, sich im Extremfall für jeden Online - Shop oder jedes Online - Auktionshaus eine gesonderte Software installieren zu müssen.

Prinzipiell sollte es aber möglich sein, den Proxy an weitere Handelsszenarien, insbesondere an weitere Auktionstypen anzupassen. Als Ausblick auf zukünftige Aktivitäten in diese Richtung dient das in *[TSF01]* vorgestellte Protokoll einer versiegelten Auktion. Die Umsetzung dieses Auktionstyps stellt aufgrund der gegenüber einer Englischen Auktion noch höheren Sicherheitsanforderungen - Gebote müssen verdeckt übertragen werden, und der Auktionator darf vor Ablauf der Auktion nicht in der Lage sein, die Gebote aufzudecken - eine besondere Herausforderung dar. Im Rahmen des EU - Projekts *eBroker* ist eine Implementierung dieses Protokolls geplant, in die die Ergebnisse der vorliegenden Arbeit einfließen werden und auf denen aufgebaut werden kann.

Anhang A

BAKO - XML - Dokumente

XML - Code des *Offer* - Dokuments:

```
[ 1] <dsig:Signature xmlns:dsig="http://www.w3.org/2000/07/xmldsig">
[ 2]   <dsig:SignedInfo>
[ 3]     <dsig:CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000710"/>
[ 4]     <dsig:SignatureMethod
    Algorithm="http://www.w3.org/2000/07/xmldsig#rsa-sha1"/>
[ 5]     <dsig:Reference
    URI="http://www.merchant.com/contracts/1045678130013366886.xml">
[ 6]       <dsig:Transforms>
[ 7]         <dsig:Transform
    Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000710"/>
[ 8]       </dsig:Transforms>
[ 9]       <dsig:DigestMethod
    Algorithm="http://www.w3.org/2000/07/xmldsig#sha1"/>
[10]       <dsig:DigestValue>
[11]         wKuIZi/gJBzF5cAlrH/PK+DWqUM=
[12]       </dsig:DigestValue>
[13]     </dsig:Reference>
[14]     <dsig:Reference URI="#tsoffer1045678130013366886">
[15]       <dsig:DigestMethod
    Algorithm="http://www.w3.org/2000/07/xmldsig#sha1"/>
[16]       <dsig:DigestValue>
[17]         86nzSs7uVrykFlNU++fNIxT9TMk=
[18]       </dsig:DigestValue>
[19]     </dsig:Reference>
[20]   </dsig:SignedInfo>
[21]   <dsig:SignatureValue>
[22]     EzxngI6RHCU7ym4/ekUqxIJKM4dcu2IuxXSpDYRQ7OqB94
[23]     xwV6i74kBMDCBvWZ3HyHYmdPGxtklEqcDowCxzcxFrSASY
[24]     ytL/7bGFCwzAcMGfrRoLppddtHWHi969oO5O0ypNwe0=
[25]   </dsig:SignatureValue>
[26]   <dsig:KeyInfo>
[27]     <dsig:X509Data>
[28]       <dsig:X509IssuerSerial>
[29]         <dsig:X509IssuerName>
```

```
[30]           CN=Merchant, O=merchant.com, C=DE
[31]         </dsig:X509IssuerName>
[32]         <dsig:X509SerialNumber>
[33]           414194
[34]         </dsig:X509SerialNumber>
[35]       </dsig:X509IssuerSerial>
[36]       <dsig:X509SubjectName>
[37]         O=merchant.com, C=DE
[38]       </dsig:X509SubjectName>
[39]       <dsig:X509Certificate>
[40]         MIIBwjCCAS8CAwZR8jAJBgUrDgMCHQUAMDcxCzAJBgN
[41]         ...
[42]         Veop45BM7pFYn+8+QurZ+ZkTlHg8KA==
[43]       </dsig:X509Certificate>
[44]     </dsig:X509Data>
[45]    </dsig:KeyInfo>
[46]   <dsig:Object Id="tsoffer1045678130013366886">
[47]     <BAKO:Bako xmlns:BAKO="http://www.gmd.de/BAKO">
[48]       <BAKO:Offer>
[49]         <ts:Timestamp xmlns:ts="http://www.gmd.de/Timestamp">
[50]           <ts:Date>
[51]             13.08.2000
[52]           </ts:Date>
[53]           <ts:Time>
[54]             11:34:11
[55]           </ts:Time>
[56]         </ts:Timestamp>
[57]       </BAKO:Offer>
[58]     </BAKO:Bako>
[59]   </dsig:Object>
[60] </dsig:Signature>
```

XML - Code des *Accept* - Dokuments:

```
[ 1] <dsig:Signature xmlns:dsig="http://www.w3.org/2000/07/xmldsig">
[ 2]   <dsig:SignedInfo>
[ 3]     <dsig:CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000710"/>
[ 4]     <dsig:SignatureMethod
           Algorithm="http://www.w3.org/2000/07/xmldsig#rsa-sha1"/>
[ 5]     <dsig:Reference URI="#offer10456781300133668886">
[ 6]       <dsig:Transforms>
[ 7]         <dsig:Transform
    Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000710"/>
[ 8]       </dsig:Transforms>
[ 9]       <dsig:DigestMethod
    Algorithm="http://www.w3.org/2000/07/xmldsig#sha1"/>
[10]       <dsig:DigestValue>
[11]         pIOM/xUwqiHb+8IVdF0kJV1ANDE=
[12]       </dsig:DigestValue>
[13]     </dsig:Reference>
[14]     <dsig:Reference URI="#tsaccept10456781300133668886">
[15]       <dsig:DigestMethod
    Algorithm="http://www.w3.org/2000/07/xmldsig#sha1"/>
[16]       <dsig:DigestValue>
[17]         anTSS+cw0xM2gD7SriY6yoa6oWY=
[18]       </dsig:DigestValue>
[19]     </dsig:Reference>
[20]   </dsig:SignedInfo>
[21]   <dsig:SignatureValue>
[22]      ...
[23]   </dsig:SignatureValue>
[24]   <dsig:KeyInfo>
[25]      ...
[26]   </dsig:KeyInfo>
[27]   <dsig:Object Id="tsaccept10456781300133668886">
[28]     <BAKO:Bako xmlns:BAKO="http://www.gmd.de/BAKO">
[29]       <BAKO:Accept>
[30]         <ts:Timestamp xmlns:ts="http://www.gmd.de/Timestamp">
[31]           <ts:Date>
[32]             13.08.2000
[33]           </ts:Date>
[34]           <ts:Time>
[35]             11:34:21
[36]           </ts:Time>
[37]         </ts:Timestamp>
[38]       </BAKO:Accept>
[39]     </BAKO:Bako>
[40]   </dsig:Object>
[41]   <dsig:Object Id="offer10456781300133668886">
[42]     <dsig:Signature
    xmlns:dsig="http://www.w3.org/2000/07/xmldsig">
[43]       ...
[44]       <dsig:Object Id ="tsoffer10456781300133668886">
```

```
[45]         <BAKO:Bako xmlns:BAKO="http://www.gmd.de/BAKO">
[46]           <BAKO:Offer>
[47]             ...
[48]           </BAKO:Offer>
[49]         </BAKO:Bako>
[50]       </dsig:Object>
[51]       ...
[52]     </dsig:Signature>
[53]   </dsig:Object>
[54] </dsig:Signature>
```

XML - Code des *Confirm* - Dokuments:

```
[ 1] <dsig:Signature xmlns:dsig="http://www.w3.org/2000/07/xmldsig">
[ 2]   <dsig:SignedInfo>
[ 3]     ...
[ 4]     <dsig:Reference URI="#accept1045678130013366886">
[ 5]       ...
[ 6]     </dsig:Reference>
[ 7]     <dsig:Reference URI="#tsconfirm1045678130013366886">
[ 8]       ...
[ 9]     </dsig:Reference>
[10]   </dsig:SignedInfo>
[11]   <dsig:SignatureValue>
[12]     ...
[13]   </dsig:SignatureValue>
[14]   <dsig:KeyInfo>
[15]     ...
[16]   </dsig:KeyInfo>
[17]   <dsig:Object Id="tsconfirm1045678130013366886">
[18]     <BAKO:Bako xmlns:BAKO="http://www.gmd.de/BAKO">
[19]       <BAKO:Confirm>
[20]         <ts:Timestamp xmlns:ts="http://www.gmd.de/Timestamp">
[21]           ...
[22]         </ts:Timestamp>
[23]       </BAKO:Confirm>
[24]     </BAKO:Bako>
[25]   </dsig:Object>
[26]   <dsig:Object Id="accept1045678130013366886">
[27]     <dsig:Signature
    xmlns:dsig="http://www.w3.org/2000/07/xmldsig">
[28]       ...
[29]       <dsig:Object Id="offer1045678130013366886">
[30]         <dsig:Signature
    xmlns:dsig="http://www.w3.org/2000/07/xmldsig">
[31]           ...
[32]         </dsig:Signature>
[33]       </dsig:Object>
[34]       ...
[35]     </dsig:Signature>
[36]   </dsig:Object>
[37] </dsig:Signature>
```

Anhang B

Beschreibung einer Auktion in XML

Das folgende Beispiel zeigt ein XML - Dokument, das eine Auktion beschreibt, und die dazugehörende DTD.

```
[ 1] <!DOCTYPE Product [
[ 2]   <!ELEMENT Product (ProductID, Name, Description, StartBid,
       LastBid, AuctionStart, AuctionEnd, InternalStart, InternalEnd,
   Status, Rules, BidList)>
[ 3]   <!ELEMENT ProductID (#PCDATA)>
[ 4]   <!ELEMENT Name (#PCDATA)>
[ 5]   <!ELEMENT Description (#PCDATA)>
[ 6]   <!ELEMENT StartBid (#PCDATA)>
[ 7]   <!ATTLIST StartBid currency CDATA #REQUIRED>
[ 8]   <!ELEMENT LastBid (#PCDATA)>
[ 9]   <!ATTLIST LastBid currency CDATA #REQUIRED>
[10]   <!ELEMENT AuctionStart (#PCDATA)>
[11]   <!ELEMENT AuctionEnd (#PCDATA)>
[12]   <!ELEMENT InternalStart (#PCDATA)>
[13]   <!ELEMENT InternalEnd (#PCDATA)>
[14]   <!ELEMENT Rules (#PCDATA)>
[15]   <!ELEMENT Status EMPTY>
[16]   <!ATTLIST Status status CDATA #REQUIRED>
[17]   <!ELEMENT BidList (Bid)*>
[18]   <!ELEMENT Bid (UserID, BidID, ContractID, Amount, Confirmation)>
[19]   <!ELEMENT UserID EMPTY>
[20]   <!ATTLIST UserID Id CDATA #REQUIRED>
[21]   <!ELEMENT BidID EMPTY>
[22]   <!ATTLIST BidID Id CDATA #REQUIRED>
[23]   <!ELEMENT ContractID EMPTY>
[24]   <!ATTLIST ContractID Id CDATA #REQUIRED>
[25]   <!ELEMENT Amount (#PCDATA)>
[26]   <!ATTLIST Amount currency CDATA #REQUIRED>
[27]   <!ELEMENT Confirmation EMPTY>
[28]   <!ATTLIST Confirmation confirmed CDATA #REQUIRED>
[29] ]>
```

```
[ 1] <eAuction:Product xmlns:eAuction="http://www.eAuction.com">
[ 2]   <eAuction:ProductID>245908</eAuction:ProductID>
[ 3]   <eAuction:Name>OMEGA Seamaster Professional</eAuction:Name>
[ 4]   <eAuction:Description>
[ 5]     Eine ungebrauchte, originalverpackte
[ 6]     "OMEGA Seamaster Professional" in Top-Zustand.
[ 7]   </eAuction:Description>
[ 8]   <eAuction:StartBid currency="DM">1.00</eAuction:StartBid>
[ 9]   <eAuction:LastBid currency="DM">1.00</eAuction:LastBid>
[10]   <eAuction:AuctionStart>01.12.2000, 00:00:00</eAuction:AuctionStart>
[11]   <eAuction:AuctionEnd>16.01.2001, 16:10:00</eAuction:AuctionEnd>
[12]   <eAuction:InternalStart>975625200094</eAuction:InternalStart>
[13]   <eAuction:InternalEnd>979657800094</eAuction:InternalEnd>
[14]   <eAuction:Status status="1"/>
[15]   <eAuction:Rules>
[16]     Das hoechste bestaetigte Gebot gewinnt die Auktion.
[17]   </eAuction:Rules>
[18]   <eAuction:BidList>
[19]     <eAuction:Bid>
[20]       <eAuction:UserID Id="kunz"/>
[21]       <eAuction:BidID Id="BKovNVufcj8IniMOYI0Duw=="/>
[22]       <eAuction:ContractID Id="5407470002815116424"/>
[23]       <eAuction:Amount Currency="DM">1.00</eAuction:Amount>
[24]       <eAuction:Confirmation confirmed="yes"/>
[25]     </eAuction:Bid>
[26]   </eAuction:BidList>
[27] </eAuction:Product>
```

Literaturverzeichnis

[ABC00] S. Adler, A. Berglund, J. Caruso [u.a.]: *Extensible Stylesheet Language (XSL) Version 1.0*, W3C Candidate Recommendation 21 November 2000.
<http://www.w3.org/TR/xsl>

[BG00] M. Benning, T. Gerber: *Kein Zuschlag mit dem virtuellen Hammer - Grundsatzurteil zu Online - Auktionen* -Hannover: Verlag Heinz Heise, c't 3/2000.

[BL96] T. Berners-Lee: *Uniform Resource Identifiers (URI): Generic Syntax*, RFC 2396.
<http://www.ietf.org/rfc/rfc2396.txt>

[Boy01] J. Boyer: *Canonical XML*, W3C Proposed Recommendation 19 January 2001.
<http://www.w3.org/TR/2001/PR-xml-c14n-20010119/>

[BHL99] T. Bray, D. Hollander, A. Layman: *Namespaces in XML*, World Wide Web Consortium 14 January 1999.
<http://www.w3.org/TR/REC-xml-names/>

[BPS98] T. Bray, J. Paoli, C. M. Sperberg-McQueen: *Extensible Markup Language (XML)*, W3C Recommendation 10 Februar 1998.
<http://www.w3.org/TR/1998/REC-xml-19980210/>

[Cac99] C. Cachin: Efficient Private Bidding and Auctions with an Oblivious Third Party - IBM Research Report RZ 3131, 1999.

[Cla99] J. Clark: *XSL Transformations (XSLT)*, W3C Recommendation 16 November 1999.
<http://www.w3.org/TR/1999/REC-xslt-19991116/>

[CR99] J. Clark, S. DeRose: *XML Path Language (XPath)*, W3C Recommendation 16 November 1999.
<http://www.w3.org/TR/1999/REC-xpath-19991116/>

[DAP01] E. Dans, D. B. Allen, N. S. Parker: *Existing Business Models for Brokering and their Adaptation to Electronic Markets* - Proc. to BitWorld Conference, El Cairo, 2001.

[DH76] W. Diffie, M. Hellman: *New Directions in Cryptography*, IEEE Transactions on Information Theory, IT-22(6), November 1976.

[ERS00] D. Eastlake, J. M. Reagle, D. Solo: *XML-Signature Syntax and Processing*, W3C Candidate Recommendation 31-October-2000.
<http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/>

[EBrok] *Projekt eBroker: Electronic Brokering Services for Open Trading Infrastructures*.
<http://www.darmstadt.gmd.de/oasys/projects/ebroker/index.html>

[Fal00] D. C. Fallside: *XML Schema Part 0: Primer*, W3C Candidate Recommendation 24 October 2000.
<http://www.w3.org/TR/2000/CR-xmlschema-0-20001024/>

[FR96] M. K. Franklin, M. K. Reiter: *The Design and Implementation of a Secure Auction Service* - IEEE Transactions on Software Engineering Vol. 22(5), 1996.

[FKK96] A. O. Freier, P. Karlton, P. C. Kocher: *The SSL Protocol Version 3.0*, Internet Draft, 1996.
<http://home.netscape.com/eng/ssl3>

[GKW97] P. Glöckner, S. Kolletzki, M. Wichert: *Signed Unique References - A BAKO Extension Proposal* - Proceedings of the 8th Joint European Networking Conference JENC8, Edinburgh, June 1997.

[Har99] E. R. Harold: *XML Bible* - Chicago [u.a.]: IDG Books Worldwide, 1999.
<ISBN: 0764532367>

[HZ00] M. Herschlag, R. Zwick: *Internet Auctions - Popular and Professional Literature Review* - Quarterly Journal of Electronic Commerce, 1(2), 2000.
<http://home.ust.hk/ mkzwick/download.html>

[HC98] J. Hunter, W. Crawford: *Java Servlet Programming* - Sebastopol [u.a.]: O'Reilly, 1998.
<ISBN: 156592391>

[Kle00] S. Klein: *The Emergence of Auctions on the World Wide Web* - In: *Handbook of Electronic Commerce*, Berlin [u.a]: Springer, 2000.

[Klm00] P. Klemperer: *What Really Matters in Auction Design* - Nuffield College, Oxford University, September 2000.
<http://www.nuff.ox.ac.uk/economics/people/klemperer.htm>

[Kol96] S. Kolletzki: *Secure Internet Banking with Privacy Enhanced Mail - A Protocol for Reliable Exchange of Secured Order Forms (BAKO)* - In: *Computer Networks and ISDN Systems 28*, Elsevier Science Publishers B.V. (North Holland), 1996.

[KF98] M. Kumar, S. I. Feldman: *Internet Auctions* - In: *3rd USENIX Workshop on Electronic Commerce*, USENIX Press 1998.

[Lee97] H. G. Lee: *AUCNET: Electronic Intermediary for Used-Car Transactions* - Andover: Routledge Subscriptions, Electronic Markets (EM) 4/1997.

[Mac97] I. Macherius: *Revolution der Experten - XML: Professionelle Alternative zu HTML* - Hannover: Verlag Heinz Heise, iX 6/1997.

[MTU99] H. Maruyama, K. Tamura, N. Uramoto: *XML and Java - Developing Web Applications* - Reading [u.a.]: Addison Wesley Longman, 1999.
<ISBN: 0201485435>

[May00] S. Mayer: *Bewertender Vergleich von Homebanking - Verfahren (Diplomarbeit)* - J. W. Goethe Universität Frankfurt / M., 10. Juli 2000.
<http://sven.mayer.net/university/diplom/>

[MOV97] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone: *Handbook of Applied Cryptography* - Boca Raton, Florida: CRC Press, 1997.
<ISBN: 0849385237>

[NS01] M. Niesing, K. Schmeh: *Schlüssel des Vertrauens - Digitale Ausweise im Internet* -Hannover: Verlag Heinz Heise, c't 4/2001.

[NPS99] M. Naor, B. Pinkas, R. Sumner: *Privacy Preserving Auctions and Mechanism Design* - ACM Conference on Electronic Commerce, 1999.
<http://www.wisdom.weizmann.ac.il/ naor/onpub.html>

[Oak98] S. Oaks: *Java Security* - Sebastopol [u.a.]: O'Reilly, 1998.
<ISBN: 1565924037>

[Por00] U. Pordesch: *Der fehlende Nachweis der Präsentation signierter Daten* - Wiesbaden: Verlag Vieweg & Sohn, Datenschutz und Datensicherheit (DuD) 2/2000.

[Rea96] J. M. Reagle Jr.: *Trust in a Cryptographic Economy and Digital Security Deposits: Protocols and Policies (Master's thesis)* - Massachusetts Institute of Technology, May 10, 1996.
<http://reagle.org/joseph/commerce/thesis/thesis.html>

[Reb00] M. Rebstock: *Elektronische Geschäftsabwicklung, Märkte und Transaktionen - Eine methodische Analyse* -HMD Praxis der Wirtschaftsinformatik 37 (2000), Nr. 215

[Rec97] M. Reck: *Trading-Process Characteristics of Electronic Auctions* - Andover: Routledge Subscriptions, Electronic Markets (EM) 4/1997.

[Rei95] M. K. Reiter: *The Rampart Toolkit for Building High-Integrity Services* - In: *Theory and Practice in Distributed Systems*, Berlin [u.a]: Springer, 1995.

[RSA78] R. L. Rivest, A. Shamir, L. M. Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM, 21(2), Februar 1978.

[RSW96] R. L. Rivest, A. Shamir, D. A. Wagner: *Time-lock puzzles and timed-release Crypto (MIT/LCS/TR-694)* - Massachusetts Institute of Technology, 21. Februar 1996.
<http://ncstrl.mit.edu/Dienst/Repository/2.0/Body/ncstrl.mit_lcs%2fMIT%2fLCS%2fTR-684/postscript>

[Sch00] A. U. Schmidt: *Signiertes XML und das Präsentationsproblem* - Wiesbaden: Verlag Vieweg & Sohn, Datenschutz und Datensicherheit (DuD) 3/2000.

[BS96] B. Schneier: *Applied Cryptography, 2nd Edition* - New York [u.a.]: John Wiley & Sons, 1996.
<ISBN: 0471117099>

[SH00] C. Schulzki-Haddouti: *Karten neu gemischt - Neue Chancen für die Chipkarte* -Hannover: Verlag Heinz Heise, c't 25/2000.

[SigG01] *Gesetz über Rahmenbedingungen für elektronische Signaturen (Signaturgesetz - SigG)* - 2001.
<http://www.dud.de/dud/documents/sigg010214.pdf>

[SigG97] *Gesetz zur digitalen Signatur (Signaturgesetz - SigG)* - 1997.
<http://www.netlaw.de/gesetze/sigg.htm>

[SigV97] *Verordnung zur digitalen Signatur (Signaturverordnung - SigV)* - 1997.
<http://www.netlaw.de/gesetze/sigv.htm>

[Sin99] S. Singh: *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography* - London: Doubleday, 1999.
<ISBN: 0385495315>

[Sha79] A. Shamir: *How to Share a Secret* - Communications of the ACM, 22, 11/1979.

[SA99] F. Stajano, R. Anderson: *The Cocaine Auction Protocol: On The Power Of Anonymous Broadcast* - In: A. Pfitzmann, ed. *Proceedings of Information Hiding Workshop 1999*, Berlin [u.a]: Springer, 1999.

[SS99] S. G. Stubblemine, P. F. Syverson: *Fair On-Line Auctions without Special Trusted Parties* - In: *Financial Cryptography*, Berlin [u.a]: Springer, 1999.

[TSF01] O. A. Tafreschi, M. Schneider, P. Fankhauser, B. Mahleko, T. Tesch: *From Call for Tenders to Sealed-Bid Auction for Mediated eCommerce* - Proc. of the 9th IFIP 2.6 Working Conference on Database Semantics, Hong Kong 2001.

[Tan96] A. S. Tanenbaum: *Computer Networks, 3rd Edition* - Upper Saddle River, New Jersey: Prentice Hall, 1996.
<ISBN: 0133499456>

[Tur97] E. Turban: *Auctions and Bidding on the Internet: An Assessment* - Andover: Routledge Subscriptions, Electronic Markets (EM) 4/1997.

[Wri97] C. D. Wrigley: *Desgin Criteria For Electronic Market Servers* - Andover: Routledge Subscriptions, Electronic Markets (EM) 4/1997.

[XHTML] *XHTML 1.0: The Extensible Hypertext Markup Language. A Reformulation of HTML 4 in XML 1.0*, W3C Recommendation 26 January 2000.
<http://www.w3.org/TR/2000/REC-xhtml1-20000126/>