Technische Universität Berlin
Fakultät IV für Elektrotechnik und Informatik

# Sicherer Dokumentenaustausch im E-Government mit OSCI und Acrobat/PDF

Diplomarbeit

Manuel Antonio Aldana
Matr.-Nr.: 204430

Betreuer

Prof. Radu Propescu-Zeletin
Uwe Holzmann-Kaiser

Ich versichere hiermit an Eides statt, dass ich die von mir am heutigen Tag eingereichte Diplomarbeit selbständig und eigenhändig verfasst und ausschließlich die angegebenen Hilfsmittel benutzt habe.

Berlin, den 22.02.2007 _____

Manuel Aldana

# Danksagungen

Ich möchte mich bei allen Beteiligten des eGovernment-Labors am Fraunhofer Institut FOKUS, insbesondere Herrn Holzmann-Kaiser und Herrn Martin bedanken, die mir tatkräftige Unterstützung bei der Bereitstellung der technischen Umgebung gaben und gute Ratschläge für die schriftlichen Ausarbeitung parat hatten. Zudem gilt mein Dank an Prof. Radu Popescu-Zeletin, der Leiter des Lehrstuhls OKS ist und die Diplomarbeit seitens der TU-Berlin im Bereich Informatik und dem Fraunhofer FOKUS ermöglichte. Auch möchte ich mich bei Adobe Systems in Person von Herrn Körner bedanken, die Support und die entsprechende Acrobat Software zur Verfügung stellten. Zuletzt, aber nicht minder wichtig gilt meine Danksagung an meine Familie und Freundin, die mich während meiner Diplomarbeit ebenfalls reichlich unterstützten.

# Inhaltsverzeichnis

***Zusammenfassung:***

*Innerhalb der Verwaltung soll mit E-Government eine Kostensenkung, eine Verbesserung der Dienstleistungsqualität und eine Beschleunigung der Abläufe erreicht werden. Allerdings kann das Erreichen dieser Ziele durch rechtliche und technische Hindernisse erschwert werden. So ist das Sicherheitsniveau der Verwaltungsvorgänge durch Gesetze und Richtlinien festgelegt und muss dementsprechend von E-Government Systemen umgesetzt werden. Auch sind technische Lösungen nötig, die sich möglichst leicht in die bestehende IT der Verwaltung integrieren lassen.*

*Im E-Government sollen Vorgänge auch dadurch modernisiert werden, dass Papierdokumente durch elektronische ersetzt werden. Diese elektronische Abwicklung wirft zwei zentrale Fragestellungen auf: Existieren Gesetze, die eine solche elektronische Kommunikation gleichwertig zur Schriftform aus Sicht der Form und Unterschrift akzeptieren und inwieweit können derzeitige technische Verfahren und Standards vorgeschriebene Richtlinien umsetzen?*

*Die rechtlichen Rahmenbedingungen ergeben sich direkt aus dem deutschen Öffentlichen Recht, in dem seit der Novellierung des Verwaltungsverfahrensgesetzes für den Verwaltungsakt standardmäßig die Möglichkeit existiert, die Schriftform durch die elektronische Form zu ersetzen. Allerdings gilt immer noch die Forderung, dass die Authentizität und Integrität der Nachricht gesichert sein muss. Diese Vorraussetzung wird durch den Einsatz der im Signaturgesetz eingeführten qualifizierten elektronischen Signatur erfüllt. Zudem werden für den Schutz der elektronisch zu übermittelnden personengebundenen Daten weitere Regelungen im Datenschutzrecht festgelegt.*

*Das Signaturgesetz und das Datenschutzrecht sind technikneutral verfasst, so dass, solange die rechtlichen Forderungen umgesetzt werden, beliebige Sicherheitstechnologien eingesetzt werden können. Dadurch sind kryptografische Techologien wie die Einweg-Hashfunktion, die symmetrische, asymmetrische bzw Hybridverschlüsselung interessant, da sie die Basis einer möglichen Umsetzung der rechtlichen Vorgaben bilden. So lässt sich eine Geheimhaltung durch eine Hybridverschlüsselung und die elektronische Signatur durch die Digitale Signatur realisieren. Da kryptografische Systeme, die Schlüssel verwenden, einer Authentizität der Schlüssel bedürfen, müssen zudem Maßnahmen für eine sichere Schlüsselinfrastruktur ergriffen werden, in denen Zertifikate eine wichtige Rolle spielen.*

*Um diese Anforderungen umzusetzen, wurde die OSCI (Online Services Computer Interface) Leitstelle mit der Entwicklung und Pflege des OSCI Standards beauftragt, der*

*die rechtlichen Vorgaben mit Sicherheitstechnologien verbindet und auf diese Weise eine Infrastruktureinheit schafft, die den Eintritt zu einer sicheren und rechtskräftigen Kommunikation in der öffentlichen Verwaltung erleichtert. OSCI definiert dazu ein offenes Nachrichtenformat, ein Protokoll und ein erweitertetes Rollenmodell, in dem u.a. der sogenannte Intermediär eine Vermittlerstelle einnimmt, dabei viele Dienste wie Protokollierung oder Zertifikatsüberprüfung übernimmt und die Komplexität auf Sender- und Empfängerseite reduziert. Ein weiterer Vorteil des OSCI Nachrichtenformats ist, dass zwischen Inhalts- und Nutzungsdaten unterschieden wird und dadurch die Geschäftsvorfalldaten und Transportdaten voneinander getrennt werden. OSCI setzt weiterhin auf offene XML-Standards und erlaubt wegen der Formatneutralität der Inhaltsdaten die Verwendung beliebiger Datenformate, wodurch die Integration in bestehende Systeme erleichtert wird.*

*Mit Adobe Acrobat existiert weiterhin eine weit verbreitete Applikation, die die vielfältigen Möglichkeiten des PDF Standards ausnutzt und deswegen für die öffentlichen Verwaltung im Zusammenhang mit elektronischen Dokumenten interessant ist. Hier bietet OPENLiMiT eine Acrobat Erweiterung an, mit der PDF Dokumente mit einer qualifizierten Signatur unterschrieben werden können, wodurch die Forderungen der elektronischen Form erfüllt werden. Dadurch ergibt sich eine interessante Beispielanwendung, in der zunächst ein PDF Dokument mit Acrobat als Inhaltsdatum digital signiert und in eine OSCI Nachricht integriert wird. Eine solche Acrobat OSCI Integrationslösung wurde bisher noch nicht entwickelt, stellt den letzten Baustein für einen E-Government gerechten PDF-Dokumentenaustausch dar und soll im Implementierungsteil der Arbeit umgesetzt werden. Dabei wird eine Integrationslösung entwickelt, in der eine Kommunikationsbrücke zwischen Acrobat und der OSCI Senderapplikation realisiert wird. Es wird das aktuell geöffnete PDF-Dokument gespeichert und von der Senderapplikation als Inhaltsdatum in eine OSCI Nachricht integriert, die innerhalb des Szenarios 'One-Way-Message, aktiver Empfänger' abgesendet wird.*

# 1 Einführung

## 1.1 Motivation der Aufgabenstellung

E-Government gewinnt in Deutschland zunehmend an Bedeutung[1]. Besonders die Geschäftsabwicklung mit elektronischen Dokumenten anstelle der Verwendung von Papier zeigt Potentiale. Um entsprechende IT Systeme in der öffentlichen Verwaltung erfolgreich umzusetzen und einzuführen, müssen dafür gesondert rechtliche und technische Rahmenbedingungen untersucht werden.

### 1.1.1 Betrachtung der rechtlichen Dimension

Im Gegensatz zur Privatwirtschaft, in der bspw. Firmen ihre internen Prozesse größtenteils nach eigenem Ermessen festlegen und definieren[2], geben bei den Behörden Bestimmungen und Gesetze vor, was die IT Systeme genau umsetzen müssen. In diesem Zusammenhang existieren gesetzliche Rahmenbedingungen, die das Anbieten von Verwaltungsdienstleistungen auf elektronischem Wege regeln.

Das Einbeziehen der rechtlichen Rahmenbedingungen in die IT Systeme ist deswegen so essentiell, da andernfalls durch ein Nichteinhalten von Gesetzen ein unrechtmäßiges Handeln seitens der Verwaltung vorliegen würde und dies weittragende Konsequenzen für die einzelnen Verwaltungsträger hätte.

### 1.1.2 Betrachtung der technischen Dimension

Rechtliche Rahmenbedingungen geben vor, was die IT Systeme umsetzen müssen. Die technischen Gegebenheiten müssen wiederum darauf untersucht werden, ob sie die rechtlichen Forderungen überhaupt realisieren können. Im Besonderen sind die aktuellen technischen Möglichkeiten von Interesse, die die Sicherheitsanforderungen der Gesetze erfüllen und im E-Government bei der elektronischen Geschäftsabwicklung potentiell zum Einsatz kommen dürfen.

In der öffentlichen Verwaltung und auch auf Bürgerseite werden für die Arbeit mit Do-

1 Dies lässt sich nur schwer quantitiv belegen. Allerdings zeigen Projekte wie Toll Collect, die Initiative BundOnline 2005 oder die aktuelle Capgemini Studie [Capgemini], dass die Durchdringung von E-Government forciert wird.

2 Natürlich existieren für Firmen auch abhängig von der Gesellschaftsform ebenfalls rechtliche Vorgaben, die die Prozesse beeinflussen (wie Datenschutz, Buchhaltungspflicht etc.). Diese sind im Gegensatz zum E-Government aber nicht der primäre Treiber für die Prozesse.

kumenten bereits gängige Applikationen verwendet. Um eine kostenintensive Einführung neuer Applikationen zu vermeiden, ist eine Integration von E-Government Lösungen in die bereits eingesetzten Applikationen empfehlenswert.

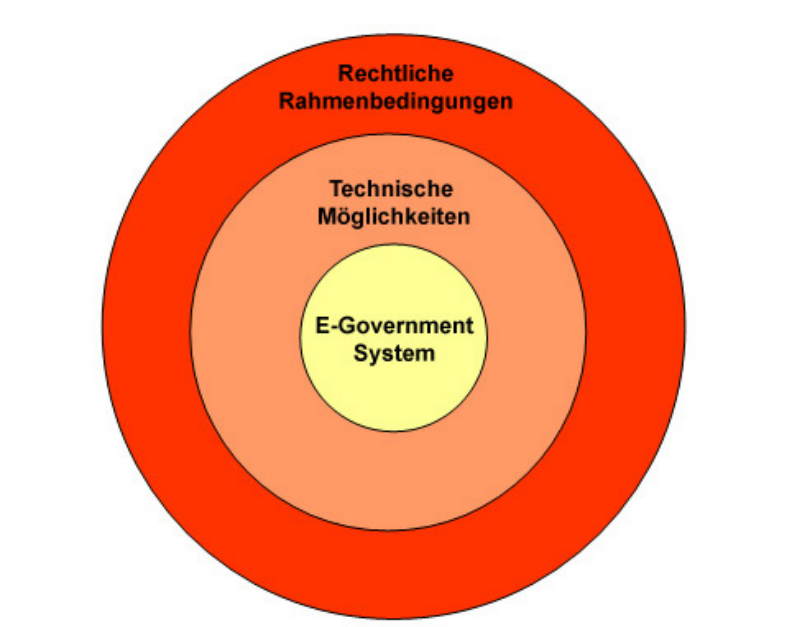


Abbildung 1: Die erste Barriere für E-Government stellen die rechtlichen Rahmenbedinungen dar. Im nächsten Schritt müssen schließlich technische Mittel existieren, um ein E-Government System umsetzen zu können.

## 1.2 Aufgabenstellung

Es soll zunächst untersucht werden, welche rechtlichen Vorraussetzungen für eine E-Government Geschäftsabwicklung mit elektronischen Dokumenten existieren und inwieweit Sicherheitsanforderungen dazu gestellt werden. Dazu wird im Speziellen die Legitimität eines elektronischen Dokumentenaustausches zwischen Behörde und Bürger im Verwaltungsverfahren betrachtet, um eine Alternative zur Schriftform zu haben. Darüberhinaus muss auf die Sicherheitsmechanismen eingegangen werden, die einen Austausch elektronischer Dokumente nach den rechtlichen Vorgaben hinsichtlich Datenschutz und elektronischer Signatur umsetzen können. Zudem soll mit OSCI (Online Services Computer Interface) ein Standard vorgestellt werden, der die Sicherheitsanforderungen der Gesetze beachtet und weitere Mehrwertdienste bietet.

Mit Adobe Acrobat existiert eine weit verbreitete Applikation, die vielfältige Möglichkeiten bei der Arbeit mit PDF Dokumenten bietet und durch ein zertifiziertes Plug-In zusätzlich eine qualifizierte Signatur ermöglicht. Dazu soll eine OSCI Integrationslösung

für Adobe Acrobat implementiert werden, die das aktuell geöffnete PDF Dokument als Inhaltsdaten verpackt und innerhalb eines OSCI Szenarios an den Intermediär absendet. Durch diesen entwickelten Prototyp soll gezeigt werden, dass eine Integration zwischen Acrobat und OSCI möglich ist und Acrobat im E-Government für einen sicheren elektronischen Dokumentenaustausch verwendet werden kann.

### 1.2.1 Abgrenzung der Arbeit

Für die erfolgreiche Einführung und den Betrieb von E-Government sind weiterhin die organisatorischen und soziologischen Dimensionen relevant. Auch die in der Arbeit vorzustellenden rechtlichen und technischen Betrachtungen zeigen nur einen Ausschnitt des gesamten Spektrums. Die Arbeit erhebt dementsprechend keinen Anspruch einer vollständigen Sicht auf die Herausforderungen im E-Government, sondern verwendet die technischen und rechtlichen Vorraussetzungen vielmehr für eine Bewertung eines elektronischen Dokumentenaustausches.

Der Implementierungsteil der Arbeit beschäftigt sich mit der prototypischen Entwicklung einer Integrationslösung, der die grundsätzlichen Möglichkeiten des Zusammenspiels zwischen Adobe Acrobat und OSCI zeigt. Daher wird mit One-Way-Message, aktiver Empfänger nur eines der drei von OSCI unterstützten Grundszenarien umgesetzt. Weiterhin werden für die Zertifikatdaten keine Signaturkarten sondern Softwarezertifikate verwendet, die rechtlich gesehen zu den fortgeschrittenen Zertifikaten gehören und als Dateien im entsprechenden Zertifikatformat auf der Festplatte gespeichert sind.

# 2 Rechtliche Rahmenbedingungen

## 2.1 Einleitung

Die öffentliche Verwaltung ist der exekutive Teil der Staatsmacht und darf ausschließlich nach existierenden Gesetzen handeln[1]. Im Zusammenhang einer Kommunikation mit elektronischen Dokumenten stellt sich die Frage, ob eine elektronische Abwicklung von Geschäftsvorfällen im Verwaltungsverfahren legitimiert ist. Hierzu sollen die Verwaltungsverfahrenstypen betrachtet werden, bei denen in erster Linie unterschriftspflichtige Papierdokumente verwendet werden. Hierzu soll geklärt werden, inwieweit eine elektronische Abwicklung im Verwaltungsverfahren legitimiert ist und ob speziell die Ablösung von Papierdokumenten durch elektronische Dokumente möglich ist.

**Elektronisches Dokument:** Ein elektronisches Dokument besitzt wie andere Medientypen zunächst ein Behälter und eine in diesem Behälter gekapselte Information. Die Besonderheit des elektronischen Dokuments ist, dass die Behälter- und Inhaltsdaten binär kodiert sind. Die Behältertypen sind dabei als Dateiformate vielfältig: Es kann sich konkret bspw. um Adobes PDF, Microsofts .doc, E-mail oder ASCII-Code handeln, die die Inhaltsinformationen nach ihren entsprechenden Formaten kodieren.

### 2.1.1 Akteure

Um zwischen einem IT System und ihren Anwendern zu unterscheiden, wird der Begriff des Akteurs als Anwender eingeführt. Im Zusammenhang von E-Government lässt sich diese Systemsicht folgendermaßen beschreiben:

Zunächst stellt die öffentliche Verwaltung, deren Einrichtungen im E-Government auch untereinander kommunizieren können, selbst einen Akteur dar. Zusätzlich existiert aus der Privatwirtschaft die Unternehmung, die in verschiedenen Rechtsformen, z.B. als Kapitalgesellschaft oder als Personengesellschaft[2], auftritt. Schließlich ist der vom Privatrecht unabhängige Bürger zu nennen, der bspw. Sozialhilfe beantragt oder seine Wohnung ummeldet. Zusammengefasst ergeben sich folgende Akteurgruppen:

1 Siehe Gesetzesvorbehalt und Gesetzesvorrang Anhang A, Seite 90.
2 Für eine vollständige Auflistung der Rechtsformen siehe www.teialehrbuch.de/DATV/15660-Rechtsformen-der-Unternehmung.html; Zugriff: 11.11.06

1. Government (G) = Öffentliche Verwaltungsorganisation[3]
2. Citizen (C) = Vom Privatrecht unabhängige Bürger
3. Business (B) = Unternehmung (Kapitalgesellschaft, Einzelunternehmen etc.)

Als Kommunikationskanal der Akteure liegen weiterhin E-Government Systeme vor, die als IT Systeme realisiert sind und der elektronischem Kommunikation folgen. Eine weitere Besonderheit ist, dass die öffentliche Verwaltung bei der Kommunikation immer beteiligt ist, wodurch sich die folgenden Kombinationen G2C, G2B und G2G[4] ergeben.

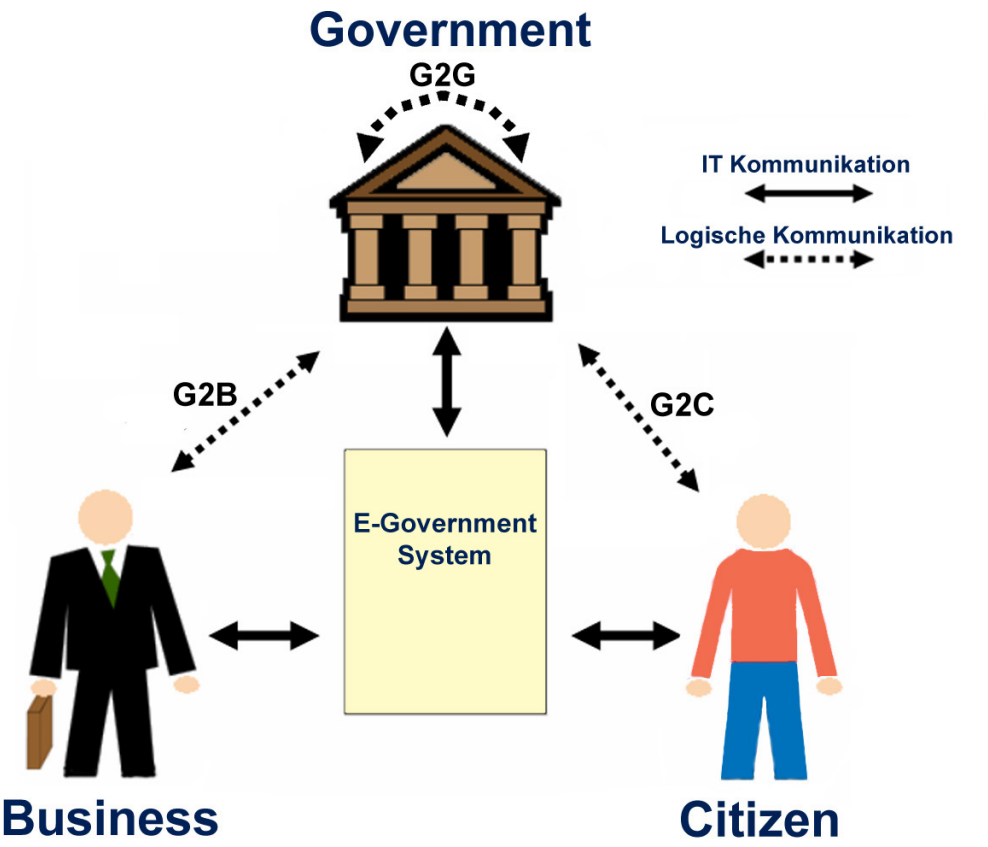


Abbildung 2: Systemsicht auf E-Government. Die Akteure kommunizieren nie direkt miteinander, sondern verwenden als Medium das E-Government System. Die fachlichen Inhalte werden zwar über dieses ausgetauscht, richten sich allerdings auf der logischen Ebene nach dem Verständnis zwischen den Akteuren.

## 2.2 Verwaltungsverfahren

Das Verwaltungsverfahren ist ein Teilgebiet des formalen Verwaltungshandelns (siehe Anhang A, S. 92), das nach außen zu den Akteuren Citizen und Business gerichtet und

3 Für eine detaillierte Beschreibung der Verwaltungsorganisation siehe Anhang B, Seite 93.
4 Großbuchstabe stellt das erste Zeichen des Akteurtyps. Die Zahl '2' stellt neudeutsch eine Kommunikation zwischen beiden Akteurtypen dar.

an der die Behörde beteiligt ist. Kodifiziert ist das Verwaltungsverfahren im Verwaltungsverfahrensgesetz (VwVfG).

### 2.2.1 Beteiligte

Es wurden bereits Akteure vorgestellt, die mit E-Government Systemen interagieren (siehe S.4). Im Kontext des Verwaltungsverfahrens werden sogenannte Beteiligte im VwVfG §11-13 ähnlich zu Akteuren konkretisiert. Danach sind die Subjekte, die an Verwaltungsverfahren beteiligungsfähig sind, natürliche, juristische Personen, Behörden und andere rechtliche legitimierte Vereinigungen. Als Rollen der Beteiligten werden Antragsteller, Antragsgegner, Adressaten von Verwaltungsakten und Vertragspartner eines öffentlich rechtlichen Vertrags angesehen. Die zur Durchführung eines Verfahrens betraute Behörde ist niemals Beteiligter sondern Träger des Verfahrens [Sod/Zie,S.451]. Die betroffene Behörde ist also kein Beteiligter im verwaltungsverfahrensrechtlichen Sinne[5].

### 2.2.2 Verwaltungsakt

Im Speziellen ist der Verwaltungsakt neben dem öffentlich rechtlichen Vertrag ein Verwaltungsverfahrenstyp, der unterschriftspflichtige Papierdokumente in den einzelnen Verfahrensschritten einsetzt. Der Verwaltungsakt ist im VwVfG §35,S.1 folgendermaßen definiert: „Verwaltungsakt ist jede Verfügung, Entscheidung oder andere hoheitliche Maßnahme, die eine Behörde zur Regelung eines Einzelfalles auf dem Gebiet des öffentlichen Rechts trifft und auf unmittelbare Rechtswirkung nach außen gerichtet ist“.

Darüberhinaus werden Verwaltungsakte weiter kategorisiert in u.a. belastende (z.B. Antragsablehnung Baugenehmigung), begünstigende (z.B. Gewährung Sozialhilfe), gestaltende und feststellende (z.B. Ummeldung Wohnsitz) Verwaltungsakte.

## 2.3 Formvorgaben

Innerhalb eines einzelnen Verwaltungsaktes existieren Vorgaben, in welcher Form bspw. Anträge oder Bescheide in den einzelnen Verfahrensschritten vorliegen müssen (z.B. schriftlich oder mündlich). Dazu lassen sich die Kriterien eines Verwaltungsaktes (siehe Definition oben) mit der Formvorgabe in Beziehung setzen[6]:

5 Im Unterschied dazu ist aus E-Government Systemsicht der Träger des durchzuführenden Verwaltungsverfahrens ebenfalls Akteur. Daraus folgt, dass Beteiligter im Sinne des VwVfG und Akteur keine deckungsgleichen Begriffe sind: alle Beteiligten sind Akteure, aber nicht umgekehrt.

6 Für die Bedeutung der Verwaltungsaktdefinition siehe [Sod/Zie,S.459].

Es handelt sich um eine hoheitliche Maßnahme, so dass eine einseitige Erklärung der übergeordneten und entscheidungsbefugten Verwaltung vorgenommen wird. Diese Erklärung muss in einer bestimmten Form bekannt gegeben werden. Außerdem wird die Aufgabe von einer Behörde übernommen, es wird also eine unmittelbare Außenwirkung erzielt. Danach muss festgelegt werden, in welcher Form die Kommunikationsinhalte zwischen Behörde und Bürger vorliegen müssen. Es findet Öffentliches Recht Anwendung, wodurch Formvorgaben für alle festgelegt und nicht verhandelbar sind.

Entsprechende Formvorgaben, die oben genannte Gesichtspunkte regeln, werden durch die sogenannte Formvorschrift festgelegt.

### 2.3.1 Formvorschrift

Nach VwVfG,§10 gilt im Verwaltungsverfahren die grundsätzliche Formfreiheit. D.h. so lange keine andere Rechtsvorschrift eine bestimmte Form ausdrücklich fordert, ist es den Akteuren im Verwaltungsverfahren gestattet, sich für eine Form zu entscheiden. Allerdings gilt die Vorgabe, die geeignete Form zu wählen, die einer einfachen, zweckmäßigen und zügigen Durchführung der entsprechenden Verwaltungsaufgabe dient.

Durch die Formfreiheit ist es also generell legitim, Verwaltungsaufgaben auf elektronischem Wege durchzuführen. Allerdings existieren viele Verwaltungsaufgaben[7], die durch Formvorschriften zugunsten der Schriftform eingeschränkt werden. Zudem ist es in der Verwaltungspraxis üblich, dass trotz Formfreiheit die Schriftform anderen Formen (mündlich, Handzeichen) vorgezogen wird, da hier Vorteile für die Akteure durch möglichen Nachweis offensichtlich sind [Be/Fü/Ge, S.75]. So lässt sich ein Antrag einer KfZ Ummeldung oder eine Entscheidungsbekanntgabe im Verwaltungsakt in der Schriftform leichter dokumentieren und beweisen, als ein Telefonat oder ein Handzeichen.

### 2.3.2 Schriftform

Die Schriftform schreibt vor, dass eine Urkunde mit einer eigenhändigen Namensunterschrift oder einem notariell beglaubigten Handzeichen von dem bzw. den Beteiligten zu unterzeichnen ist (BGB,§126)[8]. Die Begriffe Urkunde und eigenhändige Unterschrift sollen folgend näher betrachtet werden:

7 Oft sind diese im domänenspezifischen besonderen Verwaltungsrecht kodifiziert. Bspw. ist die Schriftform zwingend beim städtebaulichen Vertrag (BauGB §11,Abs.3).

8 Im Öffentlichen Recht wird die Schriftform allerdings insofern redefiniert, dass eine Unterschrift nicht immer vorliegen muss (VwVfG §37,Abs.5), bspw. bei automatisiert verfassten Dokumenten.

**Urkunde**

Die Urkunde ist eine verkörperte Gedankenerklärung, wobei mit Verkörperung ein festes körperliches Medium gemeint ist. Zudem muss der dargestellte Sachverhalt auf einer Urkunde optisch-visuell wahrnehmbar und verständlich sein [Lehre].

Beispiel einer Urkunde ist ein mit deutschem Schreibmaschinentext beschriebenes Papierstück: Es ist fest körperlich (Papier), optisch-visuell wahrnehmbar und verständlich (deutscher lesbarer Text). Als Beispiele, die der Urkundendefinition nicht folgen, könnten Tonbandaufnahmen (nicht optisch-visuell) oder Schrift im Sand (nicht fest körperlich) genannt werden.

**Eigenhändige Unterschrift**

Die eigenhändige Unterschrift der Schriftform hat folgende Funktion [Bertsch,S.15ff.]:

- Echtheitsfunktion: Die Unterschrift kann nur von einer Person stammen, dadurch ist eine eindeutige Zuordnung möglich.
- Abschlussfunktion: Der Inhalt der Urkunde wurde zur Kenntnis genommen und entspricht dem Willen des Unterschreibers. Das Entwurfsstadium des Schriftstücks wird abgeschlossen und das Schriftstück wird offiziell.
- Warnfunktion: Der Unterzeichner wird explizit zur Unterschriftsaktion aufgefordert, so dass sich dieser bereits vor dem Unterzeichnen über die Konsequenzen des Schriftstückinhalts bewusst ist.
- Identifikationsfunktion: Der Empfänger des Schriftstücks kann die Person über die Unterschrift identifizieren.
- Beweisfunktion: Bei Streitigkeiten und vor Gericht ist es nachvollziehbar, dass der Inhalt der Erklärung durch den Unterzeichner bestätigt wurde.

Die Abgrenzung zwischen Echtheits, Identifikations- und Beweisfunktion ist etwas schwierig, da sie sich alle auf die Zuordnung zwischen der Unterschrift und dem Unterzeichner beziehen. Die Unterschiede liegen hier eher in den unterschiedlichen Perspektiven (Unterschreiber, Empfänger, Gericht bei Streitfall).

### 2.3.3 Elektronische Form

Obwohl die Verkörperung bei der Urkunde nicht auf die Papierform beschränkt ist, liegt diese in der öffentlichen Verwaltung meistens als Papierschriftstück vor [Skrobotz]. Zudem kann man bei der elektronischen Form nicht wirklich von einer Verkörperung spre-

chen, da der Inhalt nicht nicht unmittelbar ohne technische Hilfsmittel (siehe Definition elektronisches Dokument S.4) interpretiert werden kann. Auch wenn die Daten auf einer CD, Festplatte oder im Hauptspeicher existieren, ist immer noch eine Applikation (bspw. Acrobat Reader, Word Prozessor) nötig, um die Daten für den Menschen interpretierbar zu machen. Auch ist unklar, inwieweit binäre Daten, falls sie denn als Schriftform akzeptiert würden, mit einer eigenhändiger Unterschrift zu signieren wären. Eine Warnfunktion über den nur schwer bis unmöglich interpretierbaren Binärdateninhalt ist z.B. für den Unterschreiber nicht nachvollziehbar.

Auf diese rechtlichen Unklarheiten, die Schriftform auf die elektronische Form [Kunstein,S.122] zu übertragen, reagierte der Gesetzgeber sowohl im Privat -als auch im Öffentlichen Recht[9]. Im dritten Verwaltungsreformgesetz im Jahre 2003 wurde das Verhältnis zwischen der Schriftform und elektronischen Dokumenten geregelt, worin die elektronische Form als Begriff neu eingeführt wurde und grundsätzlich gleichberechtigt neben der Schriftform existieren darf. Eine Ausnahme besteht, wenn explizit durch eine Rechtsvorschrift die Schriftform gefordert wird (VwVfG,§3a Abs.2). Wie bei der Schriftform existiert beim elektronischen Dokument ebenfalls die Forderung einer Unterschrift. Diese wird nicht wie beim Schriftstück eigenhändig, sondern mit einer sogenannten qualifizierten elektronischen Signatur umgesetzt.

Um ein hohes Schutzniveau und eine langzeitige Beweisbarkeit zu sichern, kann bei ausgewählten Verwaltungsverfahren auch die sogenannte akkreditierte elektronische Signatur gefordert werden (siehe S.11), die noch mehr Anforderungen als die qualifizierte erfüllen muss [Roßnagel, S.5].

## 2.4 Elektronische Signatur

### 2.4.1 Signaturgesetz

Über die qualifizierten elektronischen Signaturen hinaus, werden allgemein für alle elektronische Signaturen rechtliche Rahmenbedingungen im Signaturgesetz (SigG) geschaffen. Es werden nicht nur verschiedene Typen elektronischer Signaturen definiert, sondern es existieren auch Vorgaben, die bspw. Anforderungen an die Public Key Infrastruktur (siehe S.34) oder an Zeitstempel (siehe S.11) stellen. Das Signaturgesetz wird darüberhinaus von der Signaturverordnung (SigV) ergänzt, um Umsetzungen der

9 Eine Betrachtung der Gesetzeslage des Öffentlichen Rechts ist hier ausreichend, da sie für das Verwaltungsverfahren relevant ist.

allgemeinen Vorgaben des Signaturgesetzes zu konkretisieren.

Der Begriff Signatur taucht im ähnlichen Zusammenhang bei der Digitalen Signatur auf. Allerdings sind die Begriffe elektronische- und Digitale Signatur nicht zu verwechseln. Die Digitale Signatur ist eine Sicherheitsmechanismus, das kryptografische Technologien verwendet (siehe S.36). Dagegen definiert SigG §2Abs.1, dass elektronische Signaturen Daten in elektronischer Form darstellen, elektronischen Daten beigefügt bzw. mit ihnen verknüpft werden und der Authentikation dienen. Nach dieser Definition ist die elektronische Signatur viel allgemeiner formuliert und technologisch neutral anzusehen. Die Digitale Signatur ist demnach lediglich eine mögliche Umsetzung einer elektronischen Signatur [Kunstein, S.47].

### 2.4.2 Signaturtypen

Im Signaturgesetz werden die Begriffe Signaturschlüssel und Signaturprüfschüssel eingeführt. Mit dem Signaturschlüssel wird die Signatur erzeugt, der Signaturprüfschlüssel ermöglicht die Prüfung der Signatur[10]. Es werden weiterhin folgende elektronische Signaturtypen [BSIPrinc,S.28] eingeführt:

**Einfache elektronische Signatur**

Sind Daten in elektronischer Form, die der Authentisierung dienen. Die Daten können dabei beliebig sein (z.B. eine eingescannte Unterschrift).

**Fortgeschrittene elektronische Signatur**

Die Signatur muss dem Inhaber eindeutig zugeordnet und eine Identifikation muss ebenfalls möglich sein. Die zu signierenden Daten müssen mit der Signatur zudem so verknüpft werden, dass sie nicht korrumpiert werden können. Der Signaturschlüssel kann vom Inhaber mit eigenen Mitteln erzeugt werden. Damit die Signatur für Außenstehende nachvollziehbar ist, muss der Signaturprüfschlüssel diesen zugänglich gemacht werden.

**Qualifizierte elektronische Signatur**

Der Signaturschlüssel wird nicht wie bei der fortgeschrittenen Signatur direkt vom Inhaber, sondern von einer externen sicheren Stelle als Zertifizierungsanbietern (Trust Cen-

10 In der Kryptgrafie wird dazu der Signaturschlüssel als geheimer und der Signaturprüfschlüssel als öffentlicher Schlüssel bezeichnet (siehe S.36).

ter[11]) erzeugt. Weiterhin ist der Signaturschlüssel und der Signaturprüfschlüssel einem gültigem qualifizierten Zertifikat zugeordnet, so dass für Außenstehende die Authentizität des Signaturprüfschlüssels nachvollziehbar ist.

**Akkreditierte elektronische Signatur**

Es werden die gleichen Forderungen wie an die qualifizierte elektronische Signatur gestellt. Weiterhin muss die Erzeugungsstelle des Zertifikats ein akkreditierter Zertifizierungsanbieter sein, der von staatlicher Seite zertifiziert worden ist und eine 30-jährige Überprüfbarkeit der Signatur garantiert.

### 2.4.3 Einschränkungen der Zertifikatausstellung

Die Ausstellung eines qualifizierten Zertifikats durch ein Trust Center (siehe oben qualifizierte Signatur) ist nur für natürliche Personen möglich [Bertsch,S.34], so dass technische Systeme oder juristische Personen keine vom im E-Government geforderten qualifizierten Signaturen erzeugen können.

So ist es nicht legitim, dass für alle Mitarbeiter einer Behörde als juristische Person ein „behördenweites“ Zertifikat existiert, mit dessen zugeordneten Signaturschlüssel ein Sachbearbeiter ein Dokument signiert. Auch die Übertragung eines einer natürlichen Person zugeordneten Signaturschlüssels an Dritte widerspricht der Festlegung, dass die Signatur nur direkt vom Inhaber auszuführen ist [Kunstein,S.187]. Diese Einschränkungen gelten ebenfalls für technische Systeme, so dass eine vom Server erzeugte Signatur nicht qualifiziert sein kann.

### 2.4.4 Zeitstempel

Zeitstempel verknüpfen elektronische Daten mit einem Zeitpunkt. Um die Zeitpunktangaben später vor einer Fälschung zu schützen, können die miteinander verknüpften Daten elektronisch signiert werden. Im E-Government ist die Relevanz der Verwendung von Zeitstempeln bspw. in Fragen von Fristeinhaltungen naheliegend. Zeitstempel werden in zwei Kategorien eingeteilt[12]:

**Qualifizierter Zeitstempel**

Dieser Zeitstempel ist im E-Government rechtskräftig (SigG §4, Abs.14) und wird von einem akkreditierten Zertifizierungsdiensteanbieter ausgestellt. Der Zeitstempel ge-

11 Ähnlich zur Trusted Third Party, siehe S.33
12 Siehe www.datenschutz-bayern.de/technik/orient/virtpost.html#ab8.3; Zugriff: 28.11.06

währleistet hier wie die akkreditierte elektronische Signatur eine Überprüfbarkeit von mindestens 30 Jahren.

**Sonstige Zeitstempel**

Darunter fallen alle sonstigen Zeitstempel, die nicht den Forderungen eines qualifizierten Zeitstempels entsprechen und nicht von einem akkreditierten Zertifizierungsdienstanbieter stammen. Als Beispiel ist ein Zeitstempel zu nennen, der über die Systemzeit des lokalen Rechners ermittelt wird. Diese Systemzeit ist unabhängig von externen Stellen und lässt sich in gängigen Betriebssystemen leicht über die Systemeinstellungen manipulieren.

## 2.5 Datenschutz

Grundsätzlich hat jeder Bürger das Recht zu bestimmen, welche von seinen personengebundenen[13] Daten weitergegeben und verwendet werden dürfen. Dabei darf dieses Recht nur von weiteren Gesetzen eingeschränkt werden und muss im Sinne des Allgemeinwohls stehen[14]. Dieses informationelle Recht wird durch den Datenschutz gesichert. Rechtstexte des Datenschutzes sind u.a. das Bundesdatenschutzgesetz (BDSG), die Datenschutzgesetze der Länder und die domänenspezifischen Gesetze (bspw. Sozialgesetzbuch), die festlegen, welche persönlichen Daten erhoben und verarbeitet werden dürfen[15].

Es ist offensichtlich, dass bei E-Government Geschäftsvorfällen größtenteils personengebundene Daten bspw. über einen elektronisch versendeten Antrag übertragen werden. Die Behörde in Form des Behördenmitarbeiters darf hier wie bei der klassischen Papierform nicht beliebige, sondern nur für den Antrang zweckdienliche Daten anfordern und diese auch nicht beliebig weiterverarbeiten. Zudem muss zugesichert werden, dass nur Befugte Zugriff auf die Daten erhalten.

Da im E-Government als Medium das Internet intensiv genutzt wird, fallen über die Inhalte der übertragenen Dokumente hinaus weitere Daten an. So werden über die verschiedenen Kommunikationsschichten[16] Informationen übertragen, die die elektronische Kommunikation realisieren. Die rechtlichen Rahmenbedingungen, die im

13 Personengebundene oder persönliche Daten sind Daten, die einer bestimmten Person zugeordnet werden können.
14 Siehe www.oefre.unibe.ch/law/dfr/bv065001.html; Zugriff:02.12.06
15 Der Begriff der Verarbeitung ist weit ausgelegt (Speicherung, Weitergabe, Verändern, Löschen etc.), siehe BDSG §3.
16 Hier sind Schichten im Sinne des OSI Schichtenmodells gemeint.

Zusammenhang mit der Erhebung und Verarbeitung solcher inhaltsunabhängigen, aber dennoch personengebundenen[17] Daten, stehen, sind im TKG, TDSV, TDG, TDDSG und im MDStV verfasst [DsgEgov,S.21].

Die Gesetzestexte des Datenschutzes sind technikneutral, so lässt sich in den Texten keine Beschränkung auf eine bestimmte Technologie finden, sondern es werden lediglich Forderungen gestellt, die eine technologische Vorrichtung erfüllen muss. Sicherheitsmechanismen, wie bspw. die Kryptografie, die eine Geheimhaltung von Daten gewährleisten können und auf diese Weise dem Datenschutz dienlich sind, werden im folgenden Kapitel besprochen.

17 So lässt sich beim Surfen mit statischer IP Adresse der genutzte Rechner, und somit eine Person oder zumindest eine kleine Benutzergruppe identifizieren. Auch kann bei OSCI (siehe S.40) durch Diagnose der Nachricht auf eine Person geschlossen werden, da entsprechende Nutzungsdaten u.a. zuordnungsbare Zertifikate vorliegen.

# 3 Sicherheitsmechanismen

## 3.1 Einleitung

E-Government bietet staatliche Dienstleistungen über elektronische Kommunikationskanäle an. Da diese, wie bspw. das Internet für viele potentiell zugänglich sind, hat dies direkte Folgen für die Sicherheit in der Kommunikation mit elektronischen Dokumenten. So besteht ohne weitere Sicherheitsmaßnahmen die Gefahr, folgende Schutzziele eines Dokumentenaustausches während einer E-Government Geschäftsvorfallabwicklung zwischen zwei Personen zu verletzen [Tanenbaum, S.722]:

1. Geheimhaltung: Ein vertrauliches oder geheimes Dokument kann von Unbefugten gelesen werden.
2. Integrität: Das Dokument kann von einer dritten Person gefälscht werden.
3. Authentizität: Der eigentliche Absender des Dokumentes kann sich für eine andere Person ausgeben.
4. Nichtabstreitbarkeit: Der Absender kann behaupten, das Dokument gar nicht abgesendet zu haben.

Um diese Schutzziele im E-Government sicherzustellen werden entsprechende Forderungen von der elektrischen Form (siehe S.8) und dem Datenschutz (siehe S.12) gestellt. Um diese rechtlichen Vorgaben technisch umzusetzen, existieren verschiedene Sicherheitsmechanismen, die in diesem Kapitel dargestellt werden.

## 3.2 Kryptografie

Die Kryptografie ist neben der Kryptoanalyse eine der beiden Teilwissenschaften der Kryptologie [Anderson, S. 74]. Die Kryptografie hat das Ziel, Systeme zur Verfügung zu stellen, die eine für jeden erfassbare[1] Information auf die Weise transformiert, dass nur erwünschte bzw. befugte Instanzen[2] die ursprüngliche Information entschlüsseln und verstehen können oder eine Retransformation gar nicht mehr möglich ist. Dazu werden sogenannte Kryptosysteme entworfen, die die Schutzziele Geheimhaltung, Integrität,

1 Die Abgrenzung von „erfassbar“ ist etwas schwierig, da jede Information in einer bestimmten Präsentation übermittelt wird und interpretiert werden muss. Allerdings sind viele Präsentationstypen wie z.B. die deutsche Sprache oder die lateinische Schrift gesellschaftlich und kulturell so verbreitet und bekannt, dass man hier von einer offensichtlich erfassbaren Information reden kann.

2 Instanz soll hier als Begriff für Adressaten stehen, bspw. in Form von Personen oder Computern.

Authentizität und Nichtabstreitbarkeit (siehe oben) gewährleisten. Die Kryptoanalyse testet dagegen die Sicherheit der einzelnen Kryptosysteme und versucht Schwachstellen aufzuzeigen.

Folgende Kryptosystemklassen existieren [Opplinger,S.7]:

1. Schlüsselloses Kryptosystem: Bspw. Einweg Hashfunktion
2. Schlüsselbehaftetes Kryptosystem: Symmetrisch und asymmetrisch

### 3.2.1 Begriffe

Folgende Definitionen sind für das Verständnis der Kryptosysteme nötig:

**Klartext:** Der Klartext ist eine unverschlüsselte und zu schützende Information, d.h. die Anzahl der Personen oder Applikationen, die diesen interpretieren bzw. einsehen können ist allein dadurch beschränkt, ob sie Zugriff auf den Klartext haben und sie den Repräsentationstyp des Klartextes kennen oder nicht. Die Repräsentationstypen der Klartexte sind dabei vielfältig, es kann sich bspw. um beliebige binäre Daten (z.B. PDF-Dokument oder Webformular) oder auch um eine analoge Tonbandaufnahme mit deutscher Sprache handeln.

**Ciphertext:** Der Ciphertext ist die transformierte Information, die vom Klartext mit Hilfe eines Verschlüsselungsalgorithmus abgebildet wurde. Der Ciphertext kann mit Hilfe von Schlüsseln mit einem Entschlüsselungsalgorithmus wieder zu einem Klartext retransformiert werden.

**Algorithmus:** Der Algorithmus stellt die Schritte einer Transformation einer Definitionsmenge auf eine Zielmenge dar.

**Schlüssel:** Mit dem Schlüssel wird der Klartext zu einem Ciphertext verschlüsselt und ein Ciphertext in einen Klartext entschlüsselt.

**Hashwert:** Der Hashwert ist das komprimierte Abbildungsergebnis aus Hashalgorithmus und dem Klartext als Funktionseingang.

**Transportkanal:** Der Transportkanal verbindet Kommunikationsteilnehmer, die Klartext-Informationen miteinander austauschen wollen. Für die sichere Kommunikation werden über diesen abhängig vom Kryptosystem zusätzlich Ciphertext, Schlüssel und Hashwert ausgetauscht. Der Transportkanal ist grundsätzlich als unsicher einzustufen[3].

3 Diese Annahme wird getroffen, da keine einheitlichen Sicherheitsniveaus für die vielen Transportkanaltypen (z.B. Internet, Intranet, WLAN) verwendet werden und das Kryptosystem sich zudem als loose gekoppelte Systemeinheit sieht, die allein für Sicherheit sorgt.

**Begriffsvereinfachung:**

Da die Arbeit sich auf eine sichere Kommunikation mit elekronischen Dokumente bezieht, ist bei den Begriffen Klartext, Ciphertext, Hashwert und Schlüssel immer von einer Binärrepräsentation die Rede. Der Algorithmus ist dabei als Computerprogramm in Form von Hardware oder Software implementiert und verwendet für die Transformationen abhängig vom Kryptosystem Klartexte, Ciphertexte und Schlüssel. Beim Transportkanal handelt es sich ebenfalls um ein elektronisches Medium (z.B. Internet, Intranet, IDE-Kanal Festplatte).

### 3.2.2 Protokoll

Für jedes kryptografische System existieren einzelne Schritte, in denen die Akteure mit dem Kryptosystem interagieren. Diese Schritte in zeitlicher Reihenfolge werden als Protokoll definiert [Gehring]:

> **Protokoll:** Das Protokoll ist die Verhaltensweise und Interaktion zwischen Akteuren und Kryptosystemen in zeitlicher Reihenfolge. Die Akteure sind dabei Personen oder technische Systeme (z.B. Verschlüsselungsprogramm).

### 3.2.3 Sicherheit von Kryptosystemen

Die Sicherheit von Kryptosystemen muss immer darauf bezogen werden, wie die Ziele der einzelnen Kryptosysteme definiert sind. So existieren für schlüssellose andere Angriffsziele als für schlüsselbehaftete. Dementsprechend werden Kryptoanalytiker angemessene Schwachstellenanalysen für das jeweilige Kryptosystem durchführen. Um diese nachvollziehen zu können, werden die Angriffspunkte der verschiedenen Kryptosysteme in den einzelnen Kapiteln vorgestellt.

Grundsätzlich ist die Frage, ob man die Implementierungsdetails und die Arbeitsweise eines Kryptosystems veröffentlichen sollte oder nicht. Es ist sogar möglich die Sicherheit eines Kryptosystems mit der Geheimhaltung des Algorithmus zu definieren[4].

In der heutigen Kryptologie ist man sich allerdings einig, dass Sicherheit eines Kryptosystems nicht über die Geheimhaltung des Algorithmus definiert werden sollte. So kann der implementierte und geheime Algorithmus als binäres Programm im Internet schnell verbreitet und disassembliert[5] werden, wodurch eine Offenlegung des Algorithmus

4 In der Informationstechnologie wird hier oft der engl. Begriff 'security through obscurity' verwendet

5 Ein binäres Programm kann durch Disassemblierung in den menschenlesbaren Source Code Zustand zurücktransformiert und das entsprechende Programm so analysiert werden.

möglich ist[6]. Zudem existiert bei Offenlegung des Algorithmus der Vorteil, dass eine breite Masse von Personen den Algorithmus einer Schwachstellenanalyse unterziehen kann kann [Schneier1, S.7]. Dass ein Kryptosystem nicht auf die Geheimhaltung des Algorithmus, sondern nur von der Geheimhaltung des Schlüssels abhängen[7] soll, ist allgemein als Kerkhoff-Prinzip bekannt [Kerckhoff]. Im Zusammenhang dazu bezieht sich der Begriff der Attacke [Schneier1,S.5]:

> **Attacke:** Ein kryptoanalytischer Versuch, Klartexte, Schlüssel zu rekonstruieren und bei Hashwerten Kollisionen zu finden. Der Algorithmus des Kryptosystems ist dabei bekannt.

# 3.3 Einweg Hashfunktionen

Der Sicherheitsaspekt Integrität (siehe S.14) bei einem Dokumentenaustausch zwischen zwei Personen kann auf folgende Weise verletzt werden:

1. Der Absender verfasst ein Dokument und sendet es über einen unsicheren Transportkanal an den Empfänger.
2. Ein Unbefugter, der Zugriff auf den Transportkanal besitzt, fängt das Dokument ab, ändert dieses und leitet es daraufhin an den Empfänger weiter.
3. Der Empfänger erhält ein ohne sein Wissen gefälschtes Dokument.

Es ist möglich eine solche Fälschung mit der Einweg Hashfunktion nachzuweisen.

## 3.3.1 Mathematische Grundlagen

Einweg Hasfunktionen gehören zur Klasse der schlüssellosen Kryptosysteme und verbinden die Eigenschaften der Einweg- und der Hashfunktionen [Me,Oo,Va].

**Einwegfunktion**

Die Einwegfunktion ist eine Abbildung $f : D \rightarrow W$ , deren Umkehrung $f^{-1}$ praktisch nicht möglich ist, also ein unrealistischer Kosten- und Zeitaufwand nötig ist, um die Umkehrung durchzuführen.

Es existieren auch Einwegfunktionen mit Falltür, d.h., dass mit einer bestimmten Information die Umkehrung wieder leicht möglich ist[8].

6 So geschehen 1994, wo ungewollt der Source Code des eigentlich geheimzuhaltenen RC4 Algorithmus über ftp-Server und Newsgroups verbreitet wurde.

7 Diese Aussage kann natürlich nicht auf schlüssellose Kryptosysteme bezogen werden. Allerdings sollte bei diesen der Algorithmus ebenfalls offengelegt sein.

8 Z.B. basiert der RSA Algorithmus (siehe S.30) auf einer Einwegfunktion mit Falltür.

**Hashfunktion**

Die Hashfunktion hat folgende Eigenschaften: Für einen Klartext Definitionsraum $Kl$ und dem Hashwerteraum $H$ existiert die Hashfunktion $hf : Kl \rightarrow H$ , bei der jedem Klartext genau ein Hashwert zugeordnet wird. Ferner gilt, dass $|Kl| > |H|$ , daraus folgt dass mehrere Klartexte $Kl_1 \subseteq Kl_{gesamt}$ auf den gleichen Hashwert $h \in H$ abgebildet werden, sich dadurch Kollisionen ergeben können und eine Komprimierung stattfindet. Zudem ist die Länge des Hashwertes konstant, so dass gilt $\forall h \in H : |h| = n, n \in \mathbb{N}$ . Die Hashfunktion ist als Abbildung demnach rechtseindeutig und nicht-linkseindeutig.

**Einweg-Hashfunktion**

Kombiniert aus Einweg- und Hashfunktion ergibt sich folgendes Konstrukt:

> **Einweg Hashfunktion:** Die Abbildung eines Klartextes in einen komprimierten Hashwert mit fester Länge. Die Umkehrung dieser Abbildung ist praktisch nicht möglich.

Für Einweg-Hashfunktionen ergeben sich folgende Punkte [Schneier1, S.429]:

1. Mit dem Klartext $kl \in Kl$ ist es einfach, den Hashwert $h \in H$ zu berechnen.
2. Mit einem gegebenen Hashwert $h \in H$ ist es praktisch unmöglich, einen Klartext $kl \in Kl$ zu berechnen, so dass $hf(kl) = h$ .
3. Mit dem Klartext $kl_1 \in Kl$ ist es praktisch unmöglich, einen weiteren Klartext $kl_2 \in Kl$ zu finden, so dass $hf(kl_1) = hf(kl_2)$ . Diese Eigenschaft wird auch als Kollisionsfreiheit bezeichnet.

### 3.3.2 Protokoll

Folgende Anwendung der Einweg Hashfunktion kann eine Fälschung aufdecken:

1. Absender und Empfänger einigen sich über die zu verwendene Einweg Hashfunktion.
2. Der Absender berechnet mit Hilfe einer Einweg Hashfunktion den Hashwert des Dokumentes $hf(Dokument) = hash$ . Danach sendet er das Dokument ab.
3. Der Empfänger erhält das Dokument. Als erstes errechnet er den Hashwert des empfangenen Dokumentes mit der Einweg Hashfunktion. Als nächstes erhält er den vom Absender errechneten Hashwert über eine abhörsichere Verbindung. Wenn der Hashwert mit seinem errechneten übereinstimmt, ist die die Integrität

des übertragenen Dokumentes nachgewiesen. Andernfalls ist ersichtlich, dass das Dokument entweder durch Übertragungsfehler korrumpiert oder von einer unbefugten Person gefälscht wurde.

Zentral am Protokoll ist, dass die Hashwertübertragung von Sender zum Empfänger über einen sicheren Transportkanal stattfinden muss, andernfalls ist es dem Unbefugten möglich, diese Hashwertinformation ebenfalls abzufangen und zu fälschen.

### 3.3.3 Attacken

Attacken auf Einweg Hashfunktionen haben das Ziel, Kollisionen zu finden und sind auf zwei Weisen möglich:

1. Attacke auf die Hashwertbitlänge
2. Attacke auf den Hashalgorithmus

Für das Verständnis ist wichtig, die Wahrscheinlichkeiten von Kollisionen zu verstehen. Wenn wir eine gute Einweg-Hashfunktion haben folgt daraus, dass die Verteilung der Hashwerte konstant ist, d.h. es ist immer gleich wahrscheinlich, dass wir mit einem gegebenen Klartext einen bestimmten Hashwert erhalten. Die Wahrscheinlichkeit, eine Kollision zu erhalten wäre dann allein von der Bitlänge des Hashwertes abhängig.

Als Beispiel nehmen wir an, dass die Bitlänge des Hashwertes 8-bit beträgt: Ein Bit kann die zwei Zustände $0$ und $1$ einnehmen, insofern ist die Basis des Exponenten $2$ . Insgesamt ergibt sich durch die Bitlänge von 8 Ziffern folgende Anzahl von möglichen Kombinationen: $2^8=256$ . Statistisch gesehen wird also jeder 256-te Klartext auf den gleichen Hashwert abgebildet und es tritt eine Kollision auf.

**Attacke auf Hashwertbitlänge**

Bei den Einweg Hashfunktionen werden Attacken auf die Hashbitlänge Brute Force Attacken genannt. Dabei wird angenommen, dass der Hashalgorithmus optimal gewählt wurde und so die Wahrscheinlichkeit der Abbildung eines Klartextes auf einen bestimmten Hashwert gleich hoch ist. Es wird in zwei Brute Force Attack Typen unterschieden [Heise sec]:

**1)** Pre-Image Attacke: Zu einem gegebenen Hashwert $h \in H$ wird so lange eine Hashfunktion $hf$ auf beliebige Klartexte $kl' \in Kl, kl' \neq kl$ angewandt bis man eine Kollision gefunden hat, also $hf(kl')=hf(kl)$ eintritt. Die Wahrscheinlichkeit besagt, dass man bei einer Hashwertbitlänge von 8 Bit 256 verschiedene Klartexte ausprobieren muss,

um garantiert eine Kollision zu finden.

**2)** Kollisionsattacke: Man sucht zwei beliebige Klartexte $kl \in Kl$ und $kl' \in Kl$ , so dass $hf(kl') = hf(kl)$ . Die Wahrscheinlichkeit dies zu erreichen ist viel höher als bei der ersten Brute Force Attacke, und zwar sinken bei einer Hashwertbitlänge von 8 Bit die Möglichkeiten von 256 auf $2^{8/2} = 16$ . Dieses Phänomen, dass die Anzahl der Möglichkeiten deutlich sinkt (siehe den Teiler im Exponenten), wird durch das Geburtstagsparadoxon veranschaulicht[9] und der entsprechende Angriff aus diesem Grund auch als Geburtstags-Attacke bezeichnet [Schneier1, S.166].

**Attacke auf Hashalgorithmus**

Der Angriff auf den Algorithmus beruht darauf, dass angenommen wird, dass die Einweg Hashfunktion insofern schwach ist, dass die Wahrscheinlichkeit für das Auffinden einer Kollision höher ist als die Anwendung einer Brute Force Kollisions- oder Pre-Image Attacke [Schneier2].

So ist es bspw. einem chinesischen Forscherteam nach Analyse des SHA-1 Hashalgorithmus 2005 gelungen, den Aufwand für Kollisionen zunächst von $2^{80}$ auf $2^{69}$ Schritte zu reduzieren [Wa,Yi,Yu]. Kurze Zeit später gab es vom gleichen Forscherteam sogar noch weitere Erfolge, wo der Aufwand auf $2^{63}$ weiter minimiert wurde [Schneier3].

**Beispielattacken**

Pre-Image Attacke:

1. Der Absender ermittelt den Hashwert des Dokumentes, sendet das Dokument über einen unsicheren und den Hashwert über einen sicheren Transportkanal an den Empfänger.
2. Ein Unbefugter fängt das Dokument ab. Da die verwendete Hashwertlänge sehr kurz ist, fällt es sehr leicht, ein manipuliertes bzw. gefälschtes Dokument aufzusetzen, das den gleichen Hashwert ergibt. Nach Ermittlung des gefälschten Dokumentes leitet er dieses weiter.
3. Der Empfänger empfängt ein ohne sein Wissen gefälschtes Dokument. Er hegt keinen Verdacht, da der ermittelte Hashwert mit dem des Absenders übereinstimmt.

9 Für eine mathematische Herleitung siehe mathworld.wolfram.com/BirthdayProblem.html; Zugriff: 25.01.06

Kollisionsattacke[10]:

1. Es kommt zu einem Kaufvertrag zwischen zwei Personen. Um auf beiden Seiten Sicherheit zu schaffen und vor Gericht eine Fälschung beweisen zu können, einigt man sich, dass der Vertrag gehasht wird,
2. Jetzt erzeugt der Verkäufer über einen Kollisionsangriff zwei Dokumente, die zwar den gleichen Hashwert ergeben, allerdings unterschiedliche Inhalte besitzen. Das zweite Dokument besitzt dabei einen höheren Rechnungsbetrag als das erste.
3. Der Verkäufer bucht einen zu hohen Betrag ab, der Käufer klagt und verliert vor Gericht, weil der Hashwert des Vertrages mit dem zu hoch abgebuchten Geld identisch mit dem vorgelegten des Käufers ist und die Integrität nachgewiesen wurde.

**Folgerung**

Ziel bei einer sicheren Einweg Hashfunktionen ist also Pre-Image und Kollisionsangriffe praktisch über die Wahl einer hohen Hashwertbitlänge unmöglich zu machen[11]. Auf Algorithmusseite erreicht man dies üblicherweise über einen guten Algorithmus, dessen Brechen nur geringfügig oder idealerweise gar nicht von einer Brute Force Methode abweicht.

### 3.3.4 Beispiel SHA-1

Als Variante des SHA Algorithmus ist SHA-1 eine Einweg Hashalgorithmus, der einen bis zu $2^{64}$ Bit langen Klartext entgegennimmt und auf einen 160 Bit langen Hashwert abbildet. SHA selber wurde von NIST und der NSA in entwickelt, um eine sichere Hashfunktion für den Digital Signatures Algorithm (DSA) zur Verfügung zu haben. DSA ist wiederum selber Teil des Digital Signatures Standard (DSS) [Schneier1, S.442].

Da Attacken auf den SHA-1 Algorithmus 2005 erfolgreich waren (siehe oben), wird derzeit empfohlen, für Migration vorhandener Anwendung und Neuentwicklungen SHA-256/348/512 einzusetzen, die alle längere Haswertbitlängen verwenden [GIKrypt].

10 Das Beispiel soll die Kollisionsattacke einfach erläutern und ist aus rechtlicher Sicht stark vereinfacht.

11 So überlegt man derzeit die Bitlänge des SHA-1 Hashwertes von 160 auf 224,256,384,512 respektive Bits zu erhöhen [Heise sec].

### 3.3.5 Anwendungsgebiete für Einweg Hashfunktionen

**Downloads**

Hashwerte stellen sicher, dass bei Softwarepaketdownloads die entsprechende Datei nicht fehlerhaft übertragen wurde. Der Zugriff auf den Hashwert wird meistens dadurch erleichtert, dass er offen zugänglich auf der Downloadwebsite sichtbar ist und ebenfalls runtergeladen werden kann. Dadurch ist ein Vergleich der beiden Hashwerte leicht durchführbar.

**Authentisierungssysteme**

Um nur berechtigten Personen Zugriff auf bestimmte Bereiche zu gewähren, wird oft ein Login mit Benutzernamen und Password realisiert. Betriebssysteme wie Windows oder Linux speichern dabei die Passwörter nicht im Klartext sondern errechnen den Hashwert und vergleichen bei jedem Login das gehashte eingegebene Passwort mit dem gespeicherten Hashwert.

**Digitale Signaturen**

Hashfunktionen werden im Zusammenhang mit schlüsselbehafteten Kryptosystemen verwendet, um Digitale Signaturen zu realisieren (siehe S.36). Dabei wird vom zu signierenden Klartext ein Hashwert gebildet, der die Fälschung eines Dokumentes praktisch unmöglich macht.

## 3.4 Symmetrische Kryptosysteme

Über einen Transportkanal ist folgende Verletzung des Sicherheitsziels Geheimhaltung (siehe S.14) möglich:

1. Der Absender, eine absolute Vertrauensperson des Empfängers, sendet ein ei-elektronisches Dokument, in dem streng geheime Informationen gespeichert sind.
2. Ein Unbefugter kann die Kommunikation zwischen den Dokumentaustauschpartnern nachvollziehen und fängt das Dokument ab und erlangt Zugriff auf die geheimen Informationen. Damit kein Verdacht aufkommt, dass die geheimen Informationen gelesen wurden, leitet er das Word Dokument schließlich an den Empfänger weiter.

3. Der Empfänger erhält das Dokument, dessen geheime Inhalte von einem Unbefugtem gelesen wurden.

Die Sicherstellung der Geheimhaltung, so dass nur Befugte einen Klartext lesen können, ist über schlüsselbehaftete Kryptosysteme möglich. Symmetrische Kryptosysteme gehören dabei zu den Lösungen, bei denen ein Klartext mit Hilfe eines Algorithmus und eines einzigen Schlüssels verschlüsselt und entschlüsselt werden kann. Im Gegensatz zu den Hashfunktionen ist eine Umkehrung dieser Transformation möglich und auch erwünscht. Dafür wird für die Entschlüsselung der gleiche Schlüssel verwendet wie für die Verschlüsselung.

### 3.4.1 Mathematische Grundlagen

Es existiert ein Klartext aus einer beliebigen Klartextmenge $kl \in Kl$ . Dieser wird mit einem geheimen Schlüssel aus einer Schlüsselmenge $s_{geheim} \in S$ verschlüsselt. Die Bitlänge des Schlüssels ist dabei festgelegt und abhängig vom konkret verwendeten Kryptosystem. Die Verschlüsselungsfunktion $v$ nimmt als Eingabeparameter einen geheimen Schlüssel und den Klartext, worauf der unlesbare Ciphertext $c \in C$ entsteht: $v(kl, s_{geheim}) = c$ . Die Entschlüsselungsfunktion $e$ kehrt die Richtung um: $e(c, s_{geheim}) = kl$ . Es gilt also: $e(v(kl, s_{geheim}), s_{geheim}) = kl$ . Außerdem findet keine Kompression statt, so dass die Länge der Klar- und Ciphertexte gleich groß ist: $|kl| = |c|$ .

Die Ver- und Entschlüsselungsfunktionen wenden dabei verschiedene Operationen wie Substitution, Logisches Oder, Modulus, Transposition, Permutation etc.[12] auf die Klar- bzw. Ciphertexte an. Die tatsächlich verwendeten Operationen sind dann abhängig vom konkreten Ver- und Entschlüsselungsalgorithmus.

Ferner existieren mit den Strom- und Blockalgorithmen grob unterteilt zwei verschiedene symmetrische Algorithmustypen. Stromalgorithmen konsumieren Klartext als Strom von Bits und vollziehen mit Hilfe des aus dem Schlüssel abgeleiteten Schlüsselstroms einen Transformationsschritt pro Bit. Bei der Entschlüsselung wird der gleiche Vorgang vollzogen, mit dem Unterschied, dass diesmal der Ciphertext als Eingangsstrom verwendet wird. Dagegen teilt ein Blockalgorithmus den Klartext bzw. Ciphertext in gleich große Blöcke ein, um diese dann mit Hilfe des Schlüssels zu ver -bzw entschlüsseln.

12 Aus Platzgründen wird nicht näher auf diese Operationen eingegangen. Ausführliche Literatur dazu ist zu finden unter [Schneier1],[Me,Oo,Va] und [Opplinger].

### 3.4.2 Protokoll

Mit folgendem Protokoll lässt sich im Zusammenhang eines Dokumentenaustauschs eine Geheimhaltung sicherstellen:

1. Beide Austauschpartner einigen sich auf ein symmetrisches Kryptosystem und kennen den gleichen geheimen Schlüssel.
2. Der Absender verschlüsselt das Dokument mit dem geheimen Schlüssel und sendet es über einen unsicheren Transportkanal.
3. Der Adressat nimmt das Dokument entgegen und kann es mit dem geheimen Schlüssel entschlüsseln, so dass er das Dokument als Klartext erhält.

Bedingung, damit die Geheimhaltung sichergestellt ist natürlich, dass die Austauschpartner ein Kryptosystem verwenden, welches als sicher eingestuft wird. Andernfalls ist es möglich, dass ein Angreifer das Dokument abfängt, die Schwächen ausnutzt und so das Dokument lesen kann. Zudem ist es essentiell, dass der geheime Schlüssel nur dem Sender und Empfänger bekannt ist und nicht erst über einen Transportkanal ausgetauscht werden muss.

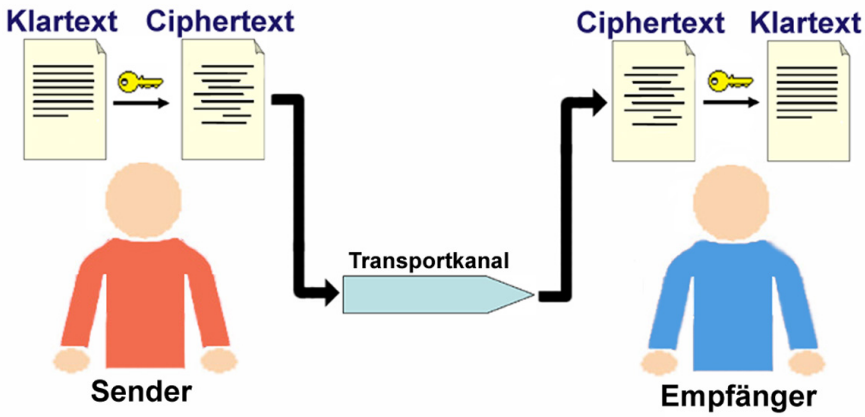


Abbildung 3: Protokoll einer symmetrischen Verschlüsselung. Der Schlüssel ist nur dem Sender und Empfänger bekannt.

### 3.4.3 Attacken

Brute Force Attacken existieren ebenfalls bei schlüsselbehafteten Kryptosystemen. Dabei werden unterschiedliche Schlüssel so lange ausprobiert, bis eine Entschlüsselung erfolgreich war.

Die Wahrscheinlichkeit wird hier über die Bitlänge des Schlüssel festgelegt: Die Länge des Schlüssels ist der Exponent, $2$ die Basis (für die Herleitung siehe S.19) . Somit ergeben sich bei einer Schlüssellänge von 64 Bit $2^{64} \approx 10^{19}$ unterschiedliche Schlüssel.

Man bräuchte also höchstens $10^{19}$ Versuche, um den Schlüssel mit der Brute Force Methode herauszufinden.

Es existieren außerdem folgende Attacken, die Prämissen an Cipher- und Klartexte stellen [Schneier1,S.5-6]:

1. Ciphertext-Only Attacke: Der Angreifer verfügt über Ciphertexte eines von ihm bekannten Kryptosystems, die alle mit dem gleichen Schlüssel verschlüsselt wurden.
2. Known-Plaintext Attacke: Der Angreifer kennt Ciphertexte und die dazugehörigen Klartexte.
3. Chosen-Ciphertext Attacke: Der Angreifer kann von ihm ausgewählte Ciphertexte entschlüsseln lassen und hat Zugriff auf die entsprechenden Klartexte.
4. Adaptive Chosen-Plaintext Attacke: Erweitert die Chosen-Plaintext Attacke darin, dass der Angreifer zuvor frei ausgewählte und verschlüsselte Klartexte beliebig modifizieren und erneut verschlüsseln kann.

Diese Attacken lassen sich noch weiter kategorisieren. So sind die obigen ersten beiden Angriffe passiv, da der Angreifer Nachrichten oder Informationen abhört, um an Klart- oder Ciphertexte zu gelangen. Dagegen erfolgen die letzten beiden Attacken aktiv, da sich der Angreifer zu ver- oder entschlüsselnde Klar -bzw. Ciphertexte beliebig aussuchen kann, er also aktiv in das Protokoll eingreift [Me,Oo,Va,S.41].

**Beispielattacken**

Ciphertext-Only Attacke:

1. Es werden mehrere verschlüsselte Dokumente vom Angreifer abgefangen. Dabei kennt er das verwendete asymmetrische Kryptosystem. Zudem kennt er den Kommunikationskontext und so den groben Dokumentinhalt (z.B. Bank- oder Vertragsangelegenheiten).
2. Die Schlüssellänge des Kryptosystems ist extrem kurz. Er probiert alle möglichen Schlüssel Brute Force aus. Er kennt nicht den Klartext, so dass er sich bei einem entschlüsselten Dokument mit plausiblen Inhalt nicht hunderzprozentig sicher sein kann, dass es der ursprüngliche ist. Allerdings kannte er schon vor dem Entschlüsseln den groben Inhalt des Dokumentes, so dass es sehr wahrscheinlich ist, dass er den Klartext rekonstruiert und zudem den geheimen Schlüssel herausgefunden hat.

Known-Plaintext Attacke:

1. Der Angreifer kennt das asymmetrische Kryptosystem und hat zudem ein Paar aus dem verschlüsselten Ciphertext und dem zugehörigen Klartext.
2. Der Angreifer probiert alle möglichen Schlüsselkombinationen aus. Sobald er aus dem Ciphertext den Klartext oder entgegengesetzt den Klartext aus dem Ciphertext mit einer bestimmten Schlüsselkombination ermittelt hat, hat er den geheimen Schlüssel herausgefunden.

Chosen-Plaintext Attacke:

1. Der Angreifer hat Zugriff auf die Verschlüsselungseinheit des Kryptosystems, die einen bestimmten geheimen Schlüssel verwendet. Er verwendet die Verschlüsselungseinheit dafür, dass er einen selbstgewählten Klartext verschlüsselt. Z.B. wählt er einen Klartext, der nur aus Nullen besteht.
2. Er erhält den Ciphertext und erhofft sich aus dem einfachen Muster des Klartextes, dass sich offensichtliche Zusammenhänge zwischen Ciphertext und Klartext ergeben und so leicht Rückschlüsse auf den verwendeten Schlüssel möglich sind. Zudem können grundsätzliche Schwächen im Ver- und Entschlüsselungsalgorithmus aufgezeigt werden.

Adaptive Chosen-Plaintext Attacke:

1. Der Angreifer verschlüsselt einen frei gewählten Klartext mit einer Verschlüsselungseinheit, auf die er Zugriff besitzt[13].
2. Aufgrund des Ciphertextes vermutet er Schwächen im Algorithmus oder Rückschlüsse auf den Schlüssel. Um seine These zu bestätigen, modifiziert er den zuvor gewählten Klartext und verschlüsselt ihn erneut. Durch iterative Anwendung dieses Vorgangs kann er den Schlüssel berechnen und entdeckt Schwächen des Algorithmus.

**Folgerung**

Es zeigt sich, dass die Schlüssellänge ein wichtiger Aspekt der Sicherheit eines Kryptosystems ist. So verhindert bzw. erschwert ein langer Schlüssel eine realistische Brute Force Methode für einen Known-Plaintext oder Known-Ciphertext Angriff. Durch einen sehr großen Schlüsselraum würden einfach zu viele Resourcen in Zeit und Speicher benötigt werden, um eine erfolgreiche Attacke zu realisieren.

Natürlich sollte die Sicherheit eines Kryptosystem nicht nur über Schlüssellängen, sondern auch über einen guten und schwer brechbaren Ver- und Entschlüsselungsalgo-

13 Identisch zu Schritt 1) bei der Chosen-Plaintext Attacke

rithmus definiert werden.

### 3.4.4 Beispiel DES

DES (Data Encryption Standard) verwendet einen Blockalgorithmus und ist im Bereich der symmetrischen Kryptosysteme der am weitesten verbreitete Algorithmus [Schneier1, S.17]. Er basiert auf einem Algorithmus namens Lucifer, der von IBM entwickelt wurde. Nach Überprüfung und Evaluierung, u.a. von der NSA, wurde dieser schließlich 1976 als Standard veröffentlicht. Die Schlüssellänge beträgt 64 Bit. Allerdings werden 8 Bit als Paritätsbits[14] verwendet, so dass als Schlüssel nur 54 Bit für den Ver- und Entschlüsselungsalgorithmus zur Verfügung stehen.

DES ermöglicht durch die kurze Schlüssellänge von 54 Bit gute Erfolgsaussichten für Brute Force Attacken. So wurde DES bereits 1999 in einer Zeit von etwa 22 Stunden gebrochen [EFF], so dass 2000 der Nachfolgestandard AES veröffentlicht wurde. Ferner existiert noch der Triple DES, der den DES um eine Dreifachverschlüsselung erweitert.

### 3.4.5 Anwendungsgebiete symmetrischer Kryptosysteme

**Bankautomaten**

An Bankautomaten wird der Kunde zusammen mit seinen Daten auf dem Magnetstreifen der Bankkarte und der zugehörigen PIN (Personal Identifiation Number) authentifiziert. Nach Eingabe der PIN wird diese verschlüsselt, über ein Netzwerk versendet und in der Bankzentrale mit dem gleichen Schlüssel wieder entschlüsselt, so dass ein Abgleich möglich ist [Anderson,S.198-199].

**Secure Shell**

Secure Shell (kurz SSH) ist ein Protokoll[15], welches eine verteilte Kommunikation zwischen zwei Rechnern ermöglicht. Um ein Abhören zwischen den Rechnern zu verhindern, ist hier eine symmetrische Verschlüsselung der Kommunikation vorgesehen[16].

14 Paritätsbits werden für eine Fehlererkennung binärer Daten verwendet.
15 SSH wird oft auch als Programm bezeichnet, das das SSH-Protokoll implementiert.
16 Eigentlich basiert SSH auf einem Hybridkryptosystem (siehe S.34), in dem zwar die Nutzdaten symmetrisch verschlüsselt werden, der Schlüsseltausch des geheimen Schlüssels allerdings mit einem asymmetrischen Kryptosystem realisiert wird (siehe S.28).

## 3.5 Asymmetrische Kryptosysteme

Dass trotz einer symmetrisch verschlüsselten Verbindung dennoch das Sicherheitskriterium der Geheimhaltung (siehe S.14) verletzt werden kann, zeigt folgendes Beispiel:

1. Beide Austauschpartner einigen sich auf ein symmetrisches Kryptosystem. Der entsprechende geheime Schlüssel ist noch nicht auf beiden Seiten bekannt.
2. Der Absender des Dokumentes generiert einen geheimen Schlüssel. Dieser geheime Schlüssel muss dem Empfänger zugänglich gemacht werden. Der Schlüssel muss dementsprechend über einen Transportkanal übermittelt werden.
3. Der Absender verschlüsselt das Dokument und sendet es ab.
4. Ein Angreifer fängt sowohl das verschlüsselte als auch den geheimen Schlüssel ab und kann so unbefugt das Dokument entschlüsseln und lesen.

Zwei Faktoren für einen solchen erfolgreichen Angriff spielen eine Rolle: Erstens war eine Schlüsselaustausch nötig, da sowohl auf Absender als auch Empfängerseite der geheime Schlüssel bekannt sein muss. Zweitens geschah dies über einen Transportkanal, der grundsätzlich als unsicher einzustufen ist.

Da die Schlüssel ausgetauscht werden müssen, ist hier der einzige Weg, den Angriff zu verhindern, die Schlüssel über einen absolut sicheren Transportkanal auszutauschen. Dies ist unrealistisch, da Transportkanäle grundsätzlich als unsicher[17] einzustufen sind (siehe Definition auf S. 15). Das große Problem der symmetrischen Kryptosysteme ist demnach die Schlüsselverteilung. Dieses Problem können asymmetrische Kryptosysteme[18] lösen, die für Ent- und Verschlüsselung unterschiedliche Schlüssel verwenden.

### 3.5.1 Mathematische Grundlagen

Ein Klartext $kl \in Kl$ wird mit einem öffentlichen Schlüssel aus einer Schlüsselmenge $s_{öffentlich} \in S_{ö}$ mit einer Verschlüsselungsfunktion $v$ verschlüsselt. Dabei entsteht ein Ciphertext $c = v(kl, s_{öffentlich})$ . Dieser Ciphertext lässt sich wieder entschlüsseln, allerdings nur mit einer Entschlüsselungsfunktion $e$ und einem geheimen Schlüssel aus einer Schlüsselmenge: $s_{geheim} \in S_g$ , so dass der Klartext wieder hergestellt wird

17 Außer die Personen würden sich persönlich an einem geheimen abhörsicheren Ort treffen und die Schlüssel austauschen. Aber dann könnte das Dokument ja gleich persönlich übergeben werden.

18 In der Literatur wird auch oft von Public Key Kryptosystemen gesprochen.

$kl = e(c, s_{geheim})$ . Beide Vorgänge lassen sich auch ausdrücken als $kl = e(v(kl, s_{öffentlich}), s_{geheim})$ .

Interessant ist, dass der gleiche Vorgang in die andere Richtung umgekehrt werden kann: Der Klartext kann also mit dem geheimen Schlüssel verschlüsselt und mit dem öffentlichen entschlüsselt werden. Es gilt also: $kl = e(v(kl, s_{geheim}), s_{öffentlich})$ .

### 3.5.2 Protokoll

Folgendes Protokoll zeigt, wie das Schlüsselverteilungsproblem mit asymmetrischer Verschlüsselung gelöst werden kann:

1. Absender und Empfänger einigen sich auf ein Kryptosystem.
2. Der Empfänger generiert ein Schlüsselpaar aus öffentlichem und geheimen Schlüssel.
3. Der Empfänger sendet dem Absender den öffentlichen Schlüssel.
4. Der Absender des Dokumentes verschlüsselt das Dokument mit dem öffentlichen Schlüssel des Empfängers und sendet das Dokument ab.
5. Der Empfänger nimmt das Dokument entgegen und entschlüsselt dieses mit dem geheimen Schlüssel und erhält so das Dokument als Klartext.

Wie zu sehen, muss der öffentliche Schlüssel nicht geheim gehalten werden und kann so ohne weitere Vorkehrungen über den Transportkanal ausgetauscht werden, wodurch das Sicherheitsproblem des Schlüsselaustauschs gelöst wurde.

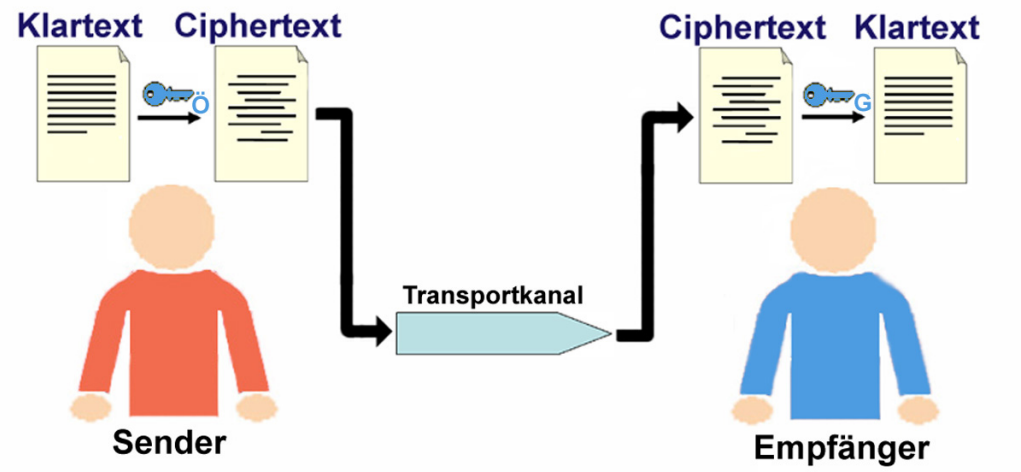


Abbildung 4: Der Sender verschlüsselt mit dem öffentlich zugänglichen Schlüssel des Empfängers. Der Ciphertext kann nur mit dem geheimen Schlüssel des Empfängers entschlüsselt werden. Beide Schlüssel sind dem Empfänger zugeordnet.

### 3.5.3 Attacken

Da bei asymmetrischer Verschlüsselung ebenfalls Schlüssel verwendet werden, um Nachrichten zu ver- und entschlüsseln, decken sich die Angriffstypen mit denen der symmetrischen Verschlüsselung (siehe S.24). Ziel ist erneut ohne Kenntnis des geheimen Schlüssels diesen zu ermitteln oder den Klartext zu rekonstruieren.

Da der öffentliche Schlüssel für potentiell jeden verfügbar gemacht wird, ergibt sich, da die Sicherheit der meisten asymmetrischen Algorithmen auf das Faktorisierungsproblem beruht, weiterhin eine Brute Force Attacke, in der versucht wird, eine große Zahl zu faktorisieren und auf diese Weise den geheimen Schlüssel aus dem öffentlichen abzuleiten [Schneier1,S.158].

### 3.5.4 Beispiel RSA

RSA, benannt nach seinen Erfindern Rivest, Shamir and Adleman, ist der am weitesten verbreitete asymmetrischer Algorithmus. Man vermutet seine Sicherheit auf der Basis, dass es sehr aufwendig ist[19], hohe Primzahlen in ihre Faktoren zu zerlegen [Schneier1,S.470]. RSA hat dabei die Faktorisierungs-Einwegfunktion[20] mit einer Falltür versehen, um es für ein Verschlüsselungsverfahren mit öffentlichen und geheimen Schlüssel nutzbar zu machen [Anderson, S.105].

### 3.5.5 Anwendungsgebiete asymmetrischer Kryptosysteme

**SSL**

SSL ist ein Verschlüsselungsprotokoll für die Internetdatenübertragung. Die Asymmetrische Verschlüsselung wird dabei verwendet, um den geheimen Schlüssel für die eigentliche symmetrischen verschlüsselte Nutzdatenübertragung auszutauschen (siehe Hybridverschlüsselung S.34).

**Digitale Signaturen**

Digitale Signaturen (siehe S.36) ermöglichen eine Unterschrift eines elektronischen Dokumentes. Dabei wird der generierte Hashwert nicht mit dem öffentlichen sondern mit dem geheimen Schlüssel des asymmetrischen Schlüsselpaares verschlüsselt.

19 Es wird vermutet, dass das Problem nur NP lösbar ist.
20 Für hohe Zahlen ist es einfach Produkte zu bilden, allerdings viel aufwendiger die entstandene Zahl zu faktorisieren.

# 3.6 Schlüssellebenszyklus

Das Kerckhoff-Prinzip definiert die Sicherheit eines schlüsselbehafteten Kryptosystems auf die Weise, dass es lediglich von der Geheimhaltung der Schlüssel abhängen darf. Allerdings spielt in der Praxis die Sicherheitsproblematik der Generierung, Verteilung und Verwaltung eine wichtige Rolle.

## 3.6.1 Schlüsselgenerierung

Zunächst muss bei symmetrischen Kryptosystemen ein geheimer Schlüssel und bei asymmetrischen ein Schlüsselpaar aus geheimen und öffentlichen Schlüssel generiert werden. Hier wird auch festgelegt welches Kryptosystem verwendet wird, da nicht alle die gleiche Schlüssellänge verwenden.

Hier ist auch wichtig, dass adequate Schlüssellängen ausgewählt werden, da bei der Annahme, dass der Algorithmus an sich nicht brechbar ist[21], der primäre Schutz vor einem Angriff die Schlüssellänge ist. Zudem sollte der geheime Schlüssel nicht von einem Benutzer gewählt werden, da sonst die Gefahr von Wortbuchattacken droht, die den Aufwand für eine Schlüsselkompromittierung erheblich minmiert. Deswegen sollten Schlüssel zufällig generiert werden [Schneier1, S.173].

## 3.6.2 Schlüsseltransport

Die Schlüssel müssen den Teilnehmern des Protokolls zugänglich gemacht werden. Dabei ergeben sich Unterschiede bei den Kryptosystemtypen.

**Symmetrisches Kryptosystem:**

Die verschlüsselte Information auf Empfängerseite lässt sich nur mit einem geheimen Schlüssel entschlüsseln. Der geheime Schlüssel muss also ebenfalls übertragen werden. Da Transportkanäle per Definition unsicher sind, ergibt sich immer das Problem, dass der Schlüssel kompromittierbar ist. Eine mögliche, aber immer noch nicht absolut sichere Lösung wäre, die verschlüsselten Daten über einen anderen Kanal zu senden als den Schlüssel [Schneier1,S.177]. So ist es relativ unwahrscheinlich, dass der Angreifer sowohl eine verschlüsselte Nachricht über das Internet abfängt, ein Telefongespräch abhört und dieses auch noch mit der Nachricht in Beziehung setzen kann.

21 Die effizienteste und einzige Attacke auf das Kryptosystem ist also die Brute Force Attacke.

**Asymmetrisches Kryptosystem:**

Da öffentliche Schlüssel verwendet werden, um Informationen zu ver- oder entschlüsseln[22], stellt sich das Problem des unsicheren Transportkanalproblems nicht, da öffentliche Schlüssel für jeden sichtbar sein können. Da der öffentliche Schlüssel einem Kommunikationsteilnehmer direkt zugeordnet ist, kann man auch von einem Inhaber des öffentlichen Schlüssels reden.

Bei beiden Kryptosystemtypen ist es allerdings möglich, dass während des Schlüsseltransports den Kommunikationspartnern unechte Schlüssel untergeschoben werden, die Authentizität der Schlüssel also nicht gewährleistet ist. Dies wird bei asymmetrischen Kryptosystem durch sogenannte Zertifikate (siehe S.33) verhindert.

### 3.6.3 Schlüsselverwaltung

Die Kommunikationsteilnehmer müssen die generierten bzw. vom Gegenüber erhaltenen Schlüssel auf ihrer Seite verwalten. Dabei ergeben sich wieder Unterschiede bei den Kryptosystemklassen.

**Symmetrisches Kryptosystem:**

Jedes Kommunikationspaar[23] muss Kopien des geheimen Schlüssels untereinander besitzen. Dabei ergibt sich bei vielen Kommunikationspartnern gemessen an der Menge der zu verwaltenden Schlüssel ein immenser Aufwand. Bei $n$ Kommunikationspartnern, die alle untereinander verschlüsselt kommunizieren wollen (Point to Point Kommunikation) ergibt sich durch Menge an Schlüsseln $schlüsselmenge = n(n-1)/2$, die zudem noch geheimgehalten werden müssen. Dieser Aufwand von $O(n^2)$ ist besonders bei vielen Kommunikationsteilnehmern sehr hoch.

**Asymmetrisches Kryptosystem:**

Es existiert der öffentliche und geheime Schlüssel, den jeder Kommunikationsteilnehmer einmal besitzt. Der Besitzer des Schlüsselpaares gibt seinen öffentlichen Schlüssel heraus, so dass alle Kommunikationsteilnehmer mit diesem ihre Informationen verschlüsseln bzw. die auf Seite des Schlüsselinhabers verschlüsselten

22 Öffentliche Schlüssel werden bspw. bei Digitalen Signaturen für eine Entschlüsselung verwendet.
23 Kommunikationspaare müssen nicht notwendigerweise aus zwei Personen bestehen. Es kann sich jeweils auch um einen Personenkreis handeln, der dann allerdings keine Geheimnisse untereinander mehr haben kann.

entschlüsseln können. Insgesamt ergibt sich bezogen auf die Kommunikationspartner eine Schlüsselmenge von $schlüsselmenge = 2n$ und dadurch ein annehmbarer Aufwand der Schlüsselverwaltung von $O(n)$ .

### 3.6.4 Zertifikate

Um die Notwendigkeit von sogenannten Zertifikaten zu verstehen, folgt ein Angriffsbeispiel auf ein asymmetrisches Kryptosystem, in dem das Sicherheitsziel der Authentizität (siehe S.14) verletzt wird:

1. Ein Angreifer gibt sich gegenüber dem Dokumentabsender für jemanden aus, der er nicht ist. Dabei übermittelt er dem Dokumentabsender seinen öffentlichen Schlüssel.
2. Der Absender glaubt dem Angreifer, dass dieser eine Person ist, dem eine Einsicht in das Dokument erlaubt ist und verschlüsselt sein Dokument mit dessen öffentlichen Schlüssel.
3. Der Angreifer erhält das Dokument und kann es mit seinem geheimen Schlüssel entschlüsseln und lesen.

Hier liegt also eine Verletzung der Schlüsselauthentizität vor. Um diesem vorzubeugen muss eine Instanz existieren, die dem Schlüsselempfänger zusichert, dass der entsprechende öffentliche Schlüssel auch der Person zugeordnet ist, mit dem sie eine verschlüsselte Kommunikation durchführen will. Es muss also eine Schlüsselverifikation des öffentlichen Schlüssels möglich sein.

#### Trusted Third Party

Die Instanz, die eine Schlüsselauthentizität gewährleistet, wird u.a. als Trusted Third Party (TTP) bezeichnet [Me,Oo,Va,S.491]. Sie dient als Vermittler zwischen Kommunikationspartnern, wobei die Vorraussetzung ist, dass die Kommunikationspartner dieser Trusted Third Party vertrauen müssen. Im Zusammenhang mit asymmetrischen Kryptosystemen[24] versuchen sie, einen öffentlichen Schlüssel mit Attributen wie Name, Adresse, Beruf o.ä. in Beziehung zu setzen und so den Schlüssel an eine Person zu binden. Dadurch wird für den Anwender des öffentlichen Schlüssels nachvollziehbar, dass er einen authentischen Schlüssel verwendet hat. Es ist auch möglich, dass die Trusted Third Party das Schlüsselpaar selber generiert und bspw. auf einer Smartcard

24 Trusted Third Parties werden auch für die Gewährleistung von Protokollsicherheit symmetrischer Kryptosysteme verwendet [Me,Oo,Va,Kapitel 8]. In dieser Arbeit wird die TTP aber lediglich im Zusammenhang asymmetrischer Systeme erwähnt.

speichert und dem eigentlichen Schlüsselpaarinhaber zusendet[25]. Als Ergebnis des Prozesses, in dem ein öffentlicher Schlüssel einer bestimmten Instanz zugeordnet wird, entsteht das Zertifikat.

**PKI**

Infrastrukturen, die asymmetrische Kryptosysteme in ihrem Schlüssellebenszyklus als Vermittler unterstützen, werden als Public Key Infrastrukturen (PKI) bezeichnet . Die PKI nimmt dementsprechend auch die Rolle einer TTP ein. Seitens der PKI werden Vorgaben gemacht, wie die Schlüssel und Zertifikate generiert, herausgegeben und verwaltet werden [Choudhury].

PKI können sowohl hierarchisch als auch verteilt aufgebaut sein. Bei einer hierarchischen Struktur vertraut man einer übergeordneten Instanz, die die PKI Maßnahmen übernimmt[26]. Bei der verteilten Struktur ist die vertrauenswürdige Instanz nicht eine zentrale, in der Hierarchie oben stehende, sondern das Vertrauen basiert auf anderen schon registrierten Nutzern, die bereits öffentliche Schlüssel besitzen und für neue Nutzer bürgen und deren öffentliche Schlüssel signieren[27].

## 3.7 Hybrides Kryptosystem

Symmetrische und asymmetrische Kryptosysteme haben allein eingesetzt u.a. folgende Vor- und Nachteile [Me,Oo;Va,S.31-32]:

**Symmetrische Kryptosysteme:**

| Stärken | Schwächen |
|---|---|
| Die Schlüssel sind relativ kurz und die Algorithmen sehr performant. | Die geheimen Schlüssel müssen auf beiden Seiten bekannt sein und deswegen ausgetauscht werden (Schlüsseltransportproblem). |
| | Auf beiden Seiten muss der Schlüssel geheim gehalten werden. Zudem ist die zu verwaltende Schlüsselmenge bei vielen Kommunikationspartnern sehr hoch (Schlüsselverwaltungsproblem). |

25 Als Beispielsdienstleister siehe http://www.signcubes.de/; Zugriff: 02.09.06
26 Als Beispiel ist der X.509 Standard zu nennen [IETF X.509].
27 Als Beispiel ist Pretty Good Privacy (PGP) zu nennen. Zu den Einzelheiten siehe [PGP].

**Asymmetrische Kryptosysteme:**

| Stärken | Schwächen |
|---|---|
| Nur der geheime Schlüssel muss geheim bleiben. Der öffentliche kann offen zugänglich sein, wobei dessen Authentizität aber z.B. durch Zertifikate sichergestellt sein muss. | Die Schlüssel sind relativ lang und die Algorithmen im Vergleich zu assymetrischen sehr imperformant. |
| Die zu verwaltende Schlüsselmenge bei vielen Kommunikationspartnern steigt linear an. | |

Beim Vergleich der Kryptosystemklassen ist zu erkennen, dass eine Kombination beider eine gute Ergänzung ergeben würde: Symmetrische Kryptosysteme sind in der Regel 1000 Mal schneller als die asymmetrischen [Schneier1,S.33] und die Probleme der Schlüsselverteilung- und verwaltung lassen sich durch asymmetrische Systeme vereinfachen.

### 3.7.1 Protokoll

Ziel ist, dass aus Performanzgründen die Nutzdaten, also die eigentliche Information, die beide Kommunikationspartner miteinander austauschen wollen und die in der Regel den größten Datenanteil haben, symmetrisch verschlüsselt werden.

Folgendes Protokoll ist für eine sichere und performante Kommunikation bei Verwendung beider Kryptosystemklassen denkbar:

1. Der Absender fordert den öffentlichen Schlüssel des Empfängers an und validiert diesen mit Hilfe eines Zertifikats.
2. Der Absender generiert einen zufälligen geheimen sogenannten Sitzungsschlüssel für eine symmetrische Verschlüsselung. Nun wird der Sitzungsschlüssel mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Schließlich werden das symmetrisch verschlüsselte Dokument und der asymmetrisch verschlüsselte Sitzungsschlüssel über den Transportkanal versendet.
3. Der Empfänger erhält das verschlüsselte Dokument und den verschlüsselten Sitzungsschlüssel. Jetzt entschlüsselt er den Sitzungsschlüssel mit dem geheimen Schlüssel des asymmetrischen Kryptosystem Schlüsselpaares und erhält so den Sitzungsschlüssel als Klartext.
4. Mit dem Sitzungsschlüssel kann der Empfänger das Dokument, den eigentli-

chen Inhalt der sicheren Kommunikation, entschlüsseln.

Dadurch ergibt sich der Vorteil, dass der geheime Schlüssel als Ciphertext über den unsicheren Transportkanal übertragen und der Aufwand für die Schlüsselverwaltung, da der öffentliche Schlüssel für jeden verfügbar ist, vermindert werden kann. Als geheimer Schlüssel für den symmetrischen Algorithmus ist der Sitzungsschlüssel dabei nur für eine begrenzte Zeit gültig[28].

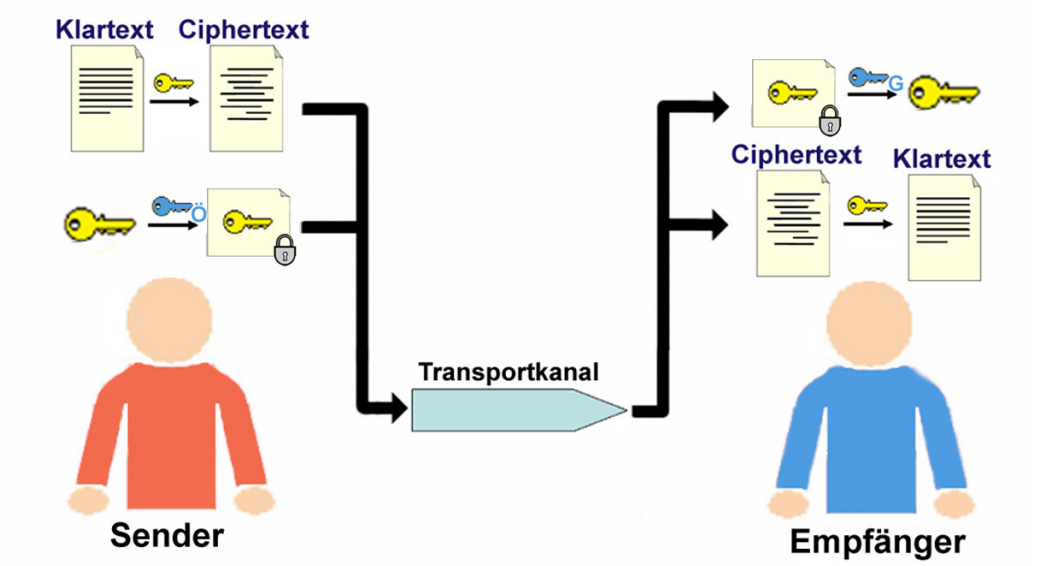


Abbildung 5: Der symmetrische Sitzungsschlüssel wird für die Dokumentenverschlüsselung verwendet, das asymmetrischen Schlüsselpaar für die Ver -bzw. Entschlüsselung des Sitzungsschlüssels. Dabei ist das asymmetrische Schlüsselpaar dem Empfänger zugeordnet.

## 3.8 Digitale Signatur

Dokumentenaustauschvorgänge setzen oft Unterschriften voraus[29]. Solche Unterschriften will man im Falle des elektronischen Dokumentenaustausches nachahmen. Dabei existieren folgende Ziele [Me,Oo,Va,S.22]:

1. Das Dokument stammt vom vorgegeben Absender (Authentizität, siehe S.14).
2. Das Dokument wurde nicht gefälscht (Integrität, siehe S.14).
3. Der Absender kann die Autorenschaft des Dokumentes nicht abstreiten (Nichtabstreitbarkeit, siehe S.14).

Diese Anforderungen kann durch die Digitale Signatur umgesetzt werden, die bisher vorgestellte Kryptosystemklassen kombiniert.

28 Als Beispiele siehe SSH (S.27).
29 Als Beispiel siehe Schriftform S.7.

### 3.8.1 Protokoll

Asymmetrische Kryptosysteme können verwendet werden, um Informationen mit dem Ziel der Geheimhaltung zu verschlüsseln. Wenn man die Anwendung der Schlüssel umkehrt, ist mit asymmetrischen Kryptosystemen ebenfalls eine Authentikation möglich[30]. Mit Hilfe von Einweg Hashfunktionen lässt sich dabei die Datenmenge der Signatur deutlich reduzieren und auch die Integrität sicherstellen. Folgendes Protokoll realisiert eine Digitale Signatur:

1. Der Absender verwendet eine Einweg Hashfunktion, mit der er das Dokument auf einen Hashwert abbildet. Der entstandene Hashwert repräsentiert die Signatur des Dokumentes.
2. Der Absender besitzt ein asymmetrisches Schlüsselpaar, dessen öffentlicher Schlüssel durch eine Trusted Third Party zertifiziert wurde.
3. Der Absender verschlüsselt den Hashwert des Dokumentes mit seinem geheimen Schlüssel und sendet es ab. Zusammen mit der verschlüsselten Signatur wird das Dokument abgesendet.
4. Der Empfänger erhält das Dokument und die verschlüsselte Signatur. Er führt eine erfolgreiche Schlüsselverifikation des öffentlichen Schlüssels über das Zertifikat durch, so dass nachgewiesen ist, dass der öffentliche Schlüssel dem Sender zugeordnet ist.
5. Der Empfänger generiert mit der gleichen Funktion wie der Absender den Hashwert des Dokumentes und vergleicht diesen mit der entschlüsselten Signatur. Bei Gleichheit ist sichergestellt, dass das Dokument vom Inhaber des öffentlichen Schlüssels stammt.

Die Authentizität des Dokumentes ist in diesem Fall dadurch sichergestellt, dass beide Hashwerte gleich waren, der öffentliche Schlüssel an den Absender gebunden ist und nur er die entsprechende Verschlüsselung mit dem nur ihm bekannten geheimen Schlüssel durchführen konnte.

Die Verwendung der Einweg Hashfunktionen stellt außerdem sicher, dass das Dokument nicht gefälscht wurde. Zudem kann der Absender später nicht behaupten, ein anderes Dokument abgesendet zu haben, da dieses einen anderen Hashwert ergeben würde. Falls das Dokument mit verschlüsselter Signatur archiviert werden würde, ist

30 Symmetrische Kryptosysteme können ebenfalls für eine Authentisierung (siehe [Schneier1,S.36]) verwendet werden, allerdings ist der entsprechende Vorgang umständlicher und für die Arbeit nicht direkt relevant, so dass nicht näher darauf eingegangen wird.

dem Absender sogar nachträglich nachweisbar, dass das Dokument von ihm stammt.

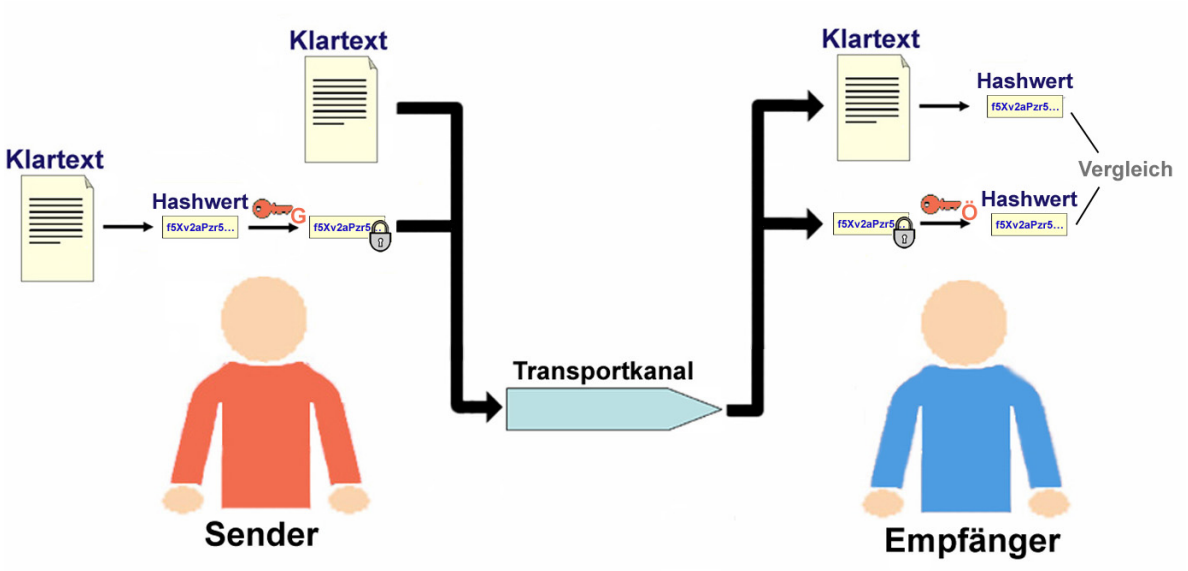


Abbildung 6: Die Integrität wird durch den Vergleich des vom Sender und Empfänger gebildeten Hashwertes gesichert. Die Sicherstellung der Integrität wird dadurch begründet, dass nur der Sender den geheimen Schlüssel kennt.

Das Sicherstellen von Authentizität, Integrität und Nichtabstreitbarkeit kann durch Digitale Signaturen natürlich nur erreicht werden, wenn der geheime Schlüssel nicht kompromittiert wurde und Attacken auf die konkret eingesetzten Kryptosysteme grundsätzlich nicht erfolgreich sind.

### 3.8.2 Zeitstempel

Die Signatur kann neben dem Hashwert auch ein Zeitstempel enthalten, um den Zeitpunkt des Signierens zu erfassen. Dabei werden die Daten mit Zeitangaben verknüpft. Hier wird das gesamte zu signierende Dokument mit einer Zeitangabe oder für die Einhaltung der Vertraulichkeit nur der Hashwert des Dokumentes mit einem Zeitstempel versehen. Die letztere Variante ist besonders bei Verwendung externer Zeitstempeldienstleister[31] sinnvoll, die die Dokumenteninhalte nicht kennen sollen.

## 3.9 Challenge Response Verfahren

Folgender Angriff ist auf eine Kommunikation ist möglich:

1. Der Sender sendet einen Auftragsdokument an eine Stelle und erwartet eine Antwortschreiben als Dokument.
2. Ein Angreifer weiß, dass eine Kommunikation stattfindet und sendet ein korrum-

31 Siehe qualifizierter Zeitstempel S.11.

piertes Antwortschreiben an den Sender. Dieser nimmt an, dass das Antwortschreiben vom eigentlichen Empfänger stammt.

Es ergibt sich also ein Problem der Authentizität, da sich der Angreifer für eine Instanz ausgibt, die er nicht ist.

### 3.9.1 Protokoll

Dieses Problem der Authentizität kann mit dem Protokoll des Challenge Response Verfahrens gelöst werden. Im Gegensatz zur Digitalen Signatur, die ebenfalls die Authentizität sicherstellt, wird im Challenge Response Protokoll selbst kein Kryptosystem vorgeschrieben:

1. Der Sender sendet ein Auftragsdokument und zusätzlich einen sogenannten Challenge, der einen beliebigen Wert darstellt[32].
2. Der Empfänger antwortet mit einem Antwortschreiben. Darüberhinaus interpretiert er den Challenge und antwortet entsprechend mit einem Responsewert.
3. Der ursprüngliche Sender des Auftragsdokuments erhält das Antwortschreiben und den Response Wert. Wenn der Response zum Challengewert passt, kann er sich der Authentizität der Antwort sicher sein.

Die Abbildung des Challenges auf den Responsewert ist nicht festgelegt. So kann der Challenge die Aufforderung einer Response eines Passworts, wie beim typischen Login oder auch nur die Wiederholung des Challengewertes[33] sein.

32 Challenge ist mit Aufforderung aus dem englischen zu übersetzen. Der Empfänger muss also dieser Aufforderung nachkommen und entsprechend antworten.

33 Durch die einfache Wiederholung des Challengewertes als Response werden z.B. in OSCI (siehe S.40) Dialognachrichten authentifiziert.

# 4 OSCI

## 4.1 Einleitung

Für die Signatur eines elektronischen Dokumentes existiert eine technische Umsetzungsmöglichkeit in Form der Digitalen Signatur. Auch liegen rechtliche Rahmenbedingungen für das Verwaltungsverfahren vor, wodurch grundsätzlich die elektronische Form gegenüber der Schriftform gleichberechtigt ist. So ist es legitim, dass ein Antragsteller mit einer qualifizierten elektronischen Signatur, umgesetzt durch das Verfahren der Digitalen Signatur, ein elektronisches Dokument signiert und an eine Verwaltungsinstanz sendet.

Um entsprechende Potentiale von Technologien in der öffentlichen Verwaltung rechtskonform zu nutzen und entsprechende IT Vorhaben in der öffentlichen Verwaltung abzustimmen, existiert mit dem KoopA ADV (Kooperationsausschuss für Automatisierte Datenverarbeitung) ein Gremium, das mit Vertretern aus Bund, Ländern und kommunalen Spitzenverbänden besetzt ist. Durch die KoopA ADV wurde die OSCI Leitstelle beauftragt, Standards für eine sichere Datenübermittlung in der öffentlichen Verwaltung zu entwickeln.

Ergebnis ist der OSCI Standard, mit dem eine sichere Datenübermittlung möglich ist. OSCI wurde dabei in der aktuellen Version 1.2 durch das BSI geprüft, im Speziellen für E-Government als geeignet befunden und über die Standards und Architekturen für E-Government Anwendungen (SAGA) sogar vorgeschrieben[1].

Als Standard ist OSCI in den zwei Spezifizierungsteilen A und B untergegliedert. OSCI A spezifiziert ein Protokoll und ein Nachrichtenformat, welche den sicheren und einheitlichen Transport betreffen. OSCI B konkretisiert dagegen fachspezifische Nachrichtenformate[2], geht also direkt auf die relevanten fachlichen Informationen in einem Verwaltungsverfahren ein. Folgend soll nur der Spezifikationsteil des OSCI Transports behandelt werden, mit OSCI ist im weiteren also immer OSCI A gemeint.

1 Siehe www1.osci.de/sixcms/detail.php?gsid=bremen02.c.1160.de ;Zugriff: 10.10.06
2 Bspw. XMeld für Meldungswesen oder XGewerb für Gewerbewesen

## 4.2 Anforderungen an den OSCI Standard

Primäre treibende Kraft des OSCI Standards waren die rechtlichen Rahmenbedingungen und die effiziente und kostengünstige Umsetzung einer sicheren Kommunikation [OSCIUeber, S.1].

Die rechtliche Seite gibt vor, dass die elektronische Form, die gleichberechtigt zur Schriftform ist, eine qualifizierte elektronische Signatur vorweisen muss, um die Integrität, Authentizität und Nichtabstreitbarkeit der Autorenschaft eines Dokumentes sicherzustellen. Desweiteren muss der Datenschutz beachtet und entsprechende Maßnahmen für eine Geheimhaltung müssen getroffen werden[3]. Die Verknüpfung dieser rechtlichen Vorgaben mit kryptografischen Sicherheitstechnologien sollte durch einen Standard entsprochen werden, der einen sicheren Datenaustausch in der öffentlichen Verwaltung möglich macht.

Trotz der Mehraufwendungen für obige Sicherheitsziele sollte eine effiziente und kostengünstige Lösung gefunden werden. Komplexität sollte, wo es möglich war, geeignet reduziert werden und die Integration in die öffentliche Verwaltung sollte auf einfache Weise realisierbar sein.

## 4.3 Konzept des OSCI Transports

Um den oben genannten Anforderungen an sich selbst zu entsprechen, wurde die Nachrichtenstruktur, das Protokoll und das dazugehörige Rollenmodell folgendermaßen entworfen.

### 4.3.1 Nachrichtenstruktur

In der OSCI Nachricht erfolgt eine Trennung zwischen Nutzungsdaten und Inhaltsdaten. Die Nutzungsdaten sind dabei relevant für das Transportprotokoll und die Inhaltsdaten stellen die fachlichen Daten dar. Inhaltsdaten sind selbst innerhalb einer OSCI Nachricht in Inhaltscontainern gekapselt. OSCI Transport legt dabei nicht fest, welche Form die Inhaltsdaten haben müssen[4]. Die Nutzungsdaten orientieren sich dagegen streng nach einem vorgegebenen XML Schema.

Die Nutzungsdaten repräsentieren die Informationen, die für einen sicheren Dokumentenaustausch erforderlich sind. Dazu gehören die Zertifikate der Beteiligten (Leser, Au-

3 Für eine detaillierte Betrachtung dieser Sicherheitsziele siehe S.14.

4 Ausnahme sind die die domänenspezifischen Formate in der OSCI B Spezifikation, die sich nach der Schema Definition XSD richten und in XML gehalten sind.

tor, Sender und Empfänger), die der Intermediär als Vermittlerstelle benötigt, um eine Zertifikatsüberprüfung durchzuführen (siehe S.). Darüberhinaus sind in den Nutzungsdaten protokollrelevante Daten, Betreffinformationen und Zeitstempel enthalten.

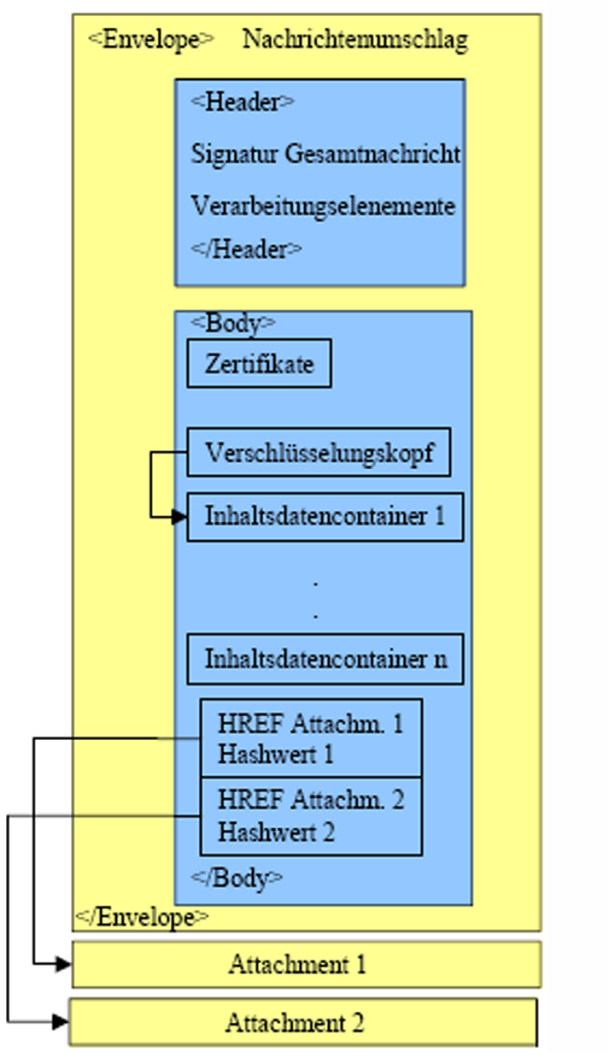


Abbildung 7: Die Nachricht orientiert sich an der SOAP Struktur und ist in Envelope, Header, Body und Attachementbereich aufgeteilt. Abbildung aus [OSCIPrinc,S.11].

### 4.3.2 Rollenmodell

Im OSCI Rollenmodell (siehe [OSCIPrinc,S.ff]) existieren beim elektronischen Dokumentenaustausch die Rollen Sender, Empfänger, Intermediär, Autor und Leser.

**Autor**

Der Autor erzeugt die Inhaltsdaten (siehe oben Abschnitt Nachrichtenstruktur). Daher ist die Rolle des Autors auf der Geschäftsvorfallsebene tätig. Dabei können mehrere

Autoren den Inhaltsdaten zugeordnet werden. Die erstellten Inhaltsdaten werden von den Autoren optional signiert und/oder verschlüsselt. Das Signieren von mehreren Autoren setzt eine Mehrfachsignatur voraus. Eine optionale Verschlüsselung für mehrere Leser wird durch eine Mehrfachverschlüsselung seitens der Autoren realisiert.

**Leser**

Die Personen, für die die Inhaltsdaten verfasst wurden, stellen die Leser dar. Diese kommunizieren mit den Autoren also auf Geschäftsvorfallsebene. Es können dabei mehrere Leser für eine Nachricht existieren. Ggf. wird die Signatur überprüft und eine Entschlüsselung der Inhaltsdaten vorgenommen.

**Sender**

Die Inhaltsdaten werden vor dem Versand um Nutzungsdaten (siehe oben Abschnitt Nachrichtenstruktur) ergänzt und versendet, der Sender ist daher auf der Auftragsebene tätig. Die Nutzungsdaten können optional verschlüsselt und signiert werden. Die Erstellung der Nutzungsdaten und der Versand der gesamten Nachricht übernimmt dabei immer nur ein Sender, so dass eine Mehrfachsignatur ausgeschlossen wird. Da immer nur ein Empfänger während einer OSCI Sendung existiert, wird eine Mehrfachverschlüsselung seitens des Senders nicht unterstützt.

**Empfänger**

Die Instanz, die der Adressat aus Sendersicht ist und die OSCI Nachricht entgegennimmt ist der Empfänger. Dabei interpretiert dieser nur die Nutzungsdaten, wodurch er innerhalb des OSCI Protokolls auf der Auftragsebene kommuniziert.

**Intermediär**

Durch die Rolle des Intermediärs werden an zentraler Stelle[5] zusätzliche Möglichkeiten im OSCI Nachrichtenaustausch geleistet [OSCIPrinc]:

- Es ist eine asynchrone Kommunikation möglich, so dass Sender und Empfänger bei einer Kommunikation nicht zwingend gleichzeitig online sein müssen.
- Die Komplexität der OSCI Szenarien (siehe S.44) wird teilweise auf den Intermediär verlagert, so dass die Implementierungen von Sender und Empfänger

5 Diese Zentralität aller relevanten Dienste wird oft mit dem Begriff 'single point of entry' verbunden. Im E-Government Kontext ist die Leistung solcher Mehrwertdienste auch oft mit dem Begriff der Virtuellen Poststelle verknüpft. siehe www.datenschutz-bayern.de/technik/orient/virtpost.html#ab8.3; Zugriff: 21.11.06

leichter realisiert werden können.

- Es kann ein von Sender und Empfänger unabhängiger Quittierungsmechanismus eingesetzt werden. Auf diese Weise ist sichergestellt, dass weder Sender noch Empfänger Quittungen manipulieren[6]. Hier kann der Intermediär u.a. Zeitstempel setzen, um die Eingänge von OSCI Nachrichten mit Zeitpunktangaben zu protokollieren. Der Quitterungsmechanismus wird durch einen sogenannten Laufzettel realisiert.
- Die Prüfung[7] der Zertifikate von Autor, Leser, Sender und Empfänger kann durch den Intermediär zentral durchgeführt werden.
- Der Intermediär kann als Formularserver dienen, der unausgefüllte, elektronische Formulare einer Behörde bereitstellt.

Der Intermediär kommuniziert wie Sender und Empfänger auf Auftragsebene und ist deswegen von den Inhaltsdaten komplett unabhängig. So können diese verschlüsselt und eigens signiert werden, um auch sicherzustellen, dass der Intermediär keine Zugriff erhält.

Anhand der Mehrwertdienste kann der Intermediär die charakeristische Rolle einer Poststelle einnehmen. Dabei wird das Konzept einer offenen Benutzergruppe unterstützt, bei der ein Sender sich für eine OSCI Kommunikation nicht explizit registrieren muss. Die Identifikation des Empfängers erfolgt dabei durch Zuordnung eines Zertifikats [OSCIPrinc,S.4], die wiederum an entsprechende Empfängerpostkörbe beim Intermediär gekoppelt sind. Daraus folgt, dass der Empfänger zunächst ein Zertifikat besitzen muss und der Sender es als indirekte Adressangabe der Nachricht beifügt.

### 4.3.3 Protokoll

Die konkrete Nachrichtenreihenfolge, die Verarbeitungsregeln und die Festlegung, welche Instanz des Rollenmodells (siehe S.42) bei der Kommunikation beteiligt ist, wird durch das Protokoll festgelegt. Ein damit zusammenhängender Dokumentenaustausch zweier Kommunikationspartner kann dabei auf verschiedene Art und Weise, auch abhängig von den fachlichen Anforderungen, erfolgen. Dazu spezifiziert OSCI drei Grundszenarien [OSCIMedia][8]:

1. One-Way-Message, aktiver Empfänger: Asynchrone Zustellung einer Nachricht

6 Vorraussetzung ist natürlich, dass der Intermediär als eine Stelle des Vertrauens arbeitet und sich selber gegenüber Manipulationen von außen schützt.

7 Für die zu tätigenden Aktionen einer Zertifikatsprüfung von X.509 Standards siehe [Bertsch,S.120].

8 Folgende Beschreibungen geben lediglich einen Überblick, für Protokolldetails siehe [OSCISpec,S.13ff.].

des Senders an den Empfänger. Der Sender versendet die Nachricht zunächst an die Vermittlerstelle des Intermediärs. Der adressierte Empfänger muss selber tätig werden und aktiv die Nachricht beim Intermediär abholen.

2. One-Way-Message, passiver Empfänger: Es erfolgt eine synchrone Zustellung, d.h. der Empfänger holt die Nachricht nicht aus einem Postkorb beim Intermediär ab, sondern erhält diese direkt vom Intermediär als Weiterleitung. Sobald die Nachricht an den Empfänger komplett übermittelt wurde, wird dies vom Intermediär registriert und der ursprüngliche Sender erhält von diesem eine Bestätigungsnachricht.
3. Request-Response, passiver Empfänger: Es erfolgt eine synchrone Zustellung, d.h. der Intermediär leitet die Nachricht vom Sender an den Empfänger direkt weiter. Der Empfänger muss nach dem Erhalt zeitnah reagieren und eine inhaltliche Nachricht als Antwort senden. Diese wird schließlich wieder vom Intermediär an den ursprünglichen Sender weitergeleitet.

Als Quittierungsmechanismus wird durch den Intermediär ein Laufzettel angelegt, auf dem u.a. Zeitpunkte der einzelnen Nachrichteneingänge beim Intermediär protokolliert werden. Für die komplette Datenstruktur des Laufzettels, siehe [OSCISpec].

OSCI ermöglicht für die einzelnen Szenarien auf Ebene der Geheimhaltung, Integrität, Authentizität und Nichtabstreitbarkeit die Verwendung verschiedener Sicherheitsstufen. Dies ist dahingegend sinnvoll, da geforderte Sicherheitsniveaus im richtigen Verhältnis zum eigentlichen Nutzen[9] stehen und deswegen konfigurierbar sein sollten. Für Details, welche Mechanismen für welche Stufen eingesetzt werden, siehe [OSCIPrinc, S.18ff.]. Die verwendeten Sicherheitsmechanismen orientieren sich dabei an den Konzepten, die bereits in Kapitel 3 erläutert wurden.

9 So sollte für einen Austausch von personengebundenen Gesundheitsdaten mehr Wert auf eine hohe Sicherheit gelegt werden als für ein Versenden von reinen Informationsdienstleistungen, die jedem zugänglich sein können.

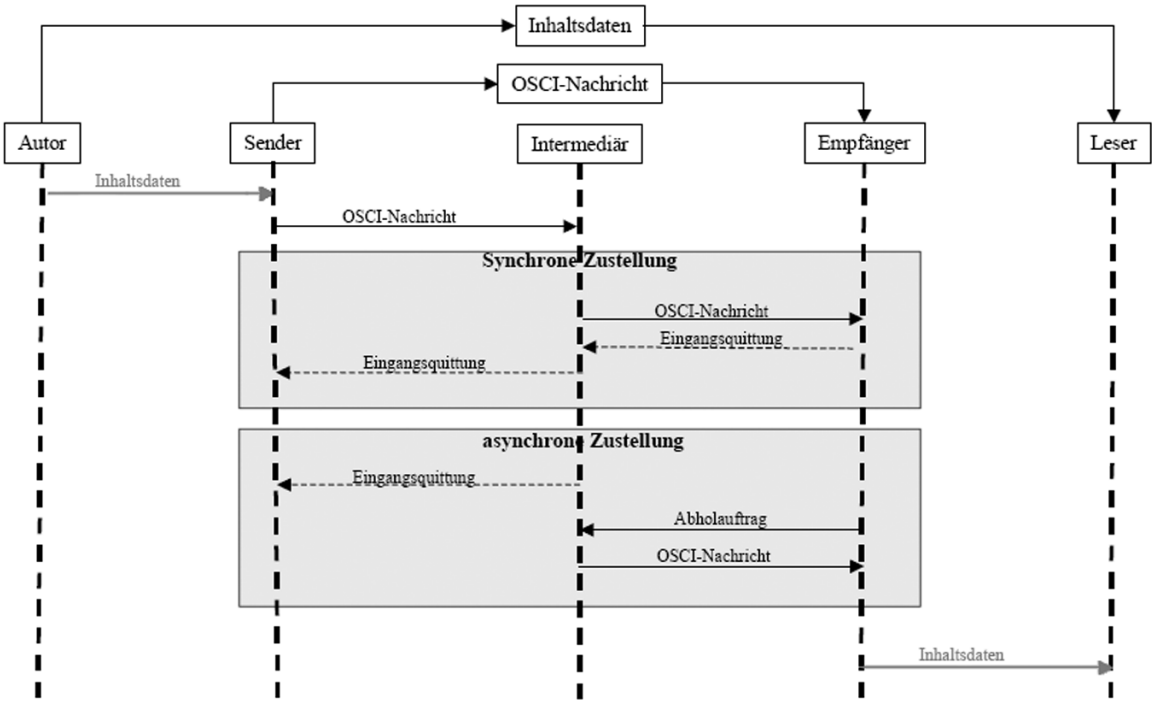


Abbildung 8: Übersicht einer OSCI Kommunikation als Sequenzdiagramm. Autor und Leser kommunizieren auf Geschäftsvorfallsebene, Sender, Intermediär und Empfänger auf Auftragsebene (Abb. aus [OSCIPrinc,S.8]).

## 4.4 Eingesetzte Technologien

OSCI verwendet als Standard folgende Technologien.

### 4.4.1 XML

XML ist ein vom W3C (World Wide Web Consortium) entworfener und gepflegter Standard, der ein Datenformat für strukturierte Daten spezifiziert [XML]. Da XML selber nur ein Format darstellt, in dem Daten in Tags in einer Baumstruktur organisiert sind, ist XML zunächst anwendungsneutral. Erst durch eine implizite[10] Interpretation eines XML Dokumentes oder anderen an XML angelehnten Standards erhält ein Dokument eine weitere Semantik. Bei OSCI werden folgende XML Standards verwendet[11]:

- XSD: XSD ist ein XML Format welches die Struktur für konkrete XML Dokumente festlegt und kann demzufolge als Metasprache angesehen werden. Alle XML

10 Implizit meint, dass ein Dokument im XML Format allein durch die Struktur und ohne Hilfe weiterer Standards selbst interpretiert werden kann, bspw. kann die Struktur eines Baumes eine einfache Eltern Kind Beziehung ausdrücken. Allerdings ist eine implizite Interpretation nur möglich, wenn der Leser bspw. als Computerprogramm oder Mensch über die entsprechenden Informationen verfügt.

11 Für eine detaillierte Ausführung der einzelnen XML Standards siehe [XSD], [XSig], [XEnc], [XNames] und [SOAP].

Dokumente die auf einer XSD basieren sind dabei Instanzen eines XSD Dokumentes. Mit Hilfe von XSD kann eine Validierung vorgenommen werden, die aussagt, dass ein XML Dokument sich an Strukturvorgaben über die Wohlgeformtheit[12] hinaus hält. OSCI verwendet durchgehend XSD, um die XML Struktur von OSCI Nachrichten festzulegen. Für die Schema Details der einzelnen Nachrichtentypen siehe [OSCISpec, S.31ff.].

- XML Signature: XML Signature ermöglicht die Einbettung von Digitalen Signaturen in ein XML Dokument. XML Signature wird von OSCI verwendet, um Nutzungsdaten zu signieren.
- XML Encryption: XML Encryption spezifiziert die Verschlüsselung von XML Dokumenten. OSCI setzt XML Encryption für die Verschlüsselung der Nutzungsdaten ein.
- XML Namespaces: XML Namespaces erlaubt die Definitionen von Namensräumen innerhalb von XML Dokumenten. Dadurch können Namenskollisionen leichter verhindert werden.
- SOAP: SOAP definiert zunächst ein XML Format, dass in einem Umschlag einen Nachrichtenkopf (zu engl. Header) und Nachrichtenrumpf (zu engl. Body) kapselt. Bei OSCI liegen die Nutzungsdaten im Nachrichtenkopf, die Inhaltsdaten innerhalb eines Inhaltsdatencontainers im Nachrichtenrumpf bzw. bei einem XML Fremdformat als ein vom Nachrichtenrumpf referenzierter Anhang. Diese OSCI Struktur wird wiederum als Anhang an einen anderen SOAP Umschlag hinzugefügt (siehe [OSCIPrinc,S.11ff.]). Für die verteilte Kommunikation wird die SOAP Nachricht mit der gekapselten OSCI Nachricht schließlich über das Protokoll HTTP oder HTTPS abgesendet. SOAP ist ein offener Standard, wird von vielen Plattformen[13] unterstützt und gilt deswegen als interoperabel.

Allerdings ist zu erwähnen, dass der OSCI Standard in der Version 1.2 aus dem Jahre 2002 mittlerweile nicht mehr auf dem neuesten Stand der Technik ist. So existieren bezogen auf die Verschlüsselung und der Digitalen Signatur im Umfeld von SOAP und Webservices durch den WS-Security Standard viele Neuerungen[14].

12 Die Wohlgeformtheit eines XML Dokumentes sagt lediglich aus, dass das Dokument die einfachen Richtlinien von XML befolgt (z.B. keine offenen Tags, korrekt geschachtelte Elemente etc.).

13 SOAP wird neben den großen Plattformen .Net und Java auch von vielen Skriptsprachen wie Ruby oder Python unterstützt.

14 Für entsprechende Standards im Umfeld von Webservices und SOAP siehe [Erl].

### 4.4.2 Kryptografiestandards

**Digitale Signatur**

OSCI unterstützt für die Hashwertbildung die Algorithmen SHA-1 (siehe S.21) und RIPEMD-160. Für die Digitale Signatur wird der Algorithmus RSA (siehe S.30) in Verbindung mit SHA-1 oder RIPEMD-160 verwendet. Die Modullänge des Schlüsselpaares muss mindestens 1024 Bit betragen [OSCISpec,S.18].

**Verschlüsselung/Entschlüsselung**

Die Ver- und Entschlüsselung wird mit einem hybriden Kryptosystem durchgeführt (siehe S.34). Die Algorithmen der entsprechenden symmetrischen Ver- und Entschlüsselung sind Two-Key-Triple-DES und AES mit den Schlüssellängen 128, 192 und 256 Bit. Der asymmetrische Algorithmus ist RSA mit einer Modullänge von mindestens 1024 Bit [OSCISpec,S.20].

## 4.5 OSCI Fazit

Der OSCI Standard folgt durch die unterstützten Sicherheitsstufen auf Ebene der Integrität, Vertraulichkeit, Nachweisbarkeit, und Authentizität den gesetzlichen Anforderungen der elektronischen Signatur und dem Datenschutz (siehe S.8 und S.12). So ist es möglich, elektronische Dokumente in einen Inhaltscontainer zu integrieren und diesen mit einer qualfizierten Signatur zu signieren. Außerdem wird bei OSCI ähnlich zum Datenschutz eine Trennung in Inhaltsdaten und transportrelevante Nutzungsdaten vorgenommen. Allerdings werden die technischen Möglichkeiten von OSCI durch rechtliche Rahmenbedingungen eingegrenzt: Da qualifizierte Signaturen auf natürliche Personen begrenzt sind (siehe S.11), lässt sich seitens der Intermediärs keine qualifizierte Signatur ausführen. Die gleiche Einschränkung liegt vor, wenn es sich bei Sender und Empfänger um eine Gruppe von Personen bspw. in Form einer Behörde handelt.

Durch die Erweiterung des Rollenmodells können außerdem Mehrwertdienste an zentraler Stelle durchgeführt werden. So sind Validierungen der Nachrichtenstruktur oder Zeritfikatsüberprüfungen durch den Intermediär möglich und es können früher Fehler auf Seiten des Senders und Empfängers abgefangen werden. Außerdem übernimmt der Intermediär mit dem Laufzettel Protokollierungs- und Quittierungsmechanismen, wodurch die eingesetzten und eingehaltenen Sicherheitsstandards transparent für die

Kommunikationsteilnehmer nachvollzogen und ggf. bei rechtlichen Streitigkeiten zu Rate gezogen werden können. Ein weiterer Vorteil ist, dass die OSCI Integrationsmodule auf Sender- und Empfängerseite die Mechanismen der Mehrwertdienste nicht implementieren brauchen. Auf diese Weise werden erneut Aufwände für entsprechende Softwarelösungen durch vereinfachte Implentierungsanforderungen minimiert. Auch ermöglichen die verschiedenen synchronen und asynchronen Szenarien eine gute Integration in die Fachverfahren einer Behörde.

OSCI als Standard stellt gute Lösungsansätze für die infrastrukturellen Herausforderungen eines sicheren Dokumentenaustauschs im E-Government zur Verfügung. Es wird ein einheitliches Nachrichtenformat für einen sicheren Dokumentenaustausch spezifiziert, welches über das interoperable SOAP in verteilten Systemen übertragen werden kann. Die freie Wahl des Format der Inhaltsdaten führt dazu, dass eine Kommunikation unabhängig von propriertären Applikationen und deren Formaten möglich ist. Zudem setzt das Nachrichtenformat auf offene XML Standards, so dass Softwareanbieter ohne große Barrieren entsprechende OSCI konforme Produkte implementieren können. Allerdings zeigt sich auch, dass durch vielfältige neuere Standards im Webservice Umfeld Teile von OSCI 1.2 veraltet sind. Eine Erweiterung von OSCI u.a. mit einer Integration von WS-Security steht deshalb für die Version OSCI 2.0 an [OSCI2.0Prev].

# 5 Dokumente und Adobe Acrobat

Da OSCI Transport keine Vorgaben hinsichtlich des Formats der Inhaltsdaten macht, können hier beliebige Dokumententypen verwendet werden können. So können bspw. eine E-Mail, ein ausgefülltes Webformular oder die gängigen Text-Prozessor Formate (`.rtf`,`.doc`,`.odt` etc.) innerhalb eines Dokumentenaustausches als Inhaltsdaten zum Einsatz kommen. Auch die domänenspezifischen XML Strukturen aus der OSCI B Spezifikation wären einer Option. Eine weitere interessante Alternative existiert außerdem mit PDF (Portable Document Format).

## 5.1 PDF Dokument

PDF ist ein weit verbreitetes Dokumentenformat, das von Adobe Systems[1] entwickelt wurde und aktuell in der Version 1.6 vorliegt. Das Format ist zudem offengelegt und als Branchenstandard anerkannt. Durch die Offenlegung existieren auch viele Applikationen[2] und Bibliotheken[3], die PDF Dokumente beabeiten und erzeugen können.

### 5.1.1 Sicherungsmechanismen

Ein weitere vorteilhafte Eigenschaft von PDF ist, dass intern Digitale Signaturen [PDFSpec, S.684ff.] und Verschlüsselungsinformationen [PDFSpec, S.91ff.] strukturiert sind. Durch die Digitale Signatur lässt sich das PDF Dokument authentifizieren und eine Fälschung des Inhalts nachweisen (siehe S.36). Die Verschlüsselung sichert dagegen Dokumentrechte bezogen auf Drucken, Editieren, Öffnen, etc.[4] ab. Die Verschlüsselung kann dabei auf zwei Weisen erfolgen:

1. Kennwortbasiert: Es existiert ein Kennwort, mit dem die einzelnen Rechte verschlüsselt werden. Beim Öffnen des Dokumentes wird das Kennwort wieder abgefragt und die Rechte werden entsprechend zur Verfügung gestellt. Das Kennwort ist dabei frei wählbar und ist keiner Instanz direkt zugeordnet.
2. Zertifikatbasiert: Der Ver- und Entschlüsselungsmechanismus basiert auf einen geheimen und öffentlichen Schlüssel. Der öffentliche Schlüssel wird dabei mit

1 Siehe www.adobe.com/
2 Als Beispiel siehe PDF Creator (www.pdfforge.org/) oder PDF Viewer (www.uni-koblenz.de/~droege/PDFViewer/; Zugriff: 25.12.2006).
3 Für eine Übersicht der PDF APIs für die Java Plattform, siehe schmidt.devlib.org/java/libraries-pdf.html; Zugriff: 25.12.2006.
4 Für eine vollständige Auflistung der einzelnen PDF Dokumentrechte, siehe [Pe/Ma/Ko, S.355].

einem Zertifikat authentifiziert (siehe S.33). Demzufolge lassen sich die Dokumentrechte nur durch die Instanz freischalten, die dem Zertifikat zugeordnet ist und Zugriff zum geheimen Schlüssel besitzt.

### 5.1.2 Formularunterstützung

PDF bietet eine Formularunterstützung an, mit der sich die bekannten Formularelemente (Textfelder, Radiobuttons etc.) in ein Dokument integrieren lassen [PDFSpec, S.634ff.]. Desweiteren lässt sich hier auch im Zusammenhang mit der Digitalen Signatur ein Unterschriftsfeld einbauen. Formularwerte können einer Validierung unterzogen werden und es lassen sich daraufhin verschiedene Aktionen[5] initiieren.

## 5.2 Adobe Acrobat

Bei Acrobat[6] handelt es sich um eine Produktfamilie von Adobe Systems, mit der die vielseitigen Möglichkeiten von PDF (siehe oben) aus einer Applikation heraus nutzbar sind. So können mit Acrobat u.a. Formulare erstellt [Pe/Ma/Ko, S.253ff.] und die Sicherheitsmechanismen der dokumentbasierten Rechtevergabe und der Digitalen Signatur angewandt werden [Pe/Ma/Ko, S.355ff.]. Die durch die zertifikatbasierte Verschlüsselung realisierte Rechtevergabe und die Digitale Signatur basieren dabei standardmäßig auf fortgeschrittenen Zertifikaten.

Acrobat ermöglicht nicht nur die Ausnutzung der Möglichkeiten von PDF, sondern bietet weitere Funktionen an. So lassen sich Dokumente in Fremdformaten nach PDF konvertieren und einzelne Arbeitsschritte zu einem Workflow zusammenfassen und automatisieren [Pe/Ma/Ko, S.644ff.]. Außerdem existiert durch die Stapelverarbeitung die Option, Arbeitsschritte auf eine Menge von Dokumenten automatisiert anzuwenden.

Acrobat lässt sich auch durch Drittanbieter durch das Acrobat SDK (Software Development Kit[7]) erweitern und es lassen sich dadurch neben den standardmäßig enthalten Funktionen weitere integrieren.

### 5.2.1 OPENLiMiT SignCubes Plug-In

Da die standardmäßige Digitale Signatur Funktion von Acrobat auf fortgeschrittenen Zertifikaten beruht, folgt sie nicht den Anforderungen der elektronischen Form (siehe

5 Ähnlich zu den HTML Formularaktionen Submit Form, Reset usw..
6 Siehe www.adobe.com/de/products/acrobat/; Zugriff: 20.12.06. Untersuchte Version ist Acrobat 7.
7 Für eine ausführlichere Betrachtung des Acrobat SDK siehe S.57.

S.8) und kann so im Verwaltungsverfahren nicht eingesetzt werden. Allerdings vertreibt OPENLiMiT ein Acrobat Plug-In[8], das eine Digitale Signatur auf PDF Dokumentenebene erlaubt und ebenfalls eine spätere Überprüfung dieser möglich macht. Die Lösung folgt den geforderten Sicherheitsstandards des Signaturgesetzes und ist seitens des BSI zertifiziert[9], so dass eine rechtskonforme Digitale Signatur im Zusammenhang von Acrobat und PDF möglich ist.

8 OPENLiMiT SignCubes 2.0, siehe www.signcubes.com/
9 Siehe www.qualisicard.de/blog/2005/12/openlimit-signcubes-software.html; Zugriff: 25.12.06

# 6 Integration OSCI in Adobe Acrobat

Rechtliche Rahmenbedingungen legitimieren eine elektronische Abwicklung im E-Government und mit OSCI existiert ein Standard, der diese rechtlichen Vorgaben mit einer technischen Lösung verbindet und die Komplexität der Sicherheitsmechanismen beherrschbarer macht. Außerdem existiert für Adobe Acrobat, das die vielseitigen Funktionen von PDF nutzbar macht, mit OPENLiMiT SignCubes eine Lösung, mit der PDF Dokumente innerhalb von Acrobat signaturgesetzkonform mit einer qualifizierten Signatur versehen werden können. Dadurch kann PDF die Forderungen der elektronischen Form erfüllen und ist dadurch gleichwertig zur Schriftform.

Allerdings existiert bisher keine Lösung einer Integration zwischen OSCI und Acrobat, die das letzte fehlende Stück in einem rechtskonformen sicheren elektronischen Dokumentenaustausch im E-Governemt mit Acrobat, PDF und OSCI darstellt. Ziel des praktischen Teils der Arbeit ist es, ein solche Integrationslösung prototypisch zu entwickeln und dadurch eine weitere sinnvolle Beispielanwendung für OSCI zur Verfügung zu stellen. Diese soll später im Rahmen des eGovernment Labors von Fraunhofer FOKUS[1] evaluiert werden.

## 6.1 Realisierungsumfang

Innerhalb der Acrobat Applikation[2] soll ein OSCI Client aufgerufen werden können, der das aktuell geöffnete Dokument als Inhaltsdaten in einen Inhaltsdatencontainer der OSCI Nachricht integriert und an den Intermediär sendet. Darüberhinaus existiert ein Nachrichtentext, der ebenfalls Teil der Inhaltsdaten ist. Die Datei wird dabei als referenziertes Attachement und der Nachrichtentext als XML Datum in einen Content Container der OSCI Nachricht (für Content Objekte siehe [OSCILib,S.21]) eingefügt. Darüberhinaus soll der Nachricht ein Betreff übergeben werden. Insgesamt entstehen Daten, die vergleichbar strukturiert sind wie eine E-Mail (Betreffzeile, Nachrichtentext und Attachement).

Umgesetzt wird die Senderseite des OSCI Szenarios One-Way-Message, aktiver Empfänger (siehe S.44). Die Nutzungsdaten werden verschlüsselt und mit einer fortgeschrittenen Signatur (siehe S.10) signiert. Für die Schlüssel bei den Signatur und Ver- und Entschlüsselungsvorgängen der Nutzungsdaten werden Softwarezertifikate ver-

1 Siehe www.fokus.fraunhofer.de/egov-lab/;Zugriff:02.11.06
2 Verwendete Version ist Adobe Acrobat Professional 7.0.7 für die Windows Plattform.

wendet. Der Aufruf des OSCI Clients erfolgt über einen Menüleisteneintrag oder einem Icon in der Toolbar.

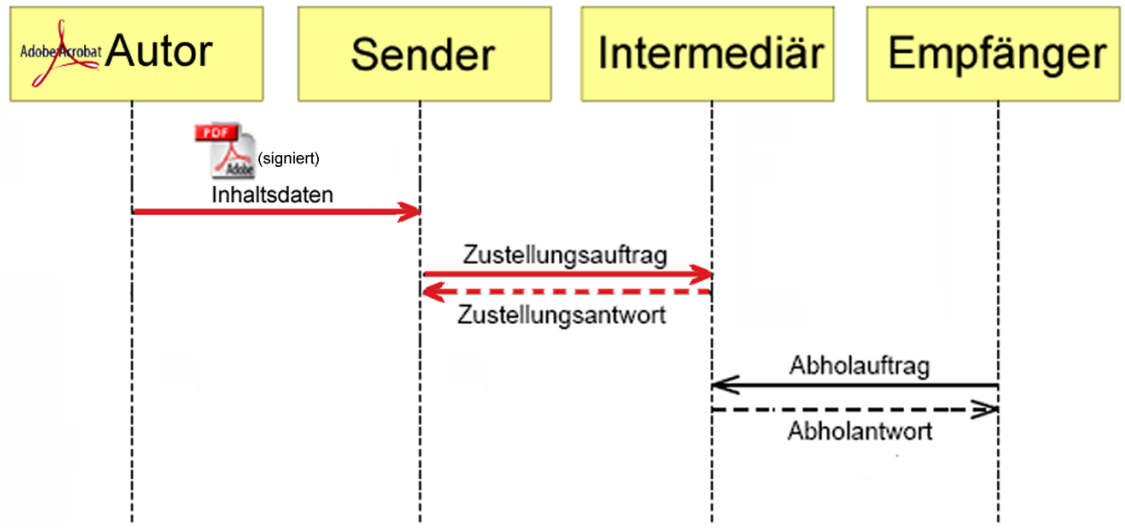


Abbildung 9: Szenario One-Way-Message, aktiver Empfänger. Die in der Grafik rot markierten Kommunikationswege sind Fokus der Implementierung. Die Funktionalität des Intermediärs ist im Implementierungsumfang ausgenommen, dementsprechend wird ein lauffähiger und erreichbarer Intermediär in der Integrationslösung vorrausgesetzt. (Abb. modifiziert aus: [AcrobatSpec,S.14]).

### 6.1.1 Konfiguration OSCI Transport

Für das OSCI Szenario müssen hinsichtlich Intermediäradressierung und Softwarezertifikaten weitere Einstellungen vorgenommen werden. Softwarezertifkate liegen dabei als `.p12`[3] und `.cer`[4] Dateiformate vor.

| Konfigurationstyp | Beschreibung |
|---|---|
| Intermediär URL | URL, über die der Intermediär adressiert und an die die OSCI Nachricht gesendet wird. |
| Intermediärzertifikat für Verschlüsselung | Softwarezertifikat (`.cer`), mit dessen öffentlichen Schlüssel die Nutzungsdaten für den Intermediär verschlüsselt werden. |
| Senderzertifikat für Signatur | Softwarezertifikat (`.p12`), mit dessen zugehörigen geheimen Schlüssel die Nutzungsdaten signiert werden. |
| Senderzertifikat für Entschlüsselung | Softwarezertifikat (`.p12`), mit dessen zugehörigen geheimen Schlüssel Nutzungsdaten entschlüsselt werden[5]. |
| Empfängerzertifikat | Softwarezertifikat (`.cer`), mit dem der Postkorb des Empfängers auf Seiten des Intermediärs identifiziert wird. |

3 Das `.p12` Format, ist mit dem mit den kryptografischen Standard der asymmetrischen Verschlüsselung PKCS#12 verknüpft. PKCS#12 definiert ein Format beschrieben wird, das das Zertifkat mit dem geheimen Schlüssel passwortgeschützt speichert. Für detailliertere Informationen siehe www.rsasecurity.com/rsalabs/node.asp?id=2124;Zugriff 11.11.06.

4 Das `.cer` Format folgt dem X.509 Standard (siehe [IETF X.509]) und ermöglicht den Zugriff auf den öffentlichen Schlüssel.

5 Es wird vorausgesetzt, dass der entsprechende öffentliche Schlüssel zuvor für die Verschlüsselung verwendet wurde.

Das Softwarezertifikat, das verwendet wird, um die Signatur des Intermediärs zu überprüfen, muss nicht eingestellt werden, da es in der Antwortnachricht des Intermediärs bereits enthalten ist.

Der Zugriff auf die geheimen Schlüssel in den Senderzertifikaten wird über eine PIN realisiert, die ebenfalls vom Nutzer eingegeben werden muss.

## 6.2 Vorgehen

Für die Entwicklung der OSCI Acrobat Integrationslösung wurden folgende Schritte vorgenommen:

1. Identifizierung der nicht-funktionalen Anforderungen, die die Lösung zusätzlich zu den im Realisierungsumfang beschriebenen funktionalen Punkten beeinflusst.
2. Auflistung der technischen Rahmenbedingungen, die sich vor allem auf die Wahl der verwendeten Technologien für die Implementierung auswirken.
3. Evaluierung und Auswahl der Implementierungstechnologien anhand der technischen Rahmenbedingungen und der funktionalen und nicht-funktionalen Anforderungen.
4. Beschreibung des groben Lösungskonzeptes der Implementierung.
5. Dokumentation der Implementierungsergebnisse[6].

## 6.3 Nicht-funktionale Anforderungen

Folgende nicht-funktionale Anforderungen werden als wichtig angesehen:

1. Benutzerfreundlichkeit
2. Sicherheit
3. Deployment
4. Plattformunabhängigkeit
5. Wartbarkeit

Näheres zur Beschreibung und der Sicherstellung dieser nicht-funktionalen Punkte sie-

6 Auf eine Zitierung des Quellcodes wird verzichtet, damit der Text nicht aufgebläht wird. Es werden lediglich für die Lösung zentrale Paketnamen, Klassennamen oder Modulnamen erwähnt.

he folgende Abschnitte.

### 6.3.1 Benutzerfreundlichkeit

Folgende ergonomischen Anforderungen, die sich auch am Standard ISO 9241 orientieren, werden betrachtet:

| Ergonomische Anforderung | Beschreibung[7] |
|---|---|
| Fehlertoleranz | Die Einstellungen für den OSCI Transport werden so weit wie möglich validiert und eine Mitteilung bei fehlerhaften Daten erfolgt. |
| Erwartungskonformität | Die grafische Oberfläche der OSCI Senderapplikation orientiert sich an einer gewohnten Fensteraufteilung und bekannten Dialogabläufen aus anderen Programmen. |
| Aufgabenangemessenheit | Es werden keine unnötigen Einstellungen vorgenommen. Die Fenster bieten nur Einstellungen an, die im aktuellen Kontext nötig sind. |
| Lernförderlichkeit, Selbstbeschreibungsfähigkeit | Der OSCI Client ist auch bei Erstbenutzung leicht zu bedienen. Mögliche Optionen sind offensichtlich erreichbar und beschrieben. |
| Steuerbarkeit | Der Benutzer kann beeinflussen, welche Einstellungen aktuell vorgenommen werden. |

Das Ergonomiekriterium der Individualisierbarkeit entfällt, da die OSCI Szenarien die Einstellungen bereits fest vorgeben und deswegen eine Individualisierbarkeit durch ein Optionsmenü o.ä. keinen Sinn macht.

### 6.3.2 Sicherheit

OSCI soll zu einer Verbesserung der Kommunikationssicherheit führen. Daher darf die Acrobat OSCI Integration keinesfalls die Sicherheitsstandards aufweichen. So muss vor allem die PIN, die den Zugriff auf den geheimen Schlüssel des asymmetrischen Schlüsselpaares des Senderzertifikats ermöglicht, geschützt werden.

### 6.3.3 Deployment

Der OSCI Client soll ohne komplexe manuelle Einstellungen (bspw. Setzen von Umgebungsvariablen, Kopieren oder Runterladen von Programmbibliotheken etc.) in Acrobat lauffähig sein. Es sollen zudem minimale Umgebungsanforderungen gestellt werden,

7 Die Beschreibungen sollen keine vollständige Wiedergabe der ISO 9241 Norm darstellen, sondern einen Überblick geben, was für die Acrobat OSCI Integrationslösung wichtig ist.

damit der OSCI Client nicht erst nach weiteren Installationsroutinen entsprechender Plattformen ausführbar ist.

### 6.3.4 Plattformunabhängigkeit

Obwohl die Acrbot OSCI Integration für die Windows Version von Acrobat entwickelt wird, sollen dennoch Maßnahmen getroffen werden, eine Portierung auf ein anderes von Acrobat unterstütztes Betriebssystem zu erleichtern.

### 6.3.5 Wartbarkeit

Da eine Erweiterung des OSCI Clients durchaus denkbar ist (siehe Ausblick S.88), wird eine gute Struktur und Dokumentation auf Source Code Ebene gefordert. Auch das Lösungskonzept und die Implementierungsergebnisse sollen die verschiedene Aspekte der OSCI Acrobat Integrationslösung klar abtrennen und modularisieren.

## 6.4 Technische Rahmenbedingungen

### 6.4.1 OSCI Abstraktionsbibliothek

Die OSCI Leitstelle hat Abstraktionsbibliotheken zur Verfügung gestellt, um von den SOAP Nachrichten zu abstrahieren. So wird die Verwendung der OSCI Funktionalitäten aus Clientsicht wesentlich erleichtert: Die Nachrichten, die in XML gehalten sind und eine durchaus komplexe Struktur (siehe Nachrichtenspezifikation [OSCISpec,S.48ff.]) besitzen, müssen nicht explizit geparst, validiert und erstellt werden. Diese Abstraktionsbibliothek[8] ist derzeit für die Programmiersprachensysteme Java und .Net verfügbar. Die während der Diplomarbeit aktuelle und verwendete Version der API ist 1.2.2.

### 6.4.2 Acrobat SDK Technologie

Da die Acrobat Applikation nicht nur die typischen Basisfunktionalitäten bereitstellt sondern offen für Erweiterungen ist und dafür ein entsprechendes SDK (Software Development Kit) anbietet, kann die Applikation auch als Plattform für externe Lösungen angesehen werden. Die OSCI Senderapplikation soll dabei als eine solche Erweiterung im Acrobat integriert werden. Für Erweiterungsrealisierungen stehen dabei drei soge-

8 Zum Download bereitgestellt auf www1.osci.de/sixcms/detail.php?gsid=bremen02.c.1403.de; Zugriff: 01.10.2006

nannte Acrobat SDK Technologien zur Verfügung [AcrobatSDK]:

### Acrobat JavaScript

Acrobat JavaScript ist eine JavaScript Erweiterung von Acrobat. Acrobat bietet dazu spezielle JavaScript Objekte an, durch die mit der Applikation oder auf PDF Dokumenten Ebene kommuniziert werden kann. Acrobats JavaScript Objekte sind dabei für Zugriffe auf die einzelnen Aufgabenbereiche eingeteilt (z.B. GUI Applikationszugriff, Dokumentenzugriff, Utility-Funktionen etc.).

Die Skripte besitzen verschiedene Kontexte, so sind sie direkt in der Applikation (folder level), für einzelne Dokumente (document level), für einzelne Komponenten innerhalb eines Dokumentes (field level) oder für eine Menge von Dokumenten (batch level) lauffähig [AcrobatSDK,S.20]. Acrobat JavaScript ist plattformunabhängig, dementsprechend müssen einmal implementierte Lösungen für die von Acrobat unterstützten Betriebssysteme Unix, MacOS und Windows nicht angepasst werden. Da es sich bei JavaScript um eine interpretierte Skriptsprache handelt, braucht sie grundsätzlich nicht kompiliert zu werden sondern ist in einer JavaScript Umgebung direkt ausführbar.

### Plug-Ins

Plug-Ins sind ANSI C/C++ Programmmodule, die als dynamische Bibliotheken existieren und die auf Acrobats API zugreifen. Die C/C++ Plug-Ins sind plattformabhängig und müssen für die einzelnen Betriebssysteme angepasst und neu kompiliert werden. Beim C++ der Plug-Ins handelt es sich um eine unmanaged C++ Variante und kann deswegen nicht direkt als .Net Lösung laufen. Die Acrobat API ist mächtig und kann ähnlich kategorisiert wie Acrobat JavaScript über verschiedene Teilmodule der API auf Acrobat zugreifen.

### Interapplication Communication (IAC)

IAC [AcrobatIAC] ermöglicht eine applikationsübergreifende Kommunikation mit Acrobat, d.h. aus anderen Applikationen kann auf Funktionalitäten von Acrobat zugriffen werden. Dies wird ermöglicht über die Schnittstellen Apple Events für Macintosh und OLE und DDE[9] für Windows Plattformen [AcrobatIAC]. Für die Unix Plattform ist dagegen keine IAC Unterstützung dokumentiert. Mit IAC ist ein Zugriff auf Acrobat Basisfunktionalitäten, wie sie über die GUI erreichbar sind, möglich. Darüberhinaus können

9 Apple Events, OLE und DDE sind betriebssystemabhängige Protokolle, die eine Kommunikation zwischen verschiedenen Applikationen ermöglichen.

auch Plug-In Erweiterungen verwenden werden, soweit diese IAC explizit als Schnittstelle zur Verfügung stellen.

## 6.5 Auswahl der Implementierungstechnologie

Da die Entscheidung, welche Acrobat Technologie für die Umsetzung des OSCI Clients verwendet wird, die weitreichendsten Folgen für die weitere Arbeit der Implementierung haben, sollen die Technologien nach maßgeblichen Kriterien evaluiert werden.

Acrobats IAC sieht vor, schon bestehende Funktionalität von Acrobat oder Plug-Ins für andere Applikationen nach außen anzubieten. Dagegen ist mit IAC ist kein Eingriff direkt in der GUI von Acrobat möglich, was allerdings im Realisierungsumfang gefordert ist (siehe S.53). Da die IAC Technologie deswegen grundsätzlich nicht in Frage kommt, wird sie in der Evaluierung nicht mitbetrachtet. Dadurch beschränkt sich die Untersuchung auf die Acrobat JavaScript und die Plug-In Technologien.

### 6.5.1 Evaluierungskriterien

Folgende Kriterien wurden identifiziert:

- .Net bzw. Java Unterstützung: Existiert eine Möglichkeit, Routinen in Java bzw. .Net aufzurufen? Diese Forderung stellt ein KO-Kriterium dar, da die OSCI Client Bibliotheken nur in Java und .Net verfügbar sind.
- Zugriffsmöglichkeiten auf PDF Dokument, Toolbar und Menü: Ist ein Zugriff auf die GUI und auf PDF Dokumente in der Acrobat Applikation möglich? Dies stellt ein KO Kriterium dar, da dies im Realisierungsumfang verlangt wird.
- Testbarkeit, Debugfähigkeit: Welche Möglichkeiten existieren, die Implementierung zu testen bzw. zu debuggen? Dieses Kriterium spielt insofern eine Rolle, da die Implementierungszeit bei guter Testbarkeit und Fehlerbehebungsunterstützung positiv beeinflusst wird. Auch die Wartbarkeit wird durch eine bessere Testbarkeit verbessert.
- Entwicklungszyklen: Welche Schritte müssen vorgenommen werden, um die Implementierung im Zusammenhang mit Acrobat lauffähig zu bekommen? Ein unkomplizierter Entwicklungszyklus wirkt sich positiv auf die Implementierungszeiten und auf die Wartbarkeit aus.

### 6.5.2 Evaluierungsergebnisse[10]

Folgende Tabelle ist nach oben genannten Evaluierungskriterien geordnet:

| Kriterium | Acrobat JavaScript | Plug-In |
|---|---|---|
| .Net bzw. Java Unterstützung | Aus Acrobat JavaScript ist ein Aufruf einer externen Java oder .Net Methode nicht möglich[11][12]. Lediglich über den Remoting Mechanismus SOAP, das von JavaScript unterstützt wird, wären Java oder .Net Bibliotheken über Umwege aufrufbar. | Da als Implementierungstechnologie C/C++ zur Verfügung steht, ist ein Aufruf von .Net und Java Bibliotheken realisierbar. Dies kann auf verschiedene Weise geschehen:<br>Sowohl auf .Net als auch auf Java Seite kann eine Kommunikation über die von C++ unterstützte Komponententechnologie COM, realisiert werden[13]. Darüberhinaus existiert für Java das JNI (Java Native Interface), über das bidirektional eine Kommunikation zwischen C++ und Java möglich ist. |
| Zugriffsmöglichkeiten auf PDF Dokument, Toolbar und Menüeintrag | Acrobat JavaScript ist eine Erweiterung des Standards JavaScript 1.5. Acrobat stellt vielfältige JavaScript Bibliotheken zur Verfügung. Der Zugriff auf das aktuell geöffnete PDF Dokument und Menü- und Toolbareinträge in der GUI sind über spezielle JavaScript Objekte möglich. | Für Plug-Ins, die in ANSI C/C++ geschrieben werden, stellt Acrobat eine mächtige API zur Verfügung, mit der auf allen Abstraktionsebenen die Acrobat Applikation erweitert werden kann. Hier ist der Zugriff auf das aktuell geöffnete PDF Dokument möglich. |

10 Für die Evaluierungsergebnisse wurde [AcrobatSDK], [AcrobatPI] und [AcrobatJS] verwendet.

11 JavaScript ist zwar eine unterstützte .Net Sprache, allerdings lässt sich Acrobat JavaScript nicht in die .Net Infrastruktur integrieren. Nach Recherche im Internet, ist bei webapplikationsentfernten JavaScript .Net Aufrufen nie die Rede. Deswegen wird angenommen, dass ein solcher .Net Aufruf über Acrobat JavaScript nur mit einem Remoting Mechanismus möglich ist.

12 Ein Aufruf von JavaScript zu Java ist direkt nur über Java Applets möglich. Der Aufruf von Java Methoden, die nicht in Java Applets implementiert sind, ist nur indirekt über ein Remoting Mechanismus möglich.
(siehe swforum.sun.com/jive/thread.jspa?threadID=104751&messageID=356782; Zugriff: 07.11.06).

13 Java selber bietet keine COM Unterstützung an. Allerdings existieren Produkte wie JACOB (siehe danadler.com/jacob/;Zugriff: 07.11.06), die diese zur Verfügung stellen. .Net gestattet auch ohne externe Tools eine Interoperabilität mit COM (siehe msdn2.microsoft.com/en-us/library/ms809970.aspx; Zugriff: 07.11.06).

| Kriterium | Acrobat JavaScript | Plug-In |
|---|---|---|
| Testbarkeit, Debugfähigkeit | Die Implementierung lässt sich praktisch nur testen, wenn sie in der Applikation läuft. Applikationsunabhängige Tests sind praktisch nicht möglich, da zu viele Abhängigkeiten zu den verwendeten JavaScript Objekten besteht. Ein Mocking oder Stubbing[14] dieser JavaScript Objekte wäre sehr aufwändig.<br>Es existiert eine JavaScript Objekt `console`, mit der Ausgaben gemacht werden können. Desweiteren bietet Acrobat als sogenannte JavaScript Tools eine GUI Konsole und einen eigenen Debugger an. | Die Implementierung lässt sich praktisch nur testen, wenn sie in der Applikation läuft. Applikationsunabhängige Tests sind praktisch nicht möglich, da zu viele Abhängigkeiten zu den verwendeten C/C++ Makros der Acrobat API bestehen. Acrobat erlaubt das Testen und Debugging innerhalb der Applikation über Popus, Beeps oder ein Debug Fenster[15]. Auch können eigene Logs über stdout erstellt werden.<br>Über die `#define DEBUG 1` Macroanweisungen kann zudem ein Typcheck und eine angepassten Fehlerbehandlung aktiviert werden. |
| Entwicklungszyklen | 1) Implementierung über JavaScript Source Code.<br>2) Schließen der geöffneten Acrobat Applikation.<br>3) Kopieren JavaScript Source Code Dateien in `Javascripts/` Ordner des Acrobat Applika-tionsverzeichnisses.<br>4) Starten der Acrobat Applikation. | 1) Implementierung über C/C++ Source Code.<br>2) Kompilieren des Source Codes in eine dynamische Bibliothek, die von Acrobat geladen wird.<br>3) Schließen der geöffneten Acrobat Applikation.<br>4) Kopieren der dynamischen Bibliothek in den plug_ins/ Ordner des Acrobat Applikationsverzeichnisses.<br>5) Starten der Acrobat Applikation. |

### 6.5.3 Evaluierungsentscheidung

Grundsätzlich lassen sich die funktionalen Anforderungen des Zugriffs auf das PDF Dokument und in die GUI sowohl durch JavaScript als auch durch ein Plug-In realisieren. Allerdings ist bei Acrobat JavaScript die Kommunikation zu .Net und Java nur über den SOAP Umweg möglich, wodurch sich klare Nachteile hinsichtlich des Deployments und der Wartbarkeit ergeben. Bei Verwendung von SOAP müssten entsprechende Ser-

14 Durch Mocks und Stubs können Tests auf Objekte ausgeführt werden, ohne dass die eigentlichen Klassen bzw. Laufzeitobjekte zu denen eine Abhängigkeit besteht verwendet werden. Durch die Auflösung dieser Abhängigkeiten sind Modultests leichter zu realisieren.

15 Das Debugfenster, das Strings ausgeben kann, ist dabei selber als im SDK mitgeliefertes Plug-In (Name 'DebugWin') implementiert.

verdienste laufen bzw. gestartet werden. Dadurch würde die Komplexität der Lösung unnötig erhöht werden. So ist SOAP eigentlich für eine verteilte Kommunikation gedacht, die allerdings für die Acrobat OSCI Integration nicht vorgesehen ist. Aus Sicht des Plug-Ins ist der COM Ansatz aus Deploymentsicht ähnlich negativ zu bewerten, da hier für eine Kommunikation mit Java oder .Net ebenfalls Artefakte registriert werden müssten[16]. Hier bietet die Kommunikation zwischen dem Plug-In und Java über JNI eine einfachere Umsetzung an.

Demensprechend ist allein aus der Betrachtung des KO Kriteriums der .Net und Java Unterstützung gesamt gesehen die Verwendung der Plug-In Technologie und JNI am sinnvollsten, auch wenn dadurch im Vergleich zu Acrobat JavaScript Nachteile in der Entwicklungszeit, Testbarkeit und Debugfähigkeit vorliegen.

Da das Handling von Java über JNI Methoden[17] schwer lesbaren und sehr langen Code hinterlässt ist und die grafischen Bibliotheken von Java (SWT oder Swing) ähnlich mächtig zu denen der Acrobat API sind, wird die Logik des OSCI Senders als externe Java Applikation realisiert, die von Acrobat isoliert getestet werden kann. Ein weiterer Vorteil der Implementierung des OSCI Senders in Java ist die Plattformunabhängigkeit: Die OSCI Senderapplikation ist im Gegensatz zum Plug-In auch auf anderen Systemen ohne Anpassungen lauffähig[18] und lässt sich auch unabhängig von der Acrobat Applikation verwenden.

## 6.6 Lösungskonzept

### 6.6.1 Module der Acrobat OSCI Integrationslösung

Die gesamte Acrobat OSCI Lösung wurde in drei verschiedene Module ausgelagert, die voneinander lose gekoppelt sind. Dadurch können sie auch unabhängig voneinander getestet und implementiert werden. Allerdings ist zu beachten, dass das Acrobat Plug-In nur lauffähig ist, wenn die Installationsroutinen des Installationsprogramms erfolgreich ausgeführt wurden.

1. Implementierung des Acrobat Plug-Ins: Das Acrobat Plug-In stellt dem Nutzer einen Toolbar- und Menüeintrag zu Verfügung, die bei einem geöffneten Doku-

16 Damit COM auf .Net zugreifen kann, muss .Net die Assembly registrieren (siehe msdn2.microsoft.com/en-us/library/ms973802.aspx#callnetfrcom_topic2;Zugriff 11.11.06). Auch die COM Komponente auf der C/C++ Seite müsste registriert und installiert werden (siehe msdn2.microsoft.com/en-us/library/ms165432(VS.80).aspx;Zugriff 11.11.06).

17 Dazu gehören u.a. Instanziierung, Methodenaufruf.

18 Natürlich mit der Vorraussetzung, dass eine JRE für das entsprechende System existiert.

ment aktiv werden. Bei Auswahl einer dieser Einträge wird das Dokument lokal gespeichert und der OSCI Sender wird als Java Applikation gestartet. Das Plug-In setzt keine direkten OSCI Funktionalitäten um.

2. Implementierung der Java OSCI Senderapplikation: Diese Java Applikation bietet mit grafischer Oberfläche dem Nutzer die Möglichkeit, die OSCI Nachricht zu konfigurieren und abzusenden, bildet also die gesamte Logik des OSCI Senders mit dem Szenario One-Way-Message, aktiver Empfänger ab. Dabei wird als Teil der Inhaltsdaten das vom Plug-In lokal gespeicherte PDF Dokument verwendet. Nach dem Beenden der Senderapplikation durch Abbruch oder erfolgreicher Sendung wird das lokal gespeicherte Dokument gelöscht.
3. Implementierung des Installationsprogramms: Da das Plug-In und die OSCI Senderapplikation auf mehreren Rechnern verwendet und das Deployment erleichtert werden soll, wird ein Installationsprogramm zur Verfügung gestellt, das die Softwareartefakte in die entsprechenden Verzeichnisse kopiert und das System so konfiguriert, dass die Acrobat OSCI Integrationslösung lauffähig ist. Auf diese Weise entfällt die manuelle Konfiguration, um das Plug-In und den OSCI Sender für den Nutzer verfügbar zu machen.

Obige Beschreibungen geben lediglich eine groben Überblick über die Funktionalität der Module. die Implementierungsdetails werden in den weiteren Abschnitten der Implementierungsergebnisse beschrieben. Auf die nicht-funktionalen Punkte der gesamten Acrobat OSCI Integrationslösung wird im Bewertungsteil auf S.84 Stellung genommen.

## 6.7 Implementierungsergebnis Acrobat Plug-In

### 6.7.1 Übersicht der Teilaufgaben

Für das Acrobat Plug-In wurden folgende Teilaufgaben identifiziert und in der Implementierung in eigene C++ Module delegiert:

1. Registrierung als Plug-In in die Acrobat Applikation und Hinzufügen von Menü- und Toolbareinträgen.
2. Speichern des aktuell geöffneten Dokumentes als temporäre Datei, die als Teil der Inhaltsdaten beim OSCI Client verwendet wird.
3. Starten der JVM (Java Virtual Machine) und Aufruf des Java OSCI Clients.

**Entwicklungstools**

Folgende Entwicklungstools -bzw. umgebungen wurden für die Implementierung der einzelnen Module verwendet:

- **JNI:** Für die Einbettung des OSCI Java Clients wurde als Teil von JNI das Java Invocation Interface verwendet [Liang,S.83]. Die JNI API wird zur Kompilierungszeit über das inkludieren des Headers `jni.h` zur Verfügung gestellt. Zur Linkzeit muss bei der Erstellung der dynamischen Bibliothek beim Java Invocation Interface zudem ein Verweis auf die Bibliothek `jvm.lib` des Java Pakets existieren. Es wurde JNI des Java 1.4 SDK Pakets verwendet.
- **Visual C++ 2005:** Als Entwicklungssprache wurde Visual C++ 8 und als Entwicklungsumgebung Visual Studio 2005 verwendet. Acrobat Plug-Ins sind nicht plattformunabhängig. Da die gesamte Entwicklung der Acrobat OSCI Lösung auf Windows XP erfolgte, wurde Windows mit x86 Architektur als Zielplattform gewählt.

Für die Lauffähigkeit des Plug-Ins muss im Zusammenhang mit JNI folgende Voraussetzung gelten: Bei Start der Acrobat Applikation muss ein Verweis in der Umgebungsvariable `$PATH` vorhanden sein, der auf die dynamische Bibliothek `jvm.dll` zeigt. Zu beachten ist, dass `jvm.dll` nicht in ein beliebiges Verzeichnis kopiert werden darf, sondern im ursprünglichen Verzeichnis der Java Installation bleiben muss. Für das entsprechende Setzen der `$PATH` Variable im Installationsprogramm siehe S.83.

## 6.7.2 Umsetzung

Für die Umsetzung des Plug-Ins wurden folgende Plug-In API Schichten (siehe [AcrobatSDK, S.23]) verwendet:

- AV (Acrobat View) Schicht: Ermöglicht u.a. einen Eingriff in die GUI der Applikationsteil (Dialogboxen, Menüs, Toobar etc.).
- PD (Portable Document) Schicht: API Teil, der sich direkt auf die PDF Dokumente bezieht und Aktionen auf diese erlaubt.
- AS (Acrobat Support) Schicht: Bietet plattformunabhängige Utitlities an (String Handling, Dateisystemfunktionen, Callbacks etc.).

**Registrierung Plug-In (Modul `OSCIPluginInit.cpp`)**

Alle Plug-Ins müssen folgende Routine als Handshake implementieren, die von Acrobat bei Start aufgerufen wird:

```
ACCB1 ASBool ACCB2 PIHandshake(ASUns32 handshakeVersion, void *hsData)
```

In dieser Routine müssen der `*hsData` Struktur vier vom Plug-In implementierte Callbacks mit Hilfe der AS API übergeben werden, die Acrobat an passender Stelle im Lebenszyklus des Plug-Ins aufruft. Folgende Callbacks existieren:

- `ACCB1 ASBool ACCB2 PluginExportHFTs(void)`: An dieser Stelle werden Routinen als Host Function Tables (HFT)[19] vom implementierten Plug-In exportiert. Da das Plug-In primär für den Aufruf der OSCI Senderapplikation entwickelt wurde, sind HFTs nicht sinnvoll und werden dementsprechend auch nicht exportiert.
- `ACCB1 ASBool ACCB2 PluginImportReplaceAndRegister(void)`: Hier können HFTs importiert, HFTs von Acrobat durch eigene ersetzt und andere Callbacks für eine asynchrone Benachrichtigung seitens Acrobat übergeben werden. Für die Reallisierung der OSCI Lösung ist diese Funktionalität nicht erforderlich und wurde dementsprechend nicht genutzt.
- `ACCB1 ASBool ACCB2 PluginInit(void)`: Hier wird die Initialisierung des Plug-Ins vorgenommen. Im Plug-In wird ein Eintrag als Icon in die Toolbarleiste und ein Menüeintrag in die Acrobat Applikation hinzugefügt. Das Icon ist eine Bitmap Grafik und ist in der erstellten Plug-In Bibliothek als Resource enthalten. Der Menü- und Toolbareintrag wurden mit der AV API umgesetzt. Die auszuführende Aktion bei Betätigen der Einträge wurde per Callbackdefinition mit der AS API realisiert.
- `ACCB1 ASBool ACCB2 PluginUnload(void)`: Diese Routine wird aufgerufen, wenn das Plug-In seitens Acrobats beim Schließen der Applikation oder bei Fehlern innerhalb der anderen Handshake Callbacks entfernt wird. Innerhalb des Plug-Ins wird der Menü- und der Toolbareintrag mit der AV API entfernt.

19 Host Function Tables sind Routinen, die eine Kommunikation zwischen einem Plug-In und der Acrobat Applikation und anderen Plug-Ins erlauben (siehe [AcrobatPlug,S.24]).

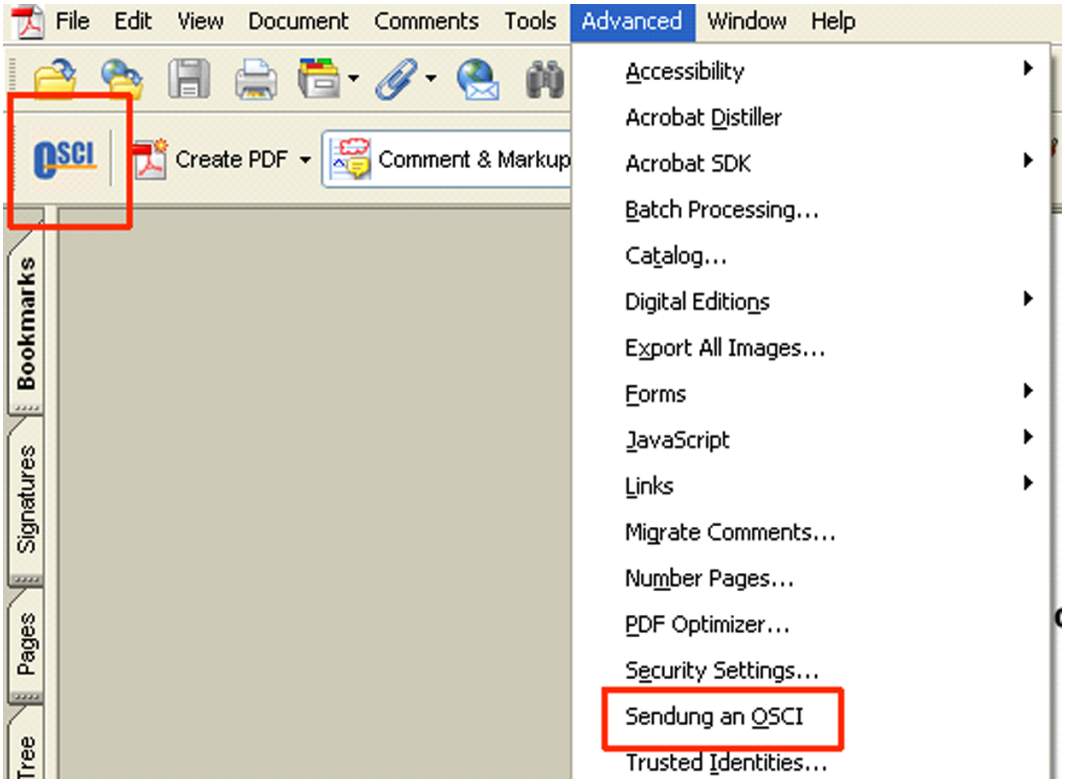

Abbildung 10: Der Senderclient wird über die Toolbar oder das Menü aufgerufen.

### Speichern Dokument als temporäre Datei (Modul `pdfTempFile.cpp`)

Die aktuell geöffnete Datei wird zunächst als temporäre Kopie gespeichert. Die Gründe dafür sind:

- Integrität der Inhaltsdaten: Für den Benutzer muss klar sein, dass das Dokument als Inhaltsdatum in der OSCI Sendung verwendet wird. Da der Client erst nach den OSCI Einstellungen durch den Nutzer das Dokument als Datei lädt und in die Inhaltsdaten der OSCI Nachricht integriert, ist es nicht garantiert, dass das aktuell geöffnete Dokument nicht verändert wurde.
- Konflikt bei verteilter Umgebung: Auch besteht die Gefahr, dass das aktuell geöffnete Dokument selbst nur temporär auf einer Netzwerkresource liegt und während der Arbeit mit der OSCI Senderapplikation durch einen anderen Nutzer oder durch Betriebssystemroutinen[20] gelöscht wird und nicht mehr zur Verfügung steht.

Da die Kopie als temporäre Datei interpretiert werden kann, muss sie nur so lange existieren, bis die OSCI Senderapplikation durch erfolgreiche Sendung oder Abbruch

20 Sogenannte Hintergrunddienste können bspw. so eingestellt sein, dass Inhalte des globalen TEMP Verzeichnisses in bestimmten Zeitabständen gelöscht werden.

beendet ist. Diese temporäre Kopie wird dabei im persönlichen temporären Verzeichnis gespeichert. Dieser ist bei gut konfigurierten Betriebssystemen eine Ressource, die exklusiv dem aktuell angemeldeten Nutzer zusteht und auf die, bis auf Systemadministratoren, niemand Zugriff erhält. Dieses temporäre Verzeichnis wird mit Hilfe der Umgebungsvariablen `$TEMP` ermittelt und als Zielverzeichnis für die Kopie des Dokumentes verwendet.

Diese 'Speichern unter' Funktion wurde mit der AV (Extrahieren des aktuell geöffneten Dokumentes), der PD (Speichern des Dokumentes) und der AS API (Erstellen Dateipfad temporäre Datei) realisiert.

### Starten JVM und OSCI Client (Modul `startJava.cpp`)

Für die Kommunikationsbrücke zwischen C++ im Plug-In und Java wurde JNI verwendet. Für den Start der Senderapplikation müssen folgende Informationen existieren:

1. Klassenpfad: Der Klassenpfad der Java Klasse, die eine `main` Methode als Eintrittspunkt des OSCI Java Client besitzt, muss bekannt sein. Da die Java OSCI Senderapplikation primär für Acrobat entwickelt wurde, befinden sich die Bibliothek der Java Senderapplikation innerhalb der Verzeichnisstruktur der Acrobat Installation. Da die Acrobat Installation selber nicht auf einen Dateipfad festgelegt ist[21], ist dementsprechend der Klassenpfad der OSCI Senderapplikation variabel. Eine Angabe eines relativen Klassenpfads ist ebenfalls nicht sinnvoll, da das Basisverzeichnis des relativen Pfades immer das Verzeichnis ist, in dem die Acrobat Applikation gestartet wurde[22]. Insofern würden die Java Bibliotheken nicht gefunden werden, wenn bspw. ein PDF Dokument einmal im Desktopverzeichnis und ein anderes Mal im Verzeichnis darunter geöffnet wird. Deshalb muss die Information des Klassenpfades absolut vorliegen.
2. Pfad temporäre Datei: Da der Dateipfad der Kopie des aktuell geöffneten PDF Dokumentes erst zur Laufzeit über die `$TEMP` Variable festgelegt wird, muss das Plug-In diese Information entsprechend an die OSCI Senderapplikation weiterleiten.

Der Klassenpfad kann ohne weitere persistente Einstellungen, bspw. über Registryein-

21 Erst zum Zeitpunkt der Installation von Acrobat wählt der Nutzer das Installationslaufwerk und Verzeichnis aus.

22 Der Start der Acrobat Applikation muss nicht explizit erfolgen sondern kann z.B. durch ein Doppelklick auf eine PDF Datei der Aufruf von Acrobat indirekt erfolgen. Dieser Mechanismus wird dadurch umgesetzt, dass das Fenstersystem des Betriebssystems Dateitypen mit Applikationen assoziiert.

träge, nicht ermittelt werden. Es existieren weder Umgebungsvariablen, die Auskunft über den Pfad der Acrobat Installation geben, noch ist ein Traversieren des Dateibaums aus C++ heraus sinnvoll, da die Suche nach der Acrobat Installation zu aufwendig wäre. So könnte besonders bei mehreren Laufwerken und vielen Verzeichnissen eine solche Suche zu langen und nicht akzeptablen Antwortzeiten führen.

Aus diesem Grunde wird eine andere Variante für den Start der OSCI Senderapplikation gewählt. Es wird über JNI eine vom Installationsprogramm (siehe S.78) gesetzte persistente Einstellung ausgelesen. Diese speichert den Aufruf der OSCI Senderapplikation als String. Diesem String wird zudem innerhalb des Plug-Ins der Pfad der temporären PDF Datei angehängt. Auf diese Weise erhält die Einstiegsmethode `public static void main(String[] args)` diesen als String Argument und kann diese Information verwenden. Diese persistenten Einstellungen werden über die Preferences API, die eine plattformunabhängige Sicht auf persistente Betriebsystemeinstellungen bietet[23], des Java Standard SDKs gesetzt bzw. ausgelesen.

Der Aufruf der Java OSCI Senderapplikation über JNI erfolgt zusammengefasst über folgende Schritte:

1. Initialisierung und Start der JVM.
2. Verwendung von `java.util.prefs.Preferences`, um den gespeicherten Java Aufruf als String auszulesen.
3. Verkettung des Java Aufrufs und dem Pfad der temporären Datei.
4. Da eine JVM initialisiert vorliegt, erfolgt nun der Start der Java OSCI Senderapplikation über `java.lang.Runtime.exec(String)`. String Argument ist der zusammengesetzte Aufruf aus Schritt 3).

## 6.8 Implementierungsergebnis Senderapplikation

### 6.8.1 Übersicht der Teilaufgaben

Für die OSCI Senderapplikation wurden verschiedene Teilaufgaben identifiziert:

1. Implementierung des Wizards und seiner Wizardseiten.
2. Implementierung der Fortschrittsanzeige.
3. Erstellung der Fensterinhalte in sogenannten Composites, die alle Controls kap-

23 Für Windows XP verwendet die Preferences API bspw. die Registry für persistente Einstellungen (siehe mindprod.com/jgloss/registry.html; Zugriff: 25.01.07).

seln, die dem Nutzer die Konfigurations- und Inhaltsdateneingaben ermöglichen.

4. Implementierung eigener Controls, die von der GUI Bibliothek nicht angeboten werden.
5. Implementierung der Dialoge, in denen der Nutzer Zertifikate auswählen kann.
6. Implementierung des Zugriffs auf im Nutzerkonto eingestellte Zertifikate.
7. Implementierung des OSCI Geschäftsvorfalls One-Way-Message, passiver Empfänger, mit Protokollierung.
8. Implementierung der Validierungen.
9. Implementierung der Integritätssicherstellung und Löschen der temporären Datei.

**Entwicklungstools**

Folgende Entwicklungstools -bzw. umgebungen wurden für die Implementierung der einzelnen Teilaufgaben verwendet:

- **Eclipse IDE:** Eclipse ist eine frei verfügbare Entwicklungsplattform, die hauptsächlich für Implementierungen von Java Applikationen verwendet wird. Die Wahl fiel auf Eclipse, da es automatisierte Refactorings ermöglicht, JUnit integriert ist, komfortable Editoren existieren und ein Plugin für die Versionsverwaltung Subversion (Subclipse) erhältlich ist.
- **SWT/JFace:** SWT (Standard Widget Toolkit) ist eine alternative Grafikbibliothek, die eine Alternative zu Swing darstellt. Es ist praxiserprobt[24] und es existieren bereits positive Erfahrungen. Als Erweiterung zu SWT bietet JFace weitere Vorteile (siehe [Sc/Ho/Ng/Mi]). Für den OSCI Client wurde JFace in erster Linie dafür verwendet, um den Wizard und die Fortschrittsanzeige zu implementieren.
- **JUnit:** JUnit ist ein Unittesttool für Java. Es wurde während der Implementierung für automatisierte Tests verwendet und war ebenfalls in Debuggingsessions[25] hilfreich.

24 Eclipse verwendet selber SWT als Grafikbibliothek.
25 Mit JUnit lassen sich komfortabel einzelne Methoden isoliert aufrufen, so dass eine Debuggingsession gezielt ausgeführt werden kann.

### 6.8.2 Umsetzung

Es werden folgend einzelne Klassen und Pakete erläutert, die für das Verständnis der Umsetzung relevant sind. Andere Klassen, die aufgrund von aufgrund der nicht-funktionalen Anforderungen Wartbarkeit und Testbarkeit eingeführt wurden, werden nicht erwähnt.

**Wizard**

Der Wizard[26] basiert auf einem Wizard Framework von JFace (siehe [Sc/Ho/Ng/Mi, S.234ff.]). Von JFace wird ein Standardwizard angeboten, der den üblichen Fensteraufbau und die Steuerungslogik eines Wizards implementiert. Der Wizard ist selbst ein Container für sogenannte Wizardseiten, in die der Nutzer seine Daten eingibt. Alle Wizardseiten sind in drei Bereiche aufgeteilt:

1. Beschreibungsteil: Dient der Information für den Nutzer, im welchen Kontext die einzugebenden Daten stehen. Hier steht ein entsprechender Text und eine kleine Grafik ist eingeblendet.
2. Datenteil: Zeigt die Dialogelemente, die der Nutzer für seine Eingabedaten wahrnimmt.
3. Steuerungsteil: Stellt die Schaltflächenleiste dar, mit der zwischen den Wizardseiten gesprungen („Zurück“, „Weiter“), die OSCI Nachricht abgesendet („Fertig stellen“) oder der Konfigurationsvorgang der OSCI Nachricht abgebrochen („Abbrechen“) werden kann.

Es müssen sowohl der Wizard als auch die Wizardseiten definiert werden, welches über die Ableitung entsprechender vorhandener Klassen geschieht. Folgende Aufgaben werden von diesen Erweiterungen vorgenommen:

- `org.fraunhofer.osci.gui.wizard.OSCIWizard`: Unterklasse von `org.eclipse.jface.wizard.Wizard` und Container für alle Wizardseiten, der diese instanziiert und deren Reihenfolge innerhalb der Steuerung definiert. Außerdem werden Callbacks für die Steuerungselemente „Abbrechen“ und „Fertig stellen“ definiert.
- Folgende Wizardseiten, die der Wizard instanziiert, wurden im OSCI Client implementiert. Diese referenzieren auf ein zugehöriges Composite (siehe S.73), das die Controls kapselt. Die Reihenfolge der Wizardseiten entspricht der fol-

26 Ein Wizard ist ein grafischer Dialog, mit dem man über mehrere Wizardseiten Einstellungen durchführt. Für den Wizard charakteristisch sind die Schaltflächen „Zurück“, „Weiter“, Fertig stellen“ und „Abbrechen“, mit denen der Wizard gesteuert werden kann.

genden Auflistung. Alle Wizardseiten leiten dabei `org.eclipse.jface.wizard.WizardPage` ab:

- `org.fraunhofer.osci.gui.wizard.IntermediaryPage`: Eingabfelder für Daten des Intermediärs (URL und Verschlüsselungszertifikat).
- `org.fraunhofer.osci.gui.wizard.OriginatorAddresseePage`: Eingabefelder für Daten des Senders (Signatur- und Entschlüsselungszertifikat, PIN) und Empfängers (Verschlüsselungszertifikat).
- `org.fraunhofer.osci.gui.wizard.MessagePage`: Eingabefelder für Daten Betreffzeile und Nachrichtentext.
- `org.fraunhofer.osci.gui.wizard.SummaryPage`: Keine Eingabefelder, sondern Übersicht aller eingegeben Werte (außer PIN für Senderzertifikate), um vor der OSCI Sendung noch einmal eine Überprüfung der Einstellungen zu ermöglichen.

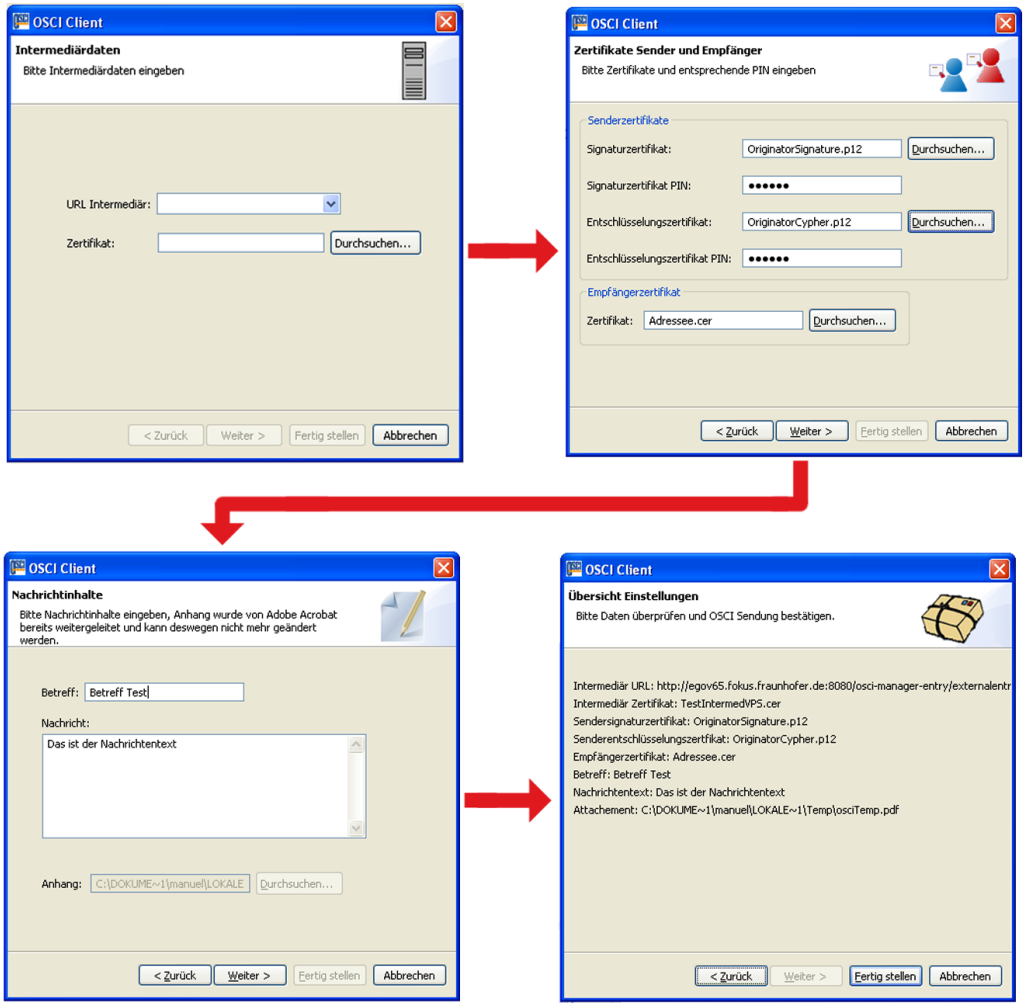


Abbildung 11: Der Wizard als OSCI Senderapplikation.

Die Aktivierung der Schaltflächen „Weiter“ und „Fertig stellen“ wird über die Anweisung `setPageComplete(boolean)` innerhalb der Wizardseitenklassen ausgeführt. Dabei besitzen die Composites der Wizardseiten Zugriff auf diese Methode und können so bei erfolgreicher oder fehlgeschlagener Validierung die nächste Seite bzw. das Abschließen des OSCI Clients freischalten oder sperren. Schließlich übergibt der Wizard bei Klick auf „Fertig stellen“, das erst auf der letzten Wizardseite aktiv ist, alle eingegeben Daten an das Fortschrittsanzeigefenster (siehe S.72).

**Fortschrittsanzeige**

Die Fortschrittsanzeige initiiert die OSCI Sendung und gibt Auskunft über den aktuellen Status während der Ausführung. Für die Fortschrittsanzeigenlösung existieren folgende Klassen:

- `org.fraunhofer.osci.gui.progress.ProgressWindow`: Stellt das Fortschrittsanzeigefenster dar. Es erwartet ein Objekt, welches die benötigten Daten für die Ausführung des Szenarios enthält. Zunächst wird die Integrität der temporären PDF Datei überprüft. Falls diese fehlschlägt, wird ein Fehlermeldungsfenster angezeigt und die Senderapplikation abgebrochen. Bei Erfolg der Integritätsüberprüfung wird die Übertragung angestoßen (siehe S.76) und die Daten werden übergeben. Wenn der Vorgang abgeschlossen ist, wird von der Fortschrittsanzeige ein Fenster geöffnet, das den Laufzettel visualisiert. Schließlich wird die temporäre Datei nach Beenden der Senderapplikation aus Gründen der Geheimhaltung gelöscht.
- `org.fraunhofer.osci.gui.progress.ProgressEventHandler`: Bietet Callbackmethoden an, die während der Ausführung des Übermittlung aufgerufen werden, dadurch den Status erhalten und die Fortschrittsanzeige aktualisieren.

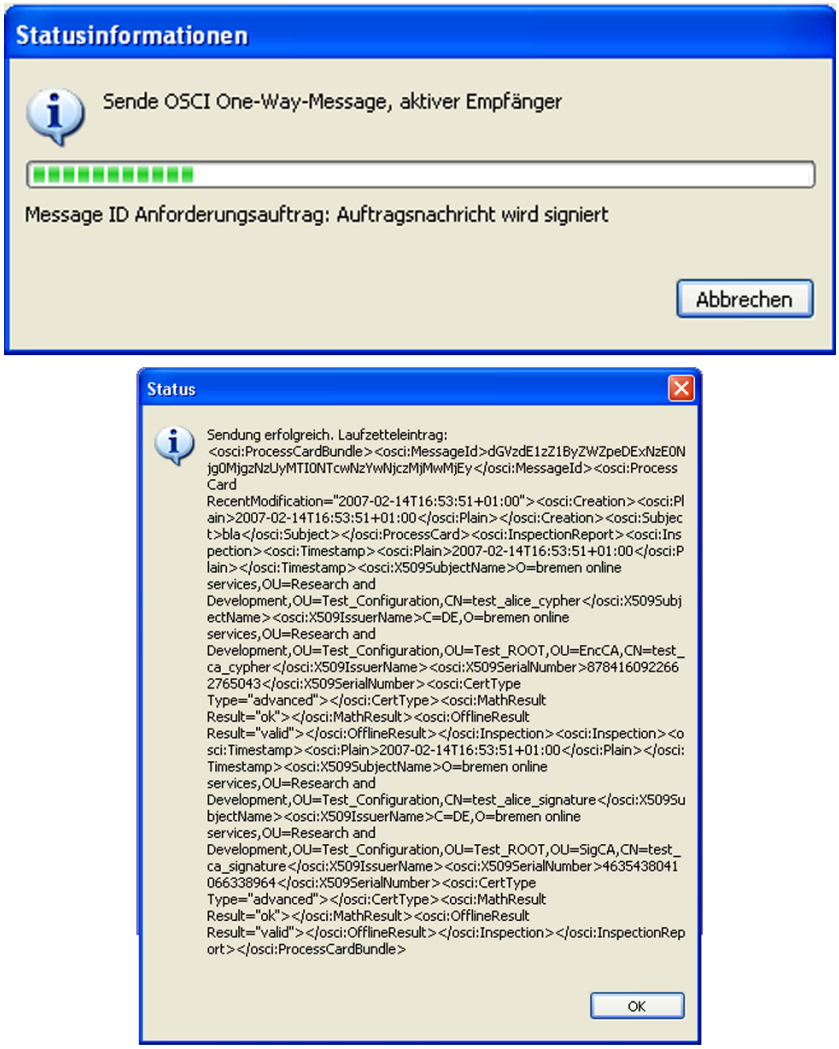


Abbildung 12: Fortschrittsanzeige für Statusinformation und Visualisierung Laufzettel.

## Wizard Composites

Die Wizard Composites kapseln alle Dialogelemente, die in den Wizardseiten vorkommen. Jeder Wizardseite ist dabei ein Composite zugeordnet. Die entsprechenden Composites sind im Pakt `org.fraunhofer.osci.gui.wizard.composites` zu finden[27].

## Zertifikatauswahldialog

Damit die Dateien für die Zertifkateinstellungen nicht immer langwierig über ein Dateiauswahldialog ausgewählt werden müssen, werden Zertifikate mit den Dateiendungen `.p12` und `.cer` als Tupel (Zertifikatname und Dateipfad) lokal abgespeichert (siehe unten). Der Nutzer führt diese Konfiguration der Zertifikatmenge und die Auswahl entspre-

27 Auf eine genaue Beschreibung der Dialogelemente wird verzichtet, da deren Bedeutung innerhalb der Wizardseite bereits erläutert wurde.

chender Zertifikate über einen eigenen Dialog aus. Dabei wird das Zertifikat aus einer Listbox ausgewählt. Außerdem kann ein Zertifikat aus der Listbox entfernt werden. Um ein Zertifikat hinzuzufügen, müssen in einem Unterdialog die Daten Zertifikatname und Zertifikatpfad eingegeben werden. Auf die Verwendung des Windows Zertifikatspeichers wurde verzichtet, um die Lösung auf andere Plattformen portabel zu halten. Dennoch ist eine Verwendung des Windows Zertifikatspeichers denkbar (siehe S.89).

Für den Zertifikatdialog existieren folgende Klassen:

- `org.fraunhofer.osci.dialogs.ChooseCertificate`: Dialog, in dem das Zertifikat ausgewählt wird. Zudem kann über die Schaltfläche „Hinzufügen“ ein neuer Dialog aufgerufen und über „Entfernen“ ein Zertifikat entfernt werden. Die entsprechende Schaltflächenleiste („Hinzufügen“, „Entfernen“) wird im Composite `org.fraunhofer.osci.dialogs.composites.AddRemoveFromCertificateListBut-tonBar` gekapselt.
- `org.fraunhofer.osci.dialogs.NewCertificate`: Stellt den Dialog dar, über den man die Daten (Zertifikatname und Dateipfad) des neuen Zertifkats für die Konfiguration eingibt.

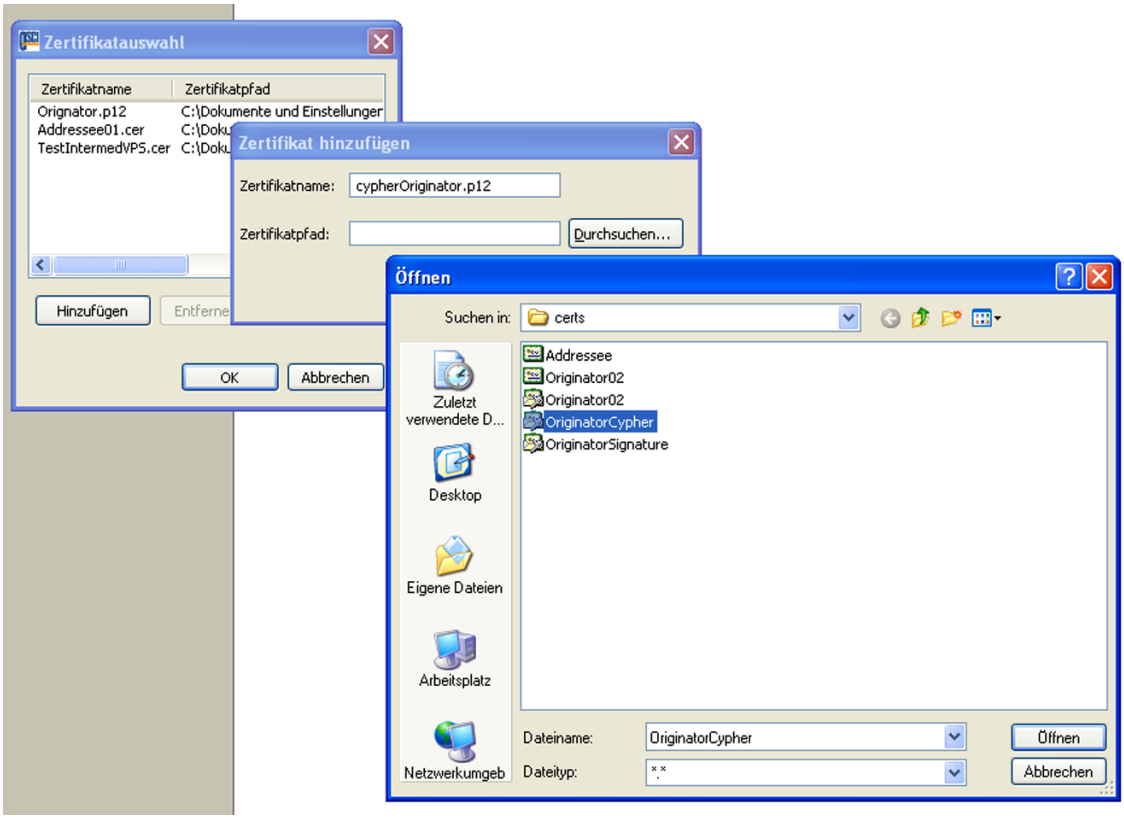


Abbildung 13: Zertifikatauswahldialog.

### Zugriff persistente Zertifikateinstellungen

Die Zertifikateinstellungen (siehe Zertifikatauswahldialog oben) werden innerhalb des Applikationsdatenverzeichnis (`$homeVerzeichnis/Anwendungsdaten/osciApp`) als XML Konfigurationsdatei gespeichert. XML wurde ausgewählt, da sich hier Daten gut strukturieren lassen und ein programmatischer Zugriff auf XML durch entsprechende Java Bibliotheken möglich ist. Dabei besteht ein Datensatz aus einem Zertifikatdatentupel (Name, Dateipfad).

Ein Zugriff auf die Konfigurationsdatei wird beim Zertifikatauswahldialog benötigt (Auslesen aller gespeicherten Zertifikatdatentupel, Hinzufügen und Löschen eines Zertifikatdatensatzes). Zusätzlich ist ein Zugriff nach der Bestätigung der OSCI Sendung nötig, wo Dateipfadinformationen mit Hilfe der Zertifikatnamen aus der Konfigurationsdatei ausgelesen werden, um Datenströme[28] der Zertifikatdateien zu erhalten.

Folgende Klassen sind für den Zugriff auf die Zertifikatseinstellungen zentral:

- `org.fraunhofer.osci.settings.xml.CertificateSettings`: Bietet nach außen die benötigten Zugriffsfunktionen (siehe oben) an: Auslesen aller Zertifikatdatensätze, Hinzufügen eines Zertifikatdatensatzes, Löschen eines Zertifikatdatensatzes, Suche nach einem Zertifikatdatensatz mit einem bestimmten Namen.
- `org.fraunhofer.osci.settings.xml.XMLFileAccess`: Kapselt die Pfadinformationen der Konfigurationsdatei und liest den entsprechenden XML Datenstrom aus.
- `org.fraunhofer.osci.settings.xml.DocumentTupelTransformer`: Transformiert den XML Datensatz in ein Java Objekt.

### Eigene Controls

Innerhalb des Fensters (Wizard und Zertifikatdialog) existieren oft gleiche Konstellationen für benötigte Controls, die von der Standardbibliotheken von SWT/JFace nicht zur Verfügung gestellt werden. Um eine Codeduplikation zu vermeiden und den Code besser lesbar zu machen wurden folgende Controls implementiert:

- `org.fraunhofer.osci.gui.controls.GridLayoutButton`: Für alle Fenster wird das GridLayout als Layout verwendet. Da die Einstellungen einzelner Schaltflächen hinsichtlich Höhe und Breite einheitlich sein und die Schaltflächengröße sich der

28 Datenströme sind in Java Datenkanäle, aus denen Daten ausgelesen oder in diese Daten geschrieben werden können. Dabei sind die Datenkanäle abstrakt gehalten, d.h. Quelle und Ziel sind nicht festgelegt, und können daher bspw. eine Datei oder auch einen einfachen String darstellen.

Textlänge anpassen soll, wurde eine entsprechende standardisierte Schaltfläche implementiert, die von der GUI verwendet werden.

- `org.fraunhofer.osci.gui.controls.ControlEnterText`: Fasst ein Eingabefeld und die zugehörige Beschriftung zusammen und präsentiert es als eigenes Dialogelement. Dazu kann diesem Control für eine Validierung ein Callback übergeben werden.
- `org.fraunhofer.osci.gui.controls.ControlEnterTextBox`: Gleiches Control wie `ControlEnterText`. Unterschied ist, dass das Eingabefeld mehrzeilig ausgelegt und scrollfähig ist.
- `org.fraunhofer.osci.gui.controls.ControlEnterTextByBrowsing`: Gleiches Control wie `ControlEnterText` mit zusätzlicher Schaltfläche. Innerhalb der Klasse existieren Standardlistener, die bei Klick auf die Schaltfläche entweder ein Dateiauswahl- oder einen Zertifikatauswahldialog öffnen.

**OSCI Szenario**

Zentral für die Ausführung des Szenarios One-Way-Message, aktiver Empfänger, Protokollierung ist die Klasse `org.fraunhofer.osci.scenario.OneWayActiveRecipient`, die folgende Szenarioschritte ausführt:

1. Initialisierung des Dialoghandlers zwischen Sender und Intermediär.
2. Anforderung der MessageID.
3. Aufbau und Sendung der OSCI Nachricht 'Zustellungsauftrag', der die Nutzungs- und Inhaltsdaten enthält.
4. Rückgabe der Antwort des Intermediärs auf den Zustellungsauftrag.

Während der Ausführung dieser Schritte wird einem Fortschrittseventhandler der Status übergeben. Dieser stellt eine Schnittstelle dar, wodurch die Implementierung der Callbacks nicht festgelegt ist. Innerhalb des OSCI Clients wurde der Forschrittseventhandler so implementiert, dass der Status der Abarbeitung an das Fortschrittsanzeigefenster weitergeleitet wird (siehe S.72).

Um die einzelnen Nachrichten zu erstellen, müssen die Rollen Intermediär, Sender und Empfänger initialisiert und ebenfalls die Nachrichtentypen und Nachrichtenteile als Objekte instanziiert werden. Diese Aufgabe wurde von `org.fraunhofer.osci.scenario.O-`

neWayActiveRecipient insofern delegiert, dass es folgende Factoryschnittstellen[29] als Klassenattribute kennt, an entsprechender Stelle die zugehörigen Methoden aufruft[30] und auf diese Weise Objekte erhält:

- org.fraunhofer.osci.factories.ICommonFactory: Instanziierung des Kommunikationskanals zwischen Sender und Intermediär.
- org.fraunhofer.osci.factories.IMessagePartFactory: Instanziierung von Content Containern, die die OSCI Inhaltsdaten kapseln.
- org.fraunhofer.osci.factories.ITypeFactory: Instanziierung der Nachrichtenklassen MessageID-Anforderung und Zustellungsauftrag.
- org.fraunhofer.osci.factories.IRoleFactory: Instanziierung der Rollen Sender, Empfänger und Intermediär.

Die vorgestellten Implementierungen verwenden dabei für die konkreten OSCI Funktionalitäten die OSCI Abstraktionsbibliothek in Java (siehe S.57). Für die Verschlüsselungs- und Entschlüsselungsroutinen der Senderzertifikate wurde die Implementierung aus dem OSCI Downloadbundle[31] im Paket de.osci.osci12.samples.impl.crypto minimal angepasst[32] und in die Klassen des Pakets org.fraunhofer.osci.scenario.crypto übernommen.

Da dem Konstruktur der Klasse org.fraunhofer.osci.scenario.OneWayActiveRecipient sechs Parameter übergeben und deswegen die Ausführung des Szenarios vereinfacht werden soll, wurde org.fraunhofer.osci.scenario.ExecuteScenarios zur Verfügung gestellt. Die enthaltene Methode fordert nur zwei Parameter: Den Fortschrittseventhandler und die im Wizard eingegebenen Daten[33]. Die Methode instanziiert mit diesen Informationen zunächst die Klasse und führt zudem den 'One-Way-Message, aktiver Empfänger' Vorgang aus.

29 Factories folgen dem Factory Design Pattern, das eine Delegierung von Objekteinstanziierung vorsieht und auf diese Weise durch Kapselung dieser Aufgabe die Wartbarkeit und bei Verwendung einer Abstract Factory ebenfalls die Testbarkeit verbessert (für eine detailliertere Diskussion siehe [Ga/He/Jo]).

30 Die Bezeichner der Factories orientieren sich an den Paketnamen der OSCI Bibliothek (siehe [OSCILib]).

31 Bereitgestellt auf www1.osci.de/sixcms/detail.php?gsid=bremen02.c.1403.de ;Zugriff: 01.10.06

32 Anpassungen erfolgten nur im strukturellem Maße durch Refactorings, die Funktionalität wurde nicht verändert.

33 Die OSCI Daten sind dabei alle in der Datenklasse org.fraunhofer.osci.settings.Settings gekapselt.

### Validierungen

Validierungen werden auf den Wizardseiten eingesetzt, um eventuelle Fehleingaben seitens des Nutzers abzufangen. Folgende Validierungen werden vorgenommen:

| | |
|---|---|
| Intermediär Adresse | Muss dem URL Pattern folgen und als Protokoll HTTP oder HTTPS verwenden. |
| Softwarezertifikate | Müssen dem Dateiformat `.p12` und `.cer` entsprechen und im Dateisystem existieren. |
| Eingabefelder | Bis auf den Nachrichtentext sind alle im Realisierungsumfang beschriebenen Daten Konfigurationen der OSCI Sendung (siehe 53) Pflichteingaben. |

Implementiert wurde die Validierungsfunktion im Paket `org.fraunhofer.osci.gui.validator` durch die Klasse `Validator`, die die Validierungsmethoden anbietet. Zudem existiert die Schnittstelle `IValidator`, die die Wizardseiten Composites (siehe 73) implementieren, um eine Validierung bei Dateneingabe des Benutzes auszuführen.

### Integritätssicherstellung und Löschen temporäres PDF Dokument

Um die Integrität des PDF Dokuments bei der OSCI Sendung sicherzustellen, wird sofort nach Start der OSCI Senderapplikation ein Hashwert der temporär abgelegten Datei gebildet und gespeichert. Direkt vor dem Einlesen des PDF Dokuments als Inhaltsdatum erfolgt dann eine erneute Berechnung und ein Vergleich mit dem vorher gespeicherten Wert. Entsprechende Methoden für die Hashwertbildung mit dem SHA Algorithmus werden durch die Klasse `SecurityActions` im Paket `org.fraunhofer.osci.security` implementiert.

Die gespeicherte PDF Datei darf aus Gründen der Geheimhaltung nach erfolgreicher Sendung oder Abbruch nicht mehr gelesen werden können und wird deswegen durch die Senderapplikation gelöscht. Dies erfolgt wie die Integritätsüberprüfung über die Klasse `SecurityActions`, die ein Löschen der Datei bei jeder Terminierung der JVM veranlasst. Dies umfasst alle möglichen Terminierungen wie Schließen Wizardfenster, erfolgreiche Sendung, Abbruch der Senderapplikation etc..

## 6.9 Implementierungsergebnis Installationsprogramm

Das Installationsprogramm liegt in einer ausführbaren Java `.jar` Bibliothek vor. Die Systemvoraussetzung für die Lauffähigkeit des Installationsprogramms ist eine instal-

lierte Java Umgebung ab Version 1.4.

Das Installationsprogramm führt Schritte in folgender Reihenfolge aus:

1. Kopieren der benötigten Bibliotheken des Plug-Ins und der OSCI Senderapplikation in ein Verzeichnis, aus dem sie geladen werden können. Als Zielverzeichnis wird der Ordner `$ACROBAT_INSTALL_DIR/Acrobat/plug_ins/OSCIPlugin/`[34] gewählt und liegt somit innerhalb des Verzeichnisses, in dem Acrobat alle Plug-Ins bei Applikationsstart lädt.
2. Setzen des Java Aufrufs als String. Innerhalb des Strings ist der Klassenpfad zur ausführbaren Bibliothek enthalten, die die OSCI Senderapplikation darstellt[35].
3. Das Acrobat Plug-In verwendet für den Start des OSCI Senderclients das Java Invocation Interface von JNI. Das Installationsprogramm setzt dafür den nötigen persistenten Eintrag in der `$PATH` Umgebungsvariable, damit die dafür benötigte und referenzierte Bibliothek `jvm.dll` beim Laden gefunden wird[36].

Fehler werden während der ausgeführten Deploymentschritte abgefangen und dem Nutzer durch entsprechende Meldungen mitgeteilt. Folgende Fehlerquellen existieren:

- Keine Schreibrechte für das persistente Setzen der `$PATH` Variable.
- Keine Schreibrechte für das Kopieren der Bibliotheken in das `$ACROBAT_INSTALL_DIR/Acrobat/plug_ins/` Verzeichnis.
- `$ACROBAT_INSTALL_DIR/Acrobat/plug_ins/` existiert nicht.

34 `$ACROBAT_INSTALL_DIR` ist keine gesetzte Umgebungsvariable sondern dient hier als Platzhalter für das Acrobat Installationsverzeichnis.

35 Ausführbare Bibliotheken bei Java definieren über die inkludierte `Manifest.mf` Datei eine Klasse mit der Einstiegsmethode `public static void main(String[] args)`, die auf diese Weise direkt durch `java -jar` aufgerufen und die Applikation direkt gestartet werden kann.

36 Für das Auffinden referenzierter dynamischer dll-Bibliotheken werden immer alle Verzeichniseinträge in der `$PATH` Variable durchsucht.

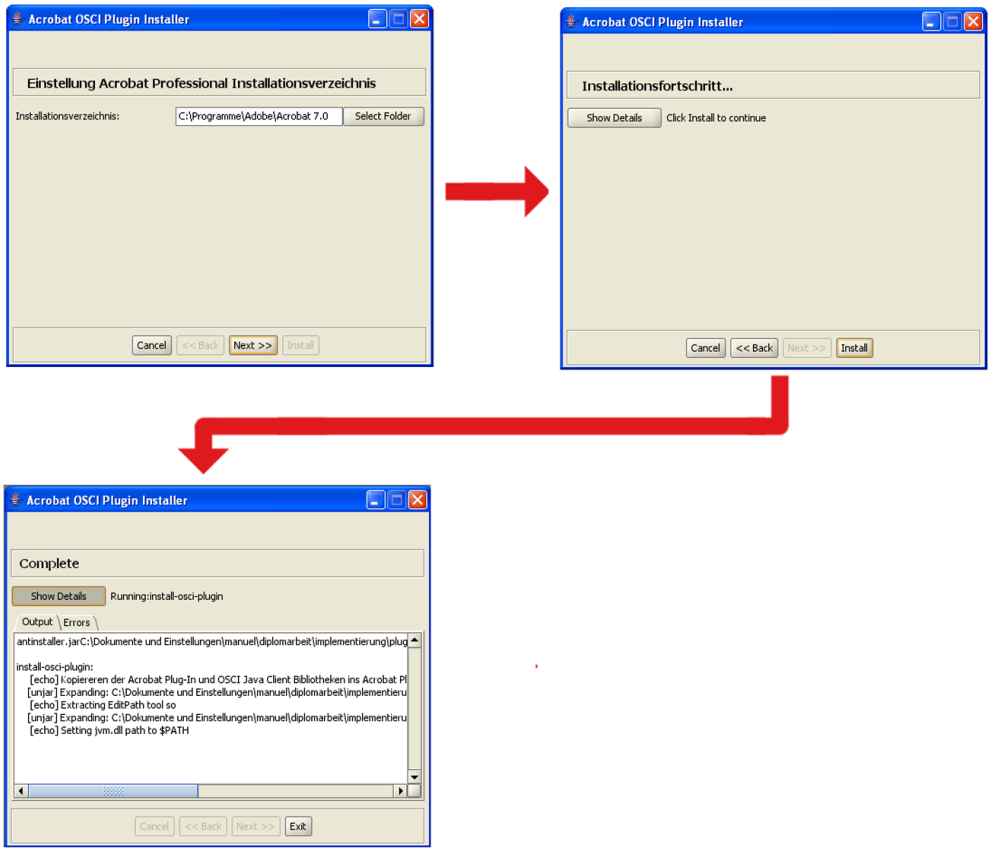


Abbildung 14: Wizard des Installationsprogramms. Einzige Eingabe des Nutzers ist der Pfad der Acrobat Installation.

### 6.9.1 Übersicht der Teilaufgaben

Für das Installationsprogramm wurden folgende Teilaufgaben identifiziert:

1. Konfiguration des Installers über Installationstool AntInstaller.
2. Implementierung Ant Task für Setzen des Java Aufrufstrings.
3. Implementierung Ant Task für Setzen `jvm.dll`-Pfad in der `$PATH` Variable.

#### Entwicklungstools

Folgende Entwicklungstools wurden verwendet:

- **Eclipse IDE:** Wurde für die Entwicklung der Ant Tasks verwendet. Für Eclipse siehe Beschreibung S.69.
- **PSPad:** PSPad ein freier Editor[37], der u.a. Syntaxhighlighting für XML Dateien unterstützt, Verwendung für die Editierung der XML Konfigurationsdateien des

37 Download unter www.pspad.com/de/

AntInstallers.

- **AntInstaller:** AntInstaller (für Download und Dokumentation siehe [AntInstaller]) ist ein Tool, mit dem Installationsprogramme mit Hilfe von XML Konfigurationsdateien generieren werden. Dabei wird Ant als Builtool verwendet, so dass sich Installationsschritte unkompliziert durch eigene Ant Tasks erweitern lassen[38]. Verwendete Version ist 0.7.4 beta.
- **Ant:** Ant ist ein populäres in Java implementiertes Buildtool (für umfassende Dokumentation siehe [Til/Bur]), das sich leicht erweitern lässt. Verwendete Version ist 1.6.5.
- **EditPath:** EditPath (für Download und Dokumentation siehe [EditPath]) ist ein Kommandozeilentool für Windows, mit dem man die Umgebungsvariable `$PATH` persistent setzen kann, so dass bei jedem Systemstart von Windows alle Bibliotheken innerhalb der Pfadeinträge geladen werden können. Verwendete Version ist 1.0.

### 6.9.2 Umsetzung

**Konfiguration Installer**

Der Installer wird über drei XML Konfigurationsdateien erstellt:

1. `antinstall-config.xml`: Konfiguriert den Installerwizard, der zwei Wizardseiten besitzt. Die erste Seite nimmt das Installationsverzeichnis der Acrobat Applikation als Dateneingabe entgegen. Auf der zweiten Seite wird der Installationsvorgang bestätigt.
2. `install-acrobat-osci.xml`: Konfiguriert die einzelnen Installationsschritte, um die Acrobat OSCI Integrationslösung lauffähig zu machen.
   - Validierung, dass `$ACROBAT_INSTALL_DIR/Acrobat/plug_ins/` existiert.
   - Kopieren der Plug-In und OSCI Senderapplikation Bibliotheken nach `$ACROBAT_INSTALL_DIR/Acrobat/plug_ins/`.
   - Kopie von EditPathTool (siehe oben Entwicklungstools) in ein temporäres Verzeichnis.
   - Aufruf Ant Task (siehe unten), um Java Aufrufstring im System persistent zu setzen. Dabei wird der Klassenpfad des ausführbaren OSCI Java Clients

38 Für Beschreibung des Task Modells von Ant und die Vorgehensweise bei der Implementierung siehe [Til/Bur,S.80ff.].

und der Pfad der vom OSCI Client verwendeten externen Bibliotheken übergeben.

- Aufruf Ant Task (siehe unten), um Pfad der `jvm.dll` in `$PATH` hinzuzufügen. Gesetzt wird der Pfadeintrag mit dem EditPath Tool.

3. `build.xml`: Diese Builddatei wird von Ant verwendet, um den Installer als ausführbare Java Bibliothek zu generieren. Konfiguriert wird, dass die Acrobat Plug-In und OSCI Java Client Bibliotheken und die eigenen Anttasks (siehe unten) als Resourcen in das Installationsprogramm eingebunden werden. Zusätzlich wird auf die anderen Konfigurationsdateien `antinstall-config.xml` und `install-acrobat-osci.xml` verwiesen[39].

Um den Installer zu generieren und auszuführen müssen über die Kommandozeile folgende Schritte vorgenommen werden. Der aktuelle Pfad muss das Verzeichnis sein, in dem die Installationsartefakte liegen:

- `$ ant build.xml` : Generierung von `installAcrobatOSCIPlugin.jar`.
- `$ java -jar installAcrobatOSCIPlugin.jar` : Starten Installationsprogramm.

Optional kann das Installationsprogramm auch per Doppelklick auf `installAcrobatOSCIPlugin.jar` über einen grafischen Dateisystemmanager gestartet werden.

### Ant Task: Setzen Java Aufrufstring

Der Java Aufruf besitzt folgendes Muster[40]:

```
java -Djava.library.path=$PFAD_OSCI_SENDER_LIBS -jar $PFAD_OSCI_SENDER_JAR
```

Die ausführbare `$OSCI_SENDER_JAR` Bibliothek inkludiert dabei über die `Manifest.mf` die von der Senderapplikation referenzierten Bibliotheken, die im Verzeichnis `$PFAD_OSCI_SENDER_LIBS` liegen. Für das Setzen des Java Aufrufstrings wurden folgende Klassen implementiert:

- `org.ant.install.acrobatosci.JavaCallSetterTask`: Schnittstelle zum Ant Buildtool, in der zwei Optionen aus der Builddatei entgegengenommen werden: Der absolute Klassenpfad zum OSCI Java Client als ausführbare Bibliothek und absoluter Pfad zum Verzeichnis, in dem die vom OSCI Client verwendeten `.jar`

39 Bei der Generierung des Installers muss beachtet werden, dass Ant in dem Verzeichnis ausgeführt wird, in dem die `build.xml` mit allen anderen Bibliotheken liegt. Andernfalls werden die Resourcen nicht gefunden und können nicht inkludiert werden. Zudem muss für den Generator Ant installiert sein. Für die Ausführung des Installationsprogramms wird Ant nicht benötigt.

40 Die mit $ angeführten Bezeichner sind Platzhalter.

Bibliotheken liegen. Diese Optionen werden danach `JavaCallSetter` direkt übergeben.

- `org.ant.install.acrobatosci.JavaCallSetter`: Die Optionen werden von `JavaCallSetterTask` entgegengenommen und persistent über die Java SDK API `java.util.prefs` systemweit[41] gespeichert.

**Ant Task: Setzen `$PATH`**

Für das persistente Setzen des Pfades der `jvm.dll` in die `$PATH` Variable wurden folgende Klassen implementiert:

- `org.ant.install.acrobatosci.JVMDllPathSetterTask`: Schnittstelle zum Ant Buildtool. Es wird der Pfad des EditPath Tools entgegengenommen und `JVMDllPathSetter` direkt übergeben.
- `org.ant.install.acrobatosci.JVMDllPathSetter`: Um einen Prozess als EditPath zu starten wird `java.lang.Runtime.exec()` aus der Java API verwendet[42]. Zunächst wird die Systemproperty `java.home` ermittelt, die den Pfad der Java Installation darstellt. Der EditPath übergebene Pfad zur Bibliothek jvm.dll ist dabei immer einheitlich[43]. Folgender Aufruf von EditPath erfolgt: `EditPath.exe -s -a $JVM_DLL_PFAD`. Die Option `-s` aktiviert dabei die systemweite Pfadvariable, `-a` signalisiert das Hinzufügen eines Eintrags.

Weiterhin überprüft der Ant Task, ob ausreichende Systemrechte für das Setzen der `$PATH` Variablen existieren und reicht bei Fehlschlag den Fehler an AntInstaller weiter.

41 Eine systemweite Speicherung über die Preferences API wird durch das Initialisieren mit `java.util.prefs.Preferences.systemRoot()` erreicht. Für weitere Informationen zur Preferences API siehe javaalmanac.com/egs/java.util.prefs/pkg.html; Zugriff 11.11.2006.

42 Für Funktionsweise siehe java.sun.com/developer/JDCTechTips/2003/tt0304.html ;Zugriff 11.11.2006.

43 Bei einer Java SDK Installation liegt jvm.dll immer in $JAVA_HOME/jre/bin/client/ und bei einer Java JRE Installation in $JAVA_HOME/bin/client/

# 7 Bewertung der Implementierung

Die Acrobat OSCI Lösung wurde erfolgreich implementiert und folgt den im Realisierungsumfang beschriebenen funktionalen Anforderungen (siehe S.53).

### 7.1.1 Nicht-funktionale Eigenschaften

**Sicherheit**

Die PIN, die den Zugang zum geheimen Schlüssel in der Zertifikatsdatei ermöglicht, wird in der GUI auf den Wizardseiten nicht im Klartext, sondern verdeckt angezeigt und schützt auf diese Weise den geheimen Schlüssel.

Das aktuell geöffnete PDF Dokument wird temporär abgespeichert und später in die OSCI Sendung integriert. Da hier die Gefahr besteht, dass bei schlechter Betriebssystemkonfiguration jemand anderer Zugriff auf die Datei erhält, sie korrumpiert und schließlich die falschen Inhaltsdaten übertragen werden, wird die Integrität der temporären Datei seitens der OSCI Senderapplikation mit der Bildung und dem Vergleich von Hashwerten sichergestellt. Die Sicherstellung der Geheimhaltung des gespeicherten Dokuments beschränkt sich auf die Rechtevergabe des Betriebsystems und ist deswegen nicht absolut verlässlich. Allerdings wird die Datei nach Beenden der Senderapplikation sofort gelöscht, so dass die Dokumente nicht länger im temporären Verzeichnis liegen und eingesehen werden können. Die Gefahr der Verletzung der Geheimhaltung besteht also nur für die Zeit, in der die Senderapplikation ausgeführt wird.

Die Sicherheit des Installers ist direkt abhängig von der Tatsache, ob die im Installationspaket enthaltenen Bibliotheken nicht korrumpiert werden, was allerdings durch Archivprogramme leicht möglich ist. Ein weiteres Sicherheitsproblem könnte das verwendete EditPath Tool sein. Zwar wurden nach intensiver Recherche im Internet keinerlei Anhaltspunkte für eine Gefahr seitens dieses Tools gefunden, aber da dieses nicht in Open Source vorliegt, kann diese Aussage nicht eindeutig überprüft werden.

**Benutzerfreundlichkeit**

Der Aufruf der Acrobat OSCI Integrationslösung erfolgt im Acrobat Plug-In über ein Toolbaricon, das durch eine OSCI Grafik visualisiert ist. Alternativ dazu existiert wie aus anderen Applikationen gewohnt auch ein funktionsgleicher Menüeintrag. Da die Acrobat OSCI Integrationslösung das aktuell geöffnete Dokument als Inhaltsdatum ver-

wenden soll, macht die Ausführung der OSCI Senderapplikation nur Sinn, wenn ein PDF Dokument als Inhaltsdatum in Acrobat geöffnet ist. Daher sind die Einträge des Menüs und der Toolbar nur aktiv, wenn ein PDF Dokument in der Acrobat Applikation auch geöffnet ist.

Die Java OSCI Senderapplikation in Form eines Wizards folgt ebenfalls den ergonomischen Richtlinien. So orientiert sich die Fensteraufteilung an den bekannten Wizards der Eclipse IDE. Die Aufteilung ist gewohnt strukturiert in Beschreibung (oben), Eingabefeldern (mitte) und einer Schaltflächenleiste (unten). Zudem wird eine Validierung der Nutzereringaben vorgenommen, um eine fehlerhafte Dateneingabe abzufangen. Wenn eine Validierung fehlschlägt, erfolgt eine aussagekräftige Fehlermeldung im oberen Bereich der Wizardseite.

Da die Eingabe der Daten für die OSCI Sendung in mehreren Schritten über Wizardseiten erfolgt, wird der Benutzer nicht mit Dialogelementen überhäuft und hat einen besseren Überblick. Dabei lässt sich der Wizard über die die Schaltflächen „Zurück“, „Weiter“, „Fertig stellen“ und „Abbrechen“ steuern. Hier sind die Schaltflächen erst aktiv, wenn die entsprechende Steuerungsfunktion im derzeitigen Zustand zur Verfügung steht. „Fertig stellen“ ist erst aktiv, wenn die letzte Seite des Wizards erreicht wurde. „Weiter“ ist nur auswählbar, wenn für die aktuelle Wizardseite alle Nutzereingaben erfolgreich validiert wurden. Alle Wizardseiten besitzen weiterhin im oberen Teil einen Bereich, der den Kontext der einzelnen Wizardseite mit Text und Grafik beschreibt. Da die Abwicklung des OSCI Szenarios[1] einige Zeit in Anspruch nehmen kann und der Nutzer nicht über einen Absturz des Programms spekulieren soll, existiert außerdem ein Fortschrittsbalken, der entsprechend Auskunft über den Zustand der Abwicklung gibt. Falls während dieser ein Fehler auftritt (bspw. der Intermediär ist nicht erreichbar) wird ein kurze Fehlermeldung durchgereicht, bevor die Senderapplikation beendet wird.

Damit die Softwarezertifikate, die oft verwendet werden, nicht immer langwierig über einen Dateiauswahldialog gesucht und geöffnet werden müssen, werden die Zertifikate in einem weiteren Dialog hinzugefügt und gelöscht[2], so dass eine Zertifikatauswahl schnell erfolgen kann. Diese Zertifikatauswahleinstellung ist dabei dem Konto des im Betriebssystem aktuell angemeldeten Nutzers zugeordnet.

Das Installationsprogramm ist ebenfalls, wie aus anderen Installationsroutinen bekannt, durch einen Wizard realisiert und nimmt eine Dateneingabe entgegen. Für den Start

1 Die Abwicklung des Szenarios beginnt, nachdem die Sendung durch den Klick auf den Button „Fertig stellen“ bestätigt wurde.

2 Die Zertifikate werden lediglich aus der Zertifikatmenge der Konfiguration entnommen. Natürlich existieren die Zertifikate immer noch im Dateisystem.

des Installationsprogramms reicht bei einer installierten Java Umgebung ein Doppelklick auf die direkt ausführbare Datei aus. Beim Fehlschlagen der Installation (für Fehlerquellen siehe S.79) wird außerdem eine Fehlermeldung mit der Beschreibung der Ursache ausgegeben.

**Plattformunabhängigkeit**

Eine komplette Plattformunabhängigkeit konnte durch die Verwendung von C/C++ nicht erreicht werden, da hier der Source Code zumindest neu kompiliert werden muss. Zudem wurde für das Laden des Toolbaricons eine windowspropriertäre Routine verwendet, so dass auf anderen Plattformen der Code dahingehend angepasst werden muss. Ansonsten wurde immer auf die portable Acrobat API und Standard APIs wie `<string>`,`<iostream>` zurückgegriffen, wodurch hier keine Anpassungen nötig sind. Das Installationsprogramm ist ebenfalls nicht plattformunabhängig, da es die `$PATH` Variable über das für Windows geschriebene EditPath Tool setzt.

Allerdings wurden Vorraussetzungen geschaffen, die Migration auf ein anderes von Acrobat unterstütztes Betriebssystem zu erleichtern. Da die OSCI Senderapplikation in Java geschrieben wurde, muss sie weder angepasst noch neu kompiliert, sondern kann direkt auf die jeweilige Plattform übernommen werden. Einzige Vorraussetzung ist, dass der Java Applikation, wie bei der aktuellen Windows Lösung, beim Aufruf ebenfalls ein Argument übergeben wird, das den Pfad zur temporären PDF Datei darstellt. Darüberhinaus wurden Maßnahmen hinsichtlich der Wartung (siehe unten) getroffen, die eine Migration und/oder Anpassung erleichtern.

**Wartbarkeit**

Alle Teilaufgaben (siehe Kapitel Implementierungsergebnisse) wurden modularisiert. Bei C++ erfolgte dies über die Deklaration der öffentlich sichtbaren Prozeduren in eigene Header Dateien (Dateiformat `.h`) und die entsprechenden Definitionen in Implementierungsdateien (Dateiformat `.cpp`). Bei Java erfolgte die Modularisierung über Klassen, öffentliche Methoden und Paketeinteilungen.

Auch wurden Methoden, Prozeduren, Funktionen und Variablen passend benannt und entsprechende Rümpfe möglichst kurz gehalten. Dadurch ist der Source Code gut nachvollziehbar, verständlich und Erweiterungen können leichter umgesetzt werden. Allerdings leidet seitens des Plug-Ins bei der Initialisierung des Plug-Ins und dem JNI Programmcode die Verständlichkeit darunter, dass keine kompaktere Schreibweise

seitens der API möglich war. Durch vermehrter Verwendung von Kommentaren innerhalb der Prozedur- und Funkionsrümpfen wurde dieser Umstand etwas abgeschwächt.

Bei Java wurde viel mit Schnittstellen und Dependency Injection[3] gearbeitet, so dass das System loser gekoppelt ist und auf diese Weise die Testbarkeit verbessert wird, was die Wartbarkeit grundsätzlich verbessert.

Die Konfigurationsdateien des Installationsprogramms sind vom Umfang her sehr überschaubar und die Installation lässt durch die deskriptive Konfiguration leicht verändern und anpassen. Auch ist eine Erweiterung von Installationsvorgängen, die AntInstaller im Kern nicht anbietet, über weitere eigene Ant Tasks einfach möglich.

### 7.1.2 Technologische Einschätzung und Empfehlungen

Grundsätzlich wurde die Lösung erfolgreich implementiert, allerdings erfolgte sie über Umwege, da die derzeitigen technischen Vorraussetzungen Hindernisse für eine Acrobat OSCI Integration mit sich brachten. Hauptproblem war hier die Kommunikation zwischen dem Acrobat SDK und den OSCI APIs in Java und .Net. Die Integrationsschwierigkeiten lassen sich aus unterschiedlichen Sichten betrachten:

1. Weder mit der Acrobat JavaScript noch mit der Plug-In Technologie ist es möglich, direkt auf Java oder .Net Bibliotheken zuzugreifen. Zwar existiert für .Net die Möglichkeit das sogenannte Managed C++ auf der .Net CLR (Common Language Runtime) laufen zu lassen, allerdings existiert auf Seiten von Acrobat C/C++ keine Möglichkeit das Plug-In auf dieser CLR laufen zu lassen oder Zugriff auf das .Net Framework zu erhalten.
2. Die Acrobat Technologien stellen in den SDK APIs keine Bibliotheken zur Verfügung, die ähnlich verwendet werden können wie die von der OSCI Leitstelle in Java und .Net.
3. Die OSCI Leitstelle stellt keine C/C++ Bibliotheken zur Verfügung, die die gleiche Funktionalität bieten wie die bereits zur Verfügung stehenden in Java und .Net.

An diese Punkte schließt sich eine direkte Empfehlung sowohl an Adobe als auch an die OSCI Leitstelle an, um zukünftige OSCI Integrationen für Acrobat zu vereinfachen: Die OSCI Leitstelle könnte eine OSCI Bibliothek für C/C++ oder JavaScript zur Verfügung stellen, die über Acrobat angesprechbar ist. Adobe könnte andererseits ihr Acro-

3 Mit Dependency Injection werden Instanzvariablen per Konstruktur oder über Setter Methoden gesetzt.

bat-System auf ein .Net kompatibles C++ migrieren[4], was allerdings aus Gründen der Migrationskosten sehr unrealisitisch erscheint. Sinnvoller wäre es, innerhalb des Acrobat SDK, OSCI Funktionalitäten ähnlich zu den bereits verfügbaren Bibliotheken für die Plug-In Variante in C/C++ oder für JavaScript anzubieten.

### 7.1.3 Ausblick für Erweiterungen

Derzeit implementiert der OSCI Client nur das Szenario One-Way-Message, aktiver Empfänger. Zukünftige Erweiterung auf die anderen Grundszenarien (siehe S.44) wären durchaus sinnvoll, um eine vollständige OSCI Senderimplementierung zur Verfügung zu haben. Da Wert auf eine gute Wartung gelegt wurde, existieren gute Vorraussetzung, die anderen Grundszenarien in die aktuelle Implementierung zu integrieren.

Momentan wird die Datei als unverschlüsseltes PDF Dokument temporär abgespeichert, so dass jeder diese während des Laufens der Senderapplikation und bei Zugriffsrecht auf die Datei lesen kann. Um eine Geheimhaltung bei besonders schützenswerten Inhalten sicherzustellen, kann ein Verschlüsselungsmechanismus eingesetzt werden, bei dem das Acrobat Plug-In die Datei mit einem Sitzungsschlüssel verschlüsselt abspeichert, den Sitzungsschlüssel der Senderapplikation übergibt und diese den eingelesenen Datenstrom wieder entschlüsselt. Ein weiteres Sicherheitsproblem könnte das Installationspaket als ausführbare `.jar` Datei sein. Hier könnten die zu installierenden Artefakte korrumpiert werden, so dass man bei der Installation der OSCI Acrobat Lösung schadhaften Code installiert. Eine Lösung wäre, das Installationspaket als `.jar`-Paket zu signieren, so dass die Integrität der Inhalte gewährleistet bzw. eine Veränderung der Artefakte aufgedeckt wird[5].

Es könnten schließlich noch einige Verbesserungen in der Benutzerfreundlichkeit vorgenommen werden. So findet während Abwicklung des OSCI Szenarios lediglich ein rudimentäre Fehlerbehandlung statt. So wird bspw. davon ausgegangen, dass die im Wizard eingegebene PIN für den Zugriff auf den geheimen Schlüssel passt, dass der Intermediär und dass das eingestellte Zertifikat diesem zugeordnet ist. Auch wird bisher keine detaillierte Diagnose der Nachrichten Return Codes (siehe [OSCISpec,S.27ff.]) vorgenommen, mit Hilfe derer eine entsprechende Meldung an den Nutzer des OSCI Clients ausgegeben werden kann. Hier könnten Erweiterungen

4 Siehe C++/CLI: msdn2.microsoft.com/de-de/library/ms235289(VS.80).aspx ; Zugriff: 02.01.2006

5 Im Installationspaket des Java SDK kann dafür das Tool jarsigner verwendet werden (siehe java.sun.com/j2se/1.5.0/docs/tooldocs/windows/jarsigner.html; Zugriff: 12.1206).

vorgenommen werden, die die von der OSCI Bibliothek internen geworfenen Exceptions besser analysieren und dem Nutzer aussagekräftigere Meldungen ausgeben. Außerdem wäre es sinnvoll, dem Nutzer die Möglichkeit zu geben, den am Ende angezeigten Laufzettel zu speichern, so dass ein späterer Zugriff auf diesen möglich ist. Derzeit wird ein Zertifikatauswahldialog verwendet, um Zertifikate einzustellen. Es wäre eine Erweiterung dahingehend sinnvoll, Zertifikate aus dem im Windows umgesetzten Zertifikatspeicher laden zu können, um die betriebssystemweite Sicht auf Zertifikate in die Senderapplikation zu integrieren.

Damit die Acrobat OSCI Integration ebenfalls auf den anderen von Acrobat unterstützten Betriebssystemen Unix und Macintosh lauffähig ist, müsste neben der minimalen Anpassung und Neukompilierung des Acrobat Plug-Ins für das Funktionieren des Java Invocation Interface das Installationsprogramm die Installationsschritte über die AntInstaller Konfiguration anpassen und weitere Ant Tasks integrieren, die den Systempfad ähnlich zum EditPath Tool für das Auffinden von `jvm.dll`[6] einstellen. Idealerweise würde insgesamt ein plattformunabhängiges Installationspaket zur Verfügung stehen, dass zunächst überprüft, auf welchem Betriebssystem es ausgeführt wurde und dementsprechend die passenden Installationsschritte ausführt.

6 Auf anderen Betriebssystemen würde die Bibliothek `jvm.dll` natürlich einen entsprechenden anderen Bezeichner haben.

# Anhang A

## Überblick der öffentlichen Verwaltung

Nach dem Staatsorganisationsrecht (als Teilmaterie des Öffentlichen Rechts) ist die Staatsgewalt als Gewaltenteilung auf drei verschiedene Organe verteilt (GG,Art.20.Abs.2): Der Judikative (Rechtssprechung), Legislative (Gesetzgebung) und Exekutive (Ausführung). Die Rechtsprechung wird dabei von den Gerichten und die Gesetzgebung auf Bundesebene vom Bundesparlament und dem Bundesrat als Vertretung der einzelnen Bundesländer wahrgenommen.

Die Macht der Exekutive als ausführende Gewalt wird dagegen durch Bundespräsidenten und der Bundesregierung eingenommen. Der Bundespräsident hat in der BRD, auch aus den Erfahrungen der Weimarer Republik, eher repräsentativen Charakter [Son,Ble, S.329 ff.], wodurch sich seine formal zugesprochene Macht in der Exekutive relativiert. Das maßgebliche Exekutivorgan ist dementsprechend die Bundesregierung, die aus dem Bundeskanzler und den von ihm ausgewählten Bundesministern besteht. Die Bundesminister stellen dabei die oberste Ebene der öffentlichen Verwaltung dar.

### Gesetzmäßigkeit der Verwaltung

Die öffentliche Verwaltung ist Teil der Exekutive, wonach sie als Teil der Staatsgewalten dem Grundsatz der Gesetzmäßigkeit unterliegt (GG,Art.20.Abs.3). Dadurch gelten folgende Grundsätze:

1. Vorrang des Gesetzes: Eine Verwaltungshandlung darf nicht gegen bestehende Gesetze verstoßen.
2. Vorbehalt des Gesetzes: Handlungen der Verwaltung dürfen nur ausgeübt werden, wenn diese ausdrücklich durch vorhandene Gesetze legitimiert sind[1].

Ferner ist die Forderung an das Gesetz, dass es dem sogenannten Bestimmtheitsgrundsatz obliegt, also konkret genug und nicht zu vage definiert ist und auf diese Weise auf Seiten der exekutiven Staatsgewalt und des Staatsvolkes Rechtssicherheit

1 In bestimmten Einzelfällen wird auch die Lehre des Teilvorbehalts vertreten, durch den die öffentliche Verwaltung auch ohne ein konkret existierendes Gesetz auf Basis von Auslegungen handeln darf. Dies soll keineswegs den Grundsatz des Vorbehalts des Gesetzes verletzen sondern soll nur der dynamischen Natur von Entscheidungsumständen Rechnung tragen .

herrscht[2].

Kodifiziert ist das Recht der Verwaltung im Verwaltungsrecht (als Teilmaterie des Öffentlichen Rechts), welches untergliedert wird in Verwaltungsprozessrecht, allgemeines und besondere Verwaltungsrecht [Arn/Rud,S.215]. Das allgemeine Verwaltungsrecht enthält dabei Grundsätze, die jedes Verwaltungshandeln durch Verwaltungsträger festlegen. Das Verwaltungsprozessrecht regelt den Rechtsschutz des Bürgers gegenüber der Verwaltung und das spezielle Verwaltungsrecht legt domänenspezifische Normen wie z.B. im Polizeirecht oder Gewerberecht fest.

## Definition der Verwaltung

Eine feste Definition[3] der öffentlichen Verwaltung ist aufgrund der komplexen Eigenheiten dieser schwierig und sollte vermieden werden. Daher ist es sinnvoll auf eine beschreibende Definition zurückzugreifen [Weltecke]:

### Aspekte der Verwaltung

- Öffentliche Verwaltung im materiellen Sinn: Eine konkrete Verwaltungstätigkeit, wie Erteilung von Genehmigungen oder Informationsdienstleistungen.
- Öffentliche Verwaltung im formellen Sinn: Die gesamte von den Verwaltungseinrichtungen ausgeübte Tätigkeit.
- Öffentliche Verwaltung im organisatorischen Sinn: Organisationsstruktur der Verwaltung (siehe Kapitel  und ).

### Aufgaben der Verwaltung

- Ordnungsverwaltung: Aufrechterhalten der öffentlichen Sicherheit und Gefahrenabwehr.
- Leistungsverwaltung: Unterstützung Einzelner, bspw. finanzielle Hilfe oder Bereitstellung öffentlicher Einrichtungen.
- Bedarfsverwaltung: Bereitstellung von Sachmitteln (Rechner, Schreibtische etc.).
- Abgabenverwaltung: Erhebung und Eintreibung von Geldmitteln aus Steuern.
- Lenkungsverwaltung: Maßnahmen zur Steuerung ganzer Bereiche (wie Kultur,

2 Natürlich muss die Rechtssicherheit auf Seiten des Staatsvolkes relativiert werden, da wohl kaum ein Bürger als Nichtjurist alle Gesetze kennen wird und so die Auswirkungen seiner Handlungen komplett überblicken könnte.

3 Also positive oder negative Definition

Wirtschaft) bspw. durch Subventionen.

**Rechtswirkung auf den Bürger**

- Leistungsverwaltung[4]: Der Bürger nimmt hier staatliche Hilfen in Anspruch, es ergeben sich also Vorteile für diesen, wie die Beanspruchung auf staatliche Hilfen wie Arbeitslosengeld oder Sozialhilfe.
- Eingriffsverwaltung: Der Bürger wird durch Gebote oder Verbote in seiner Freiheit benachteiligt bzw. eingeschränkt.

## Verwaltungshandeln

Die Ausführung aller Aufgaben, die der Verwaltungsorganisation[5] zugeordnet ist, wird als Verwaltungshandeln bezeichnet. Ferner wird in formalen und informellen Handlungsformen kategorisiert [Sod/Zie,S.455ff].

**Formales Verwaltungshandeln**

Hier ist die Verwaltungshandlung rechtlich geregelt, es wird also Gebrauch von einer Rechtsnorm gemacht und es soll dementsprechend einen Rechtserfolg[6] erzielt werden. Beispiele sind Verwaltungsakte, der öffentlich rechtliche Vertrag und der Erlass von Rechtsverordnungen und Satzungen[7]. Privatrechtliches Verwaltungshandeln, bspw. Kauf von Dienstfahrzeugen, wird zum formalen Verwaltungshandeln zugewiesen, da hier ebenfalls ein Rechtserfolg (Kaufvertrag nach BGB) entsteht.

**Informelles Verwaltungshandeln**

Informelles Verwaltungshandeln handelt nicht direkt aus rechtlichen Vorschriften, es erfolgt also keine Rechtsbindung. Ziel dabei ist kein rechtlicher sondern ein tatsächlicher Erfolg. Die Motivation beim tatsächlichen Erfolg geht also nicht aus einem Gesetz sondern aus der Herbeiführung eines konkreten Zustands hervor. Allerdings ist es möglich, dass die Entscheidung über die Ausführung der Aufgabe zum formalen Verwaltungshandeln zugeordnet werden kann [Becker,S.461]. Als Beispiel für informelle Verwal-

4 Unterscheidet sich zum bereits identisch benannten Begriff als Aufgabentyp dadurch, dass hier keine allgemeine Aufgabe der Verwaltung gemeint ist. Im Gegensatz dazu wird eine konkrete Rechtsfolge angewendet und es ergibt sich ein konkretes Ergebnis im Einzelfall des Bürgers.

5 Für eine nähere Beschreibung der Verwaltungsorganisation siehe Anhang B.

6 Ein Rechtserfolg ist die konkrete Anwendung eines Gesetzes.

7 Rechtsverordnungen und Satzungen werden durch die Exekutive erlassen und sind nur rechtskräftig, wenn formale Gesetzen sie ausdrücklich erlauben. Formale Gesetze sind Gesetze, die von der legislativen Gewalt durch Mitglieder des Bundestages, Bundesrates, der Landesparlamente und bei Plebisziten direkt vom Volk verabschiedet werden.

tungshandlungsformen kann eine Dienstfahrt, die Instandsetzung einer Straße, behördliche Auskünfte, Warnungen, Bestellung eines Stiftes oder auch der Schulunterricht gezählt werden.

# Anhang B

## Verwaltungsorganisation

Um einschätzen zu können, welche rechtlichen Gesetze die öffentliche Verwaltung befolgen und einhalten muss (Vorbehalt und Vorrang des Gesetzes), ist es nötig zu wissen, in welchen Rechtsformen diese als Organisationseinheit in der Verwaltung tätig ist.

In der Verwaltungsorganisation werden die Organisationseinheiten in Makro- und Mikrostruktur unterschieden [Becker,S.190][1].

### Makrostruktur

Die Makrostruktur beschreibt die grobe Struktur und die Aufgabenverteilung der Verwaltung. Grundsätzlich sind die einzelnen Verwaltungsinstitutionen, die auch Verwaltungsträger genannt werden, hierarchisch strukturiert. Grundsätzlich sind alle Verwaltungsträger organisatorisch selbstständig und rechtsfähig und besitzen dementsprechend die Befugnis bzw. die Pflicht Verwaltungsaufgaben auszuführen.

Bei den Verwaltungsträgern wird darüberhinaus noch in Landesverwaltung und Bundesverwaltung kategorisiert. Die Landesverwaltungen führen sowohl Bundes -als auch Landesgesetze durch. Dabei ist zu bemerken, dass die Kompetenzen der Ausführung der Bundesgesetze klar auf Seiten der Länder besteht (GG Art.83 und 84), der Bund nimmt hier lediglich die Stellung der Aufsicht ein (GG Art.84 S.3 und Art. 85 S.3,4). Einige Verwaltungskompetenzen werden allerdings durch den Bund wahrgenommen (Art.87-89), bspw. Auswärtiges Amt oder Bundeswehrverwaltung.

Darüberhinaus werden Verwaltungsaufgaben auch von Kommunen ausgeführt, die allerdings den Ländern untergeordnet sind. Diese Einteilung wird naheliegenderweise

1 Es werden folgend die anschaulicheren Begriffe Makro- und Mikrostruktur verwendet. [Becker] verwendet anstelle derer die deckungsgleichen Fachausdrücke institutionelle und innere Organisation.

auch räumlich vorgenommen. So ist die Bundesverwaltung auf oberste Staatsebene angelegt, die Länderverwaltung den Bundesländern zugeordnet und die Kommunalverwaltung lokal in den Gemeinden, Städten und Kreisen organisiert. Diese räumliche Aufteilung der Verwaltung wird, da sie hierarchisch organisiert ist, dementsprechend vertikale Dezentralisierung genannt [Becker].

Eine Dezentralisierung hat außerdem die Funktion, dass der Staat einerseits als Bund oder Land selbst als Verwaltungsträger tätig wird oder einen anderen öffentlich rechtlichen Verwaltungsträger für Verwaltungstätigkeiten beauftragt und auf diese Weise Aufgaben auslagert. Daraus entsteht entsprechend eine direkte und indirekte Verwaltungsaufgabenausführung durch unmittelbare bzw. mittelbare Verwaltungsträger. Ferner können Verwaltungsaufgaben auch durch nicht öffentlich rechtlich sondern privatrechtlich organisierte juristische oder natürliche Personen wahrgenommen werden.

Um den Dezentralisierungssachverhalt besser zu verdeutlichen soll die unmittelbare, mittelbare und privatrechtliche Verwaltung näher betrachtet werden:

**Unmittelbare Verwaltung**

Unmittelbare Verwaltungsträger liegen vor, wenn Verwaltungsaufgaben durch den Staat in Form von Bund oder Ländern in eigenen Behörden wahrgenommen werden, die organisatorisch, aber nicht rechtlich selbstständig sind [BundOnline]. Legitimiert werden diese Handlungen auf Bundes- und Landesebene direkt aus den Gesetzen des Grundgesetzes.

Dadurch entstehen auf Bundesebene oberste Bundesbehörden (u.a. alle Bundesministerien), Bundesoberbehörden (z.B. Bundesamt für Zivildienst), Bundesmittelbehörden (z.B. Bundespolizeidirektion) und Untere Bundesbehörde (z.B. regionale Hauptzollämter). Auf Landesebene ergeben sich deckungsgleich oberste (z.B. Landesregierung), obere (z.B. Landesämter), mittlere (z.B. Bezirksregierungen) und untere Landesbehörden (z.B. Schulämter)[2]. Die konkrete Landesverwaltungsorganisation ist dabei von Land zu Land unterschiedlich [Sod/Zie,S.375][3].

2 Die aufgezählten Verwaltungsträger in Klammern dienen nur zur groben Anschauung und wurden durch Internetrecherche vornehmlich auf www.bund.de und www.thueringen.de/de/politisch/behoerden_land/behoerdenstruktur/ ermittelt.Zugriff:11.9.06

3 So besitzt ein von der Fläche größerer Flächenstaat in der Regel eine mehrstufigere Hierarchie als ein Stadtstaat.

**Mittelbare Verwaltung**

Die mittelbare Staatsverwaltung ist dadurch gekennzeichnet, dass sie als vom Staat verselbstständigte Verwaltungsträger öffentlich rechtlich handeln, allerdings noch unter der Aufsicht des Staates stehen. Durch diese Dezentralisierung wird der Staat einerseits von Verwaltungsaufgaben entlastet, verfügt aber dessen ungeachtet noch über eine Kontrolle. Sowohl auf Landes -als auch auf Bundesebene existieren dabei drei Formen von Verwaltungsträgern: Körperschaften, Anstalten und Stiftungen des öffentlichen Rechts [Sod/Zie,377 ff]:

- Körperschaften des öffentlichen Rechts (z.B. Universitäten, Gemeinden) existieren aufgrund von Gesetzen und bestehen aus Mitgliedern. Ziel dieser Körperschaften ist die Ermöglichung der Selbstverwaltung durch die Betroffenen als Mitglieder.
- Anstalten des öffentlichen Rechts (z.B. Sparkassen, öffentliche rechtliche Rundfunkanstalten) existieren aufgrund Gesetz und sind nicht wie Körperschaften mitgliedschaftlich organisiert sondern besitzen vielmehr Benutzer.
- Stiftungen des öffentlichen Rechts (z.B. Stiftung Preußischer Kulturbesitz, Alexander von Humboldt Stiftung) sind Verwaltungsträger, die weder Mitglieder noch Benutzer hat, sondern Nutznießer. Aufgabe von Stiftungen ist es, übertragenes Vermögen zu verwalten und für einen bestimmten öffentlichen Zweck zu verteilen.

**Verwaltung durch Private**

Die privatrechtliche Verwaltung erfolgt durch Verwaltungsträger, die nicht öffentlich rechtlich sondern privatrechtlich als natürliche oder juristische Personen organisiert sind. Diese können allerdings nur öffentlich rechtliche Aufgaben ausführen, wenn ihnen die Fähigkeit hoheitlich zu handeln übertragen wird. In diesem Fall spricht man von sogenannten Beliehenen [Sod/Zie,S.389]. Eine solche Beleihung kann sich nur auf einzelne Verwaltungsaufgaben beziehen (z.B. TÜV für Autountersuchung, Toll Collect für LKW Mauterhebung[4]) und ist nur durch gesetzliche Grundlage zulässig.

## Mikrostruktur

Ein Verwaltungsträger selbst kann keine Verwaltungstätigkeiten ausführen, sondern es sind die Unterstrukturen und die Personen, die entsprechende Verwaltungsaufgaben

4 Siehe [Weltecke] und www.toll-collect.de/faq/tcrdifr004-5_datenschutz.jsp ;Zugriff:11.9.06

wahrnehmen. Eine Ausnahme stellt die Ausführung durch eine privatrechtlich organisierte natürliche Person dar, da hier der Verwaltungsträger selbst schon eine einzelne Person darstellt und keine kleinere untergeordnete Organisationseinheit existieren kann. In den Fällen der juristischen Personen als Verwaltungsträger existieren als untegeordnete Organisationseinheiten das Organ und der Organwalter [Son/Zie, S.367 ff].

**Organ**

Der Verwaltungsträger schafft innerhalb seiner Institution Organe, denen er Verwaltungsaufgaben zuordnet. Rechtlich gesehen ist ein Organ allerdings nicht selbstständig, sondern handelt allein im Namen des übergeordneten Verwaltungsträgers. Organen werden zwar Verwaltungstätigkeiten übertragen, sind aber selbst nicht handlungsfähig.

**Organwalter**

Organwalter sind Organen zugeordnet, treten als natürliche Personen bspw. in Form von Beamten oder Angestellten auf und führen die dem Organ zugewiesenen Verwaltungsaufgaben aus.

**Beispiel**

Der Verwaltungsträger Rundfunkanstalt als Anstalt des öffentlichen Rechts besitzt die Organe Rundfunkrat, Verwaltungsrat und Intendant/in. Organwalter sind hierbei die konkreten Mitglieder des Rundfunk- und Verwaltungsrats und dem Indentant als Person[5].

**Behörde**

Ein spezielles Organ eines Verwaltungsträgers ist die Behörde. Die Behörde tritt nach außen zum Bürger hin in Erscheinung und erlässt entweder Verwaltungsakte, schließt Verträge ab oder handelt informal [Weltecke].

5 Siehe Abschnitt Organisation www.brandenburg.de/cms/detail.php?id=38783&_siteid=70; Zugriff:11.9.06

# Abkürzungsverzeichnis

| | |
|---|---|
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| BDSG | Bundesdatenschutzgesetz |
| BGB | Bürgerliches Gesetzbuch |
| BSI | Bundesamt für Sicherheit in der Informationstechnik |
| CA | Certificate Authority |
| DDE | Dynamic Data Exchange |
| DES | Data Encryption Standard |
| DSA | Digital Signature Algorithm |
| DSS | Digital Signature Standard |
| GG | Grundgesetz |
| GUI | Graphical User Interface |
| HFTs | Host Function Tables |
| HTTP | HyperText Transfer Protokol |
| HTTPS | HyperText Transfer Protocol Secure |
| JRE | Java Runtime Environment |
| JVM | Java Virtual Machine |
| KoopA ADV | Kooperationsausschuß Automatisierte Datenverarbeitung |
| MDStV | Mediendienstestaatsvertrag |
| NIST | National Institute of Standards and Technology |
| NSA | National Security Agency |
| OLE | Object Linking and Embedding |
| OSCI | Online Services Computer Interface |
| OSI | Open Systems Interconnection |
| PDF | Portable Document Format |
| PGP | Pretty Good Privacy |
| PIN | Personal Identification Number |
| PKCS# | Public-Key Cryptography Standards |
| PKI | Public Key Infrastruktur |
| SAGA | Standards und Architekturen für E-Government Anwendungen |
| SHA | Secure Hash Algorithm |
| SigG | Signaturgesetz |
| SigV | Signaturverordnung |
| SWT | Standard Widget Toolkit |
| TDDSG | Teledienstedatenschutzgesetz |
| TDG | Teledienstegesetz |
| TDSV | Telekommunikations-Datenschutzverordnung |
| TKG | Telekommunikationsgesetz |
| TTP | Trusted Third Party |
| VwVfG | Verwaltungsverfahrensgesetz |
| XHTML | Extensible Hypertext Markup Language |
| XML | Extensible Markup Language |
| XSD | XML Schema Definition |

# Literaturverzeichnis

## Printmedien:


[AcrobatIAC] Adobe Systems; „Acrobat SDK User Guide“; 2004

[AcrobatJS] Adobe Systems; „Acrobat JavaScript Scripting Guide“; 2004

[AcrobatPlug] Adobe Systems; „Acrobat SDK Plug-In Guide“; 2004

[AcrobatSDK] Adobe Systems; „Acrobat Interapplication Communication Overview“; 2004

[Anderson] Anderson, Ross; „Security Engineering"; John Wiley & Sons; 2001

[Arn/Rud] Arndt/Rudolf; „Öffentliches Recht“; Vahlen; 2000

[Be/Fü/Ge] Becker/Fügemann/Gerlach; „Praxis Ausbildung Verwaltungsverfahren und Verwaltungsprozessrecht“; Deutscher Anwaltverlag; 2005

[Becker] Becker,Bernd; „Öffentliche Verwaltung“; R.S Schulz; 1989

[Bertsch] Bertsch,Andreas; „Digitale Signaturen“; Springer; 2001

[BIBB] Elsner,Martin; „Vom regel- und verfahrensorientierten Staatsdiener zum ergebnisorientierten Public Manager“; Bundesinstitut für Berufsbildung; 2004

[Capgemini] Capgemini; „Online Availability of Public Services: How is Europe Progressing?“; 2006

[Choudhury] Choudhury, Surajan; „Public Key Infrastructure, Implementation and Design; Hungry minds; John Wiley & Sons; 2002

[DsgEgov] Konferenz der Datenschutzbeauftragten; „Datenschutzgerechtes eGovernment“; 2002

[Erl] Erl, Thomas; „Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services“; Prentice Hall; 2004

[Ga/He/Jo] Gamma/Helmer/Johnson; „Design Patterns. Elements of Reusable Object-Oriented Software“; 1995

[Gehring] Gehring,Robert; „Digitale Signaturen“; Inf.-Diplomarbeit TU Berlin;1998

[Ipsen] Ipsen,Jörn; „Allgemeines Verwaltungsrecht“; Heymanns Carl; 2000

[Kunstein] Kunstein,Florian; „Die elektronische Signatur als Baustein der elektronischen Verwaltung“; Tenea; 2004

[Liang] Liang,Sheng; „The Java Native Interface“; Addison Wesley; 1999

[Me/Oo/Va] Menezes/Oorschot/Vanstone; „Handbook of applied Cryptography“; CRC Press; 1997

[Opplinger] Opplinger,Rolf; „Contemporary Cryptography“;Artech House; 2005

[OSCILib] OSCI Leitstelle; „Funktionsbeschreibung der OSCI-Bibliothek (Java)“; 2004

[OSCIPrinc] OSCI Leitstelle; „OSCI-Transport 1.2, Entwurfsprinzipien, Sicherheitsziele und Mechanismen“; 2002

[OSCISpez] OSCI Leitstelle; „OSCI-Transport 1.2, Spezifikation“; 2002

[OSCIUeber] OSCI Leitstelle; „OSCI: Sicherheit und Performanz im Internet“; *Herausgabejahr nicht angegeben*

[Palandt] Palandt,Otto; „Bürgerliches Gesetzbuch“; Beck Juristischer Verlag; 2004

[PDFSpec] Adobe Systems; „PDF Reference fifth edition, Version 1.6“; 2004

[Pe/Ma/Ko] Pereira/Martins/Kobylinska; „Adobe Acrobat 7“; Springer; 2006

[Roßnagel] Roßnagel,Alexander; „Digitale Signaturen und e-Government Anwendungen. Notwendige rechtliche Anpassungen“; Universität Kassel und Institut Europäisches Medienrecht Saarbrücken“; 2001

[Sc/Ho/Ng/Mi] Scarpino/Holder/Ng/Mihalkovic; „SWT/JFace in Action“; Manning;2005

[Schneier1] Schneier, Bruce; „Applied Cryptography"; John Wiley & Sons; 1996

[Son/Ble] Sontheimer/Bleek; „Grudnzüge des politischen Systems der Bundesrepublik Deutschland; Bundeszentrale für politische Bildung; 2000

[Sod/Zie] Sodan/Ziewkov; „Grundkurs Öffentliches Recht“; C.H. Beck; 2005

[Tanenbaum] Tanenbaum, Andrew S.; „Computer Networks"; Prentice Hall; 2002

[Til/Bur] Tilly,Burke; „Ant - The Definite Guide“; O'reilly; 2002

# Onlinemedien:

**Hinweis:** Da Hyperlinks oft lange Adressen besitzen, wurden Zeilenumbrüche vorgenommen. Damit vom Text-Prozessor Links erkannt werden konnten und so ein Export auf PDF möglich ist, wurde bei Zeilenumbrüchen von langen Links ein Trennungszeichen '-' nach einem Slash '/' verwendet. Damit der Link bei evtl. Copy/Paste funktioniert, muss dementsprechend das Trennungszeichen entfernt werden.

[AntInstaller] AntInstaller Tool Dokumentation; antinstaller.sourceforge.net/index.html; Zugriff:01.10.2006

[BundOnline] Bundesverwaltungsamt; „Behörden“; www.bund.de/nn_253286/DE/-BuB/Behoerden/Behoerden-knoten.html__nnn; Zugriff: 10.09.2006

[BSI] BSI; „Chefsache E-Government“; www.bsi.de/fachthem/-egov/download/1_Chef.pdf; Zugriff: 10.09.2006

[EditPath] EditPath Tool Dokumentation; www.jsifaq.com/SF/Tips/-Tip.aspx?id=8840; Zugriff:20.10.2006

[EFF] Electronic Frontier Foundation; „Cracking DES“; www.eff.org/Privacy/-Crypto/Crypto_misc/DESCracker; Zugriff: 10.08.2006

[GIKrypt] Gesellschaft für Informatik, Fachgruppe Krypt; „Hashfunktionen offenbar gebrochen,Digitale Signature in Gefahr“; www.gi-ev.de/fachbereiche/sicherheit/fg/krypto/pmSHA1.html; Zugriff: 02.08.2006

[He/Me] Hellman/Merkle; On the Security of Multiple Encryption;Stanford Univ.; cs.purdue.edu/homes/ninghui/courses/Spring04/homeworks/p465-

merkle.pdf ; Zugriff: 10.08.2006

[Heise sec] Wobst/Schmidt;„Hash mich“; Heise security; www.heise.de/security/-artikel/56555; Zugriff: 02.08.2006

[Kerckhoff] Kerckhoff, Auguste; „La cryptographie militaire (1883)"; www.petitcolas.net/kerckhoffs/crypto_militaire_1.pdf; Zugriff: 20.07.2006

[Lehre] Lehre,Lars; „Urkundenstraftaten“; www.jurawelt.com/studenten/-skripten/strafr/1842; Zugriff: 25.09.2006

[OSCI2.0Pr] Dietrich/Steimke; „Entwicklungen von OSCI Transport 2.0“; www1.osci.de/sixcms/media.php/13/2006-11-27-vorlage.pdf; Zugriff: 25.12.2006

[OSCIMedia] MediaKomm Esslingen; „OSCI-FAQ“; osci.mediakomm.esslingen.de/-osci/faq.htm; Zugriff:18.12.2006

[PGP] Network Associates; „PGP – An Introduction To Cryptography“; technology.ohio.edu/email/files/IntroToCrypto.pdf; Zugriff 03.12.2006

[Schneier2] Scheiner, Bruce; Weblog of Bruce Schneier „Cryptoanalysis of SHA-1“; www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html; Zugriff: 02.08.2006

[Schneier3] Scheiner, Bruce; Weblog of Bruce Schneier: „New Cryptologic Results Against SHA-1“; www.schneier.com/blog/archives/2005/08/-new_cryptanalyt.html Zugriff: .08.2006

[Skrobotz] Skrobotz,Jan; „Probleme des elektronischen Verwaltungsakts“; www.jurpc.de/aufsatz/20020086.htm#fn21; Zugriff: 25.09.2006

[SOAP] World Wide Web Consortium (W3C); „Simple Object Access Protocoll (SOAP)1.1; www.w3.org/TR/2000/NOTE-SOAP-20000508; Zugriff: 01.11.2006

[XML] World Wide Web Consortium (W3C); „Extensible Markup Language (XML)“; www.w3.org/XML; Zugriff: 01.11.2006

[Wa/Yi/Yu] Wang/Yin/Yu;„Finding collisions in the full SHA-1“; www.infosec.sdu.edu.cn/paper/sha1-crypto-auth-new-2-yao.pdf; Zugriff: 02.08.2006

[Weltecke] Weltecke,Christoph; Skript Arbeitsgemeinschaft Verwaltungsrecht Uni Marburg; www.students.uni-marburg.de/~Weltecke/; Zugriff: 11.09.2006

[XEnc] World Wide Web Consortium (W3C); „XML Encryption Syntax and Processing“; www.w3.org/TR/2002/CR-xmlenc-core-20020304; Zugriff: 01.11.2006

[XNames] World Wide Web Consortium (W3C); „Namespaces in XML“; www.w3.org/TR/1999/REC-xml-names-19990114; Zugriff: 01.11.2006

[IETF X.509] IETF; „Public-Key Infrastructure (X.509)“; www.ietf.org/html.charters/pkix-charter.html;Zugriff 06.12.2006

[XSig] World Wide Web Consortium (W3C); „XML Signature Syntax and Processing“; www.w3.org/TR/2002/REC-xmldsig-core-20020212; Zugriff: 01.11.2006

[XSD] World Wide Web Consortium (W3C); „XML Schema“; www.w3.org/XML/-Schema; Zugriff: 01.11.2006