Diplomarbeit

Darstellung und Risikoanalyse existierender Single Sign-On Lösungen

Willem Froehling

Matrikelnummer: 4608832

29. März 2005

Betreuer:

Prof. Dr. rer. nat. Klaus Brunnstein
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich AGN

und

Prof. Dr. rer. nat. Joachim Posegga
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich SVS

Danksagung:

Ich möchte mich an dieser Stelle bei meinen Betreuer, Prof. Dr. Klaus Brunnstein, für die ausführliche und kompetente Betreuung bedanken. Ausserdem danke ich ebenfalls meinem Zweitbetreuer, Prof. Dr. Joachim Posegga.

Weiterhin danke ich Petra Arnold und Boris Simons sowie Serge Königsmann für das Korrekturlesen meiner Diplomarbeit.

Markenzeichen:

Arbeitsmittel:

Die vorliegende Diplomarbeit wurde unter dem Betriebssystem GNU/Linux erstellt. Als Textverarbeitungssystem wurde LaTeX 2_ε in der teTeX Version 2.0.2 verwendet. Zur Eingabe und Verwaltung des Dokuments wurde das Programm Kile aus der KDE Suite benutzt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Einleitung

Das Identifizieren und Authentisieren von Objekten stellt einen wichtigen Aspekt innerhalb verteilter Systeme dar, z.B. in privaten, öffentlichen oder Firmen Netzwerken. In einer ungeschützten Netzwerkumgebung kann jeder Client jeden Server/Dienst benutzen. Dies kann sehr leicht zu Betrugsmöglichkeiten und ungewollter Offenlegung von Informationen führen. Können sich die Kommunikationspartner, entweder einseitig oder beidseitig, identifizieren und authentisieren, böte ein solches System mehr Sicherheit. Häufig werden dabei Benutzeridentifikator-Passwort Paare verwendet. Es hat sich gezeigt, dass ein normaler Anwender nur eine geringe Anzahl von solchen Kombinationen auswendig weiss. Je mehr solcher Kombinationen der Anwender verwalten muss, desto häufiger leidet die Sicherheit des Systems. Oft werden solche Paare dann niedergeschrieben und irgendwo am Rechner aufbewahrt, oder es werden immer wieder die gleichen Kombinationen bei den unterschiedlichen Diensten benutzt. Abhilfe sollen hier Single Sign-On Systeme schaffen. Diese Systeme erlauben es dem Anwender sich nur an einer Stelle anzumelden, um anschließend ohne eine erneute Anmeldung eine Vielzahl von unterschiedlichen Diensten zu nutzen.

1.2 Motivation

Single Sign-On Verfahren haben sich in geschlossenen Firmen oder Behörden Netzwerken etabliert. Viele aktuelle Betriebssysteme bringen schon Fähigkeiten für ein eigenes Single Sign-On mit. Doch wie sieht eine solche Realisierung in einem offenen Verbund unterschiedlicher Netzwerke wie dem vielgenutzten Internet aus? Durch die fortschreitende Kommerzialisierung des Internets hat ein Benutzer heute eine Vielzahl unterschiedlicher Zugangszeugnisse zu verwalten. Es wird erwartet, dass ein erfolgreiches Single Sign-On Verfahren dem Benutzer den Zugang vereinfachen kann und den Handels- und Informationsmärkten einen Auftrieb beschert.

Zwei aktuelle und zur Zeit stark beworbene Single Sign-On Systeme für das kommerzielle Internet sind das Microsoft Passport System und das Liberty Alliance Verfahren. Doch wie sieht es mit der Sicherheit und dem Datenschutz, besonders aus der Sicht des Anwenders, bei diesen Systemen aus? Diese Fragestellung soll in dieser Diplomarbeit untersucht werden. Denn nur wenn die Anwender Vertrauen in das angebotene System haben, werden diese es auch in großem Maße nutzen.

1.3 Ausblick auf die Diplomarbeit

Die Diplomarbeit behandelt zunächst relevante Grundlagen, welche zum Verständnis der Hauptarbeit benötigt werden. Anschliessend folgt die Vorstellung der untersuchten Systeme. Zunächst werden unterschiedliche Single Sign-On Verfahren in vier verschiedene Kategorien eingeteilt. Es folgt die Darstellung der Funktionsweise eines der ersten und vielfach in geschlossenen Netzwerken eingesetzen Single Sign-On Verfahren mit Namen Kerberos. Daneben werden das Microsoft Passport System und das Liberty Alliance Verfahren vorgestellt.

Das Microsoft Passport System und das Liberty Alliance Verfahren werden dann einer Risikoanalyse unterzogen. Hierbei muss erwähnt werden, dass bei beiden Systemen nur eine sogenannte „Postfunktionale Risikoanalyse“ sinnvoll ist. Da diese beiden Verfahren entweder fertig spezifiziert[1] sind oder als fertiges

[1] Dies betrifft zumindest den in dieser Arbeit zu untersuchenden Teil.

Produkt vorliegen, können keine Änderungen an Systemparameter oder Systemobjekten vorgenommen werden. Im Unterschied dazu existiert die „Präventive Risikoanalyse“, die während der Entwicklung von Systemen zum Einsatz kommt und die bei Ihren Ergebnissen noch Einfluss auf das System und seine Systemumgebung nehmen kann. Innerhalb der „Postfunktionalen Risikoanalyse“ wird sich diese Diplomarbeit hauptsächlich auf die Schwachstellenanalyse konzentrieren.

Kapitel 2

Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen zum besseren Verständnis der Hauptarbeit vorgestellt. Ein Teil dieses Inhalts ist meiner Studienarbeit „Konzept und exemplarische Implementation eines gesicherten Kanals zur Übertragung biometrischer Daten“ [Fro03] entnommen.

2.1 Notwendigkeit von Kommunikationsprotokollen

Zunächst soll der Begriff Kommunikation definiert werden.

Kommunikation: Austausch von Information. Meist legt man fünf „Stationen“ für die Übertragung einer Information fest; Sender, Codierung, Übertragungsmedium (*Kanal*), Decodierung und Empfänger. [Lek93]

Bei Kommunikationen unterscheidet man verschiedene Formen und Ebenen.
Formen: Die Informationen fließen in nur einer (*unidirektional*), in zwei (*bidirektional*) oder in viele Richtungen; sie können gezielt an einen Empfänger (wie im Gespräch zweier Personen), an festgelegte Empfänger (wie bei einem Vortrag oder bei der Einladung zu einer Sitzung) oder anonym an beliebig viele angeschlossene Empfänger (wie bei Funk und Fernsehen) gerichtet sein.
Ebenen: Kommunikation kann sich auf einer rein physikalischen Ebene, auf den darüberliegenden Schichten des ISO/OSI-Schichtenmodells oder auf einer semantischen Ebene abspielen.

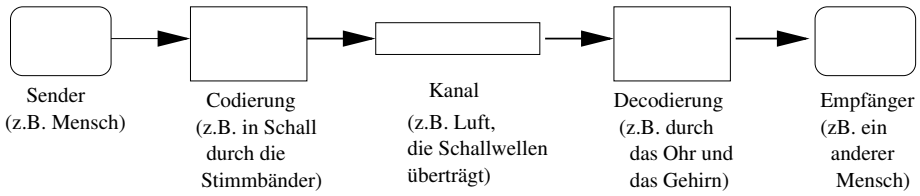

Abbildung 2.1: Kommunikation

Damit zwischen zwei Kommunikationspartnern ein Austausch von Information stattfinden kann, benötigt man ein Kommunikationsprotokoll.

Protokoll: Vereinbarung über den geordneten Ablauf einer Kommunikation, wobei die Vereinbarung in der Informatik diktatorischen Charakter besitzt: Wer sich nicht an sie hält, wird von der Kommunikation ausgeschlossen. Entsprechend der aufeinander aufbauenden Schichten bei der Übertragung von Informationen wird für jede Ebene ein eigenes Protokoll vereinbart, dessen Realisierung sich auf das Protokoll der nächst tieferen Ebene stützt. Protokolle sind bei der Kopplung von Systemen und in „Offenen Systemen“ (bisher) unverzichtbar.
Für die präzise Definition von Protokollen eignen sich endliche Automaten oder Petri-Netze. Hierbei nimmt man an, dass sich jeder Kommunikationspartner und das übertragende Medium in Zuständen befinden, die man zu einem Gesamtzustand zusammenfasst. [Lek93]

In einem Kommunikationsprotokoll wird also exakt definiert, wie ein Datenaustausch zwischen zwei Parteien auszusehen hat. Das Protokoll legt Format, Bedeutung und Reihenfolge der zu übertragenden Nachrichten fest. Somit sind die Stationen der Kommunikation genau festgelegt, und beide Parteien können die Informationen verwerten, die sie zuvor ausgetauscht haben.

2.2 Netz Architektur und zugrundeliegende Standards

Da es in dieser Arbeit um spezielle Kommunikationskanäle geht, verwirklicht über das Netzwerksystem TCP/IP, wird dieses zugrundeliegende Protokoll genauer

erläutert werden. Doch zunächst wird ein allgemeineres Modell erklärt und in dieses anschließend die TCP/IP Protokollfamilie eingebettet.

2.2.1 ISO/OSI

Zur Beschreibung der Struktur und Funktion von Protokollen für die Datenkommunikation wird häufig ein Architekturmodell herangezogen. Zum Vergleich von unterschiedlichen Netzwerkprotokollen hat sich das ISO/OSI-Referenzmodell[1] bewährt. Das Modell unterteilt die Netzkommunikation in sieben Schichten (layers). Jede Schicht des OSI-Modells repräsentiert eine Funktion, die beim Austausch von Daten zwischen Anwendungen über ein dazwischenliegendes Netzwerk hinweg ausgeführt wird. Diese Art der Darstellung erscheint wie ein „Haufen von Ziegelsteinen", die übereinander gestapelt wurden - deshalb spricht man oft von einem *Stack* (Stapel) oder *Protokollstack*.

Schicht	**Schichtname**	**Beschreibung**
7	Anwendungs-schicht	besteht aus den Anwendungen, mit denen man das Netz nutzen kann
6	Darstellungs-schicht	standardisiert das Format der Daten auf dem Netz
5	Kommunikations-schicht	verwaltet die Verbindung zwischen den Anwendungen
4	Transport-schicht	garantiert die fehlerfreie Datenübertragung durch Fehlererkennung und -korrektur
3	Vermittlungs-schicht	verwaltet die Verbindungen zwischen den Rechnern im Netz für die höheren Schichten
2	Sicherungs-schicht	sorgt für die zuverlässige Übertragung der Daten über die physikalische Verbindung
1	Bitübertragungs-schicht	definiert die physikalischen Eigenschaften der Übertragung

Tabelle 2.1: ISO/OSI Schichtenmodell

[1] Architekturmodell der Internationalen Standard Organisation (ISO) für offene Netze (Open System Interconnection, OSI)

Anwendungsschicht: Die Anwendungsschicht, die höchste Schicht des Referenzmodells, stellt die Mittel zur Kooperation zwischen verteilten Anwendungsprozessen zur Verfügung. Der Anwendungsprozess hat ausschließlich über die Schicht sieben Zugang zum OSI-Netzwerk.

Darstellungsschicht: Die Darstellungsschicht stellt Ausdrucksmittel (z.B. Zeichenvorrat, Datentypen) zur Verfügung, die es den Anwendungsinstanzen ermöglichen, Begriffe eindeutig zu benennen, und legt im Darstellungsprotokoll die Regeln fest, wie die in der gemeinsamen Sprache dargestellte Information auszutauschen ist.

Kommunikationssteuerungsschicht: Die Kommunikationssteuerungsschicht regelt den Gesprächswechsel zwischen kommunizierenden Partnern und liefert ab einem zwischen den Partnern vorher vereinbarten Punkt im Datenstrom (Synchronisationspunkt) Vorkehrungen zum Wiederanlaufen einer unterbrochenen Übertragung. Sie regelt den Betriebsablauf.

Transportschicht: Die Transportschicht nimmt die Anforderung der Anwenderprozesse hinsichtlich der Übertragungsqualität entgegen, erstellt entsprechende Aufträge an die darunterliegenden Schichten und gleicht gegebenenfalls ungenügende Leistungen dieser Schichten aus.

Vermittlungsschicht: Die Vermittlungsschicht transportiert Pakete über die Teilstrecken des Netzes von Endsystem zu Endsystem.

Sicherungsschicht: Die Sicherungsschicht verbessert ungesicherte Verbindungen auf Teilstrecken zu gesicherten Verbindungen. Gesichert heißt dabei eine Verbindung, die Fehler in der Datenübertragung erkennt und korrigiert.

Bitübertragungsschicht: Die Bitübertragungsschicht stellt ungesicherte Verbindungen zwischen Systemen für die Übertragung von Bits zur Verfügung.

Jede einzelne Schicht definiert nicht etwa ein Protokoll, sondern sie stellt vielmehr eine Funktionalität der Datenkommunikation dar, die von beliebig vielen Protokollen ausgeführt werden kann. Jede Schicht kann also mehrere Protokolle enthalten, von denen jedes solche Dienste bereitstellt, wie sie für die Erfüllung

der Funktion dieser Schicht benötigt werden. Jedes Protokoll kommuniziert mit seinem Peer (Gegenüber). Ein Peer ist die Implementierung desselben Protokolls in der entsprechenden Schicht des Kommunikationspartners.

2.2.2 TCP/IP

Der Name TCP/IP bezeichnet eine ganze Reihe von Protokollen für die Datenkommunikation. Diese Protokollfamilie bezieht ihren Namen von zweien der enthaltenen Protokolle - dem Transmission Control Protocol (TCP) und dem Internet Protocol (IP). Obwohl noch einige andere Protokolle enthalten sind, sind TCP und IP sicherlich zwei der wichtigsten. In diesem Text wird nur auf die noch aktuelle Version 4 von TCP/IP eingegangen.

TCP/IP wurde für das ARPANET (Advanced Research Projects Agency), dem Vorgänger des heutigen Internets, entwickelt. Das ARPANET war in seiner ursprünglichen Nutzung ein rein militärisches Netz, das zur Koordination von weiträumig verteilten militärischen, behördlichen und wissenschaftlichen Einrichtungen dienen sollte. Die Entwicklung der TCP/IP-Architektur, welche die bis dahin verwendeten Protokolle ablösen sollte, begann etwa 1978, 1981 wurde TCP/IP im RFC[2] 793 standardisiert. Im Jahre 1983 wurden die TCP/IP-Protokolle zu „Military Standards “ (MIL STD) erhoben und alle Institutionen, die am ARPANET partizipierten, mussten auf die neuen Protokolle umstellen. Eine der Hauptvorgaben bei der Entwicklung des ARPANETs und der damit verbundenen Protokolle war, dass auch beim Ausfall einiger Knoten (z.B. durch äußere Einwirkung des Feindes) das restliche Netz weiterhin funktionsfähig bleiben sollte.

Aus dieser Vorgabe resultieren einige Eigenschaften des TCP/IP-Modells.

- Offene Protokollspezifikationen, die frei zugänglich und unabhängig von der Hardware und dem Betriebssystem sind. Aufgrund seiner weitreichenden

[2]RFC steht für: request for comments. Die RFCs sind Arbeitspapiere, Protokollspezifikationen oder auch nur Kommentare zu aktuellen Themen der Internet Community. RFCs werden von der Internet Engineering Task Force (IETF) erstellt und von der Internet Engineering Steering Group (IESG) formell abgesegnet. Siehe auch `http://www.ietf.org`

Unterstützung eignet sich TCP/IP in idealer Weise dazu, unterschiedliche Hardware und Software miteinander zu verbinden.

- Unabhängigkeit von einer bestimmten Netzwerkhardware. So ist es möglich, mittels TCP/IP viele unterschiedliche Netze miteinander zu verbinden. TCP/IP kann über ein Ethernet, einen Token Ring, eine Wählleitung, ein X.25 Netz und beinahe jedes andere physikalische Übertragungsmedium betrieben werden.

- Ein einheitliches Adressierungsschema, das es jedem Rechner in einem TCP/IP-Netz ermöglicht, jeden beliebigen anderen Rechner in diesem Netz eindeutig zu identifizieren.

- Standardisierte Protokolle der höheren Schichten, die dem Benutzer einen einheitlichen und weithin verfügbaren Dienst zur Verfügung stellen.

TCP/IP lässt sich in vier Schichten einteilen.

Schichtebene	Schichtname	Beschreibung
4	Anwendungsschicht	enthält Anwendungen und Prozesse die auf das Netzwerk zugreifen
3	Transportschicht	stellt End-zu-End Datendienste zur Verfügung
2	Internetschicht	definiert den Aufbau von Datagrammen und routet Daten
1	Netzzugangsschicht	enthält Routinen für den Zugriff auf physikalische Netze

Tabelle 2.2: IP Schichtenmodell

Wie man sieht, besteht der Protokollstack von TCP/IP nicht aus sieben, sondern nur aus vier Schichten. Dieses Modell mit vier Schichten basiert auf den drei Ebenen, die im Protocol Model des DOD (US Verteidigungsministerium) im *DDN Protocol Handbook* beschrieben werden.

Es fehlen die Darstellungsschicht und die Kommunikationssteuerungsschicht aus dem OSI-Modell. Die Kommunikationssteuerungsschicht ist bei TCP/IP hauptsächlich in der Transportschicht untergebracht. Es exisitieren auch keine *Sessions* wie bei OSI, sondern Verbindungen werden bei TCP/IP über *sockets* und *ports* referenziert und somit die zuständigen Applikationsendpunkte festgelegt. Falls Applikationen bei TCP/IP zusätzliche Kommunikationssteuerungsdienste benötigen, müssen diese auf Applikationsebene erbracht werden. Ein Beispiel sei „Network File System“ (NFS), welches seine eigene Kommunikationssteuerung mitbringt, die „Remote Procedure Call“ (RPC).
Auch die Darstellungsschicht fehlt bei TCP/IP. Diese Funktionalität liegt auf Applikationsebene. Ein Beispiel sei „Multipurpose Internet Mail Extensions“ (MIME).

Bei TCP/IP teilt sich die Tranportschicht vertikal auf in „User Datagram Protocol“ (UDP) und „Transmission Control Protocol“ (TCP).

UDP: *UDP* baut einen verbindungslosen Kommunikationskanal auf. Pakete können verloren gehen, sich verdoppeln oder korrupt sein. Die darunter liegende Netzwerkschicht behandelt nämlich jedes Paket wie eine eigene Einheit und deswegen kann jedes Datenpaket bei einem paketvermittelnden Netz, wie IP, unterschiedliche Netzrouten nehmen. Wenn bei der Übertragung von Daten ein hoher Durchsatz erforderlich ist, wobei nicht jedes Datum relevant ist, dann nutzt man meist den *UDP* Dienst.

TCP: *TCP* ist für Verbindungen gedacht, die verlässlich bei der Übertragung der Daten arbeiten sollen. *TCP* ist zuständig für die Datenwiederherstellung von verlorenen, korrupten oder auch in falscher Reihenfolge erhaltenen Datenpaketen. Dies gewährleistet *TCP* durch Sequenznummern in jedem Datenpaket. TCP erfordert, dass jedes übertragene Datenpaket quittiert wird.

TCP und UDP kommunizieren mit dem Konzept der Ports. Ein Port ist eine virtuelle Schnittstelle, die auf einer Netzschnittstelle geöffnet werden kann. Viele Ports sind bei TCP/IP mit einem zuständigen Dienst assoziiert. Über einen Port kann man quasi bestimmen, welche Applikation die Daten erhalten sollen. Durch

Ports besteht die Möglichkeit zu multiplexieren, also mehrere paralelle Verbindungen zu öffnen.

Die Portnummern zusammen mit der Quell- und Zieladresse spezifizieren einen Socket. Jede Maschine, die über TCP/IP kommuniziert, wird ein Socket zu der empfangenden Maschine hin öffnen. Sind die Sockets verbunden, steht bei TCP ein verlässlicher Dienst zu Verfügung. Eine Applikation kann mehrere Sockets öffnen und über diese parallel kommunizieren.

Die Internetschicht bei TCP/IP ist fast identisch zu der Vermittlungsschicht vom OSI-Modell. Ein Unterschied ist, dass bei IP nur verbindungslose Kanäle existieren. Die Vermittlungsschicht ist zuständig für den Weg, die die Pakete von einem Kommunikationspartner zum anderen nehmen. Dabei wird für jedes Datenpaket unabhängig entschieden, welche Route dieses durch das Netz nehmen soll.
Ein weiterer Unterschied zu OSI ist, dass IP eine feste Länge für die Adresse (32 Bit) verlangt.

Die Vermittlungsschicht ist dafür zuständig, die Daten, die sie von der Transportschicht erhalten hat, in Pakete mit bestimmten Längen zu unterteilen, die den Beschränkungen der physikalischen Schicht Rechnung tragen (Fragmentierung).

Über die Sicherungsschicht und Bitübertragungsschicht wird bei TCP/IP nichts weiter erwähnt, da TCP/IP auf unterschiedlichen darunterliegenden Netzwerktypen aufgebaut ist. Es verlangt nur irgendein Protokoll, welches dem Rechner ermöglicht irgendwelche IP-Pakete zu verschicken. Da also die unteren Protokolle nicht näher spezifiziert sind, variieren diese von Rechner zu Rechner und Netzwerk zu Netzwerk.

Im Folgenden wird die zugrundeliegende Netz Architektur zusammengefasst als Grafik dargestellt:

Schicht	ISO/OSI	TCP/IP
7	Anwendungsschicht	Andwendungsschicht
6	Darstellungsschicht	
5	Kommunikationsschicht	
4	Transportschicht	Transportschicht
3	Vermittlungsschicht	Internet-Schicht
2	Sicherungsschicht	Netzzugangsschicht
1	Bitübertragungsschicht	

Tabelle 2.3: TCP/IP-ISO/OSI

Genau wie im OSI-Modell werden die Daten im Stack nach unten weitergereicht, wenn Daten verschickt werden. Beim Empfang verläuft der Weg genau umgekehrt, von Netzzugangsschicht zu der Anwendungsschicht. Jede Schicht fügt ihre eigenen Kontrollinformationen hinzu. Diese Information nennt man *Header* (Kopf), da sie den eigentlichen Daten vorangestellt wird. Jede Schicht betrachtet die gesamte Information, die sie von der darüberliegenden Schicht empfängt, als zu übertragende Daten. Beim Empfangen von Paketen wird der umgekehrte Weg genommen. Jede Schicht trennt ihren Header von dem Datenpaket ab und reicht, falls keine Fehler aufgetaucht sind, den restlichen Datenteil eine Schicht höher.

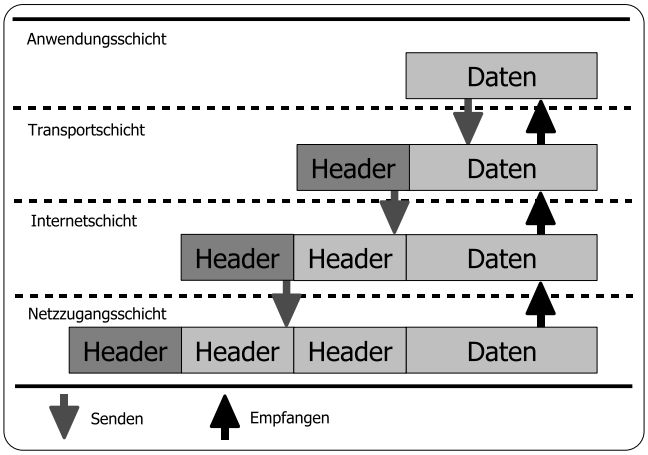

Abbildung 2.2: IP-Paketschachtelung

2.2.3 SSL/TLS

Die „Secure Sockets Layer“ (**SSL**) Protokollfamilie ist ursprünglich bei der Firma Netscape entstanden. Die aktuelle Version 3 des SSL-Protokolls ist auch als „Internet Draft“ veröffentlicht [FKK96], und die IETF hat daraus den allgemeinen Standard TLS 1.0 [DA99] entworfen, auch bekannt als „Transport Layer Security“ (**TLS**). Diese erste Version von TLS kann als SSLv3.1 angesehen werden und ist sehr eng an SSLv3 angelehnt. Deswegen wird hier im Allgemeinen SSL sowohl für SSLv3, als auch synonym für TLS benutzt. Am Ende dieses Unterkapitels werden kurz die Unterschiede zwischen SSLv3 und TLS erläutert.

SSL Architektur

SSL ist eine Protokollschicht, welche sich direkt über die Transportschicht im TCP/IP-Protokollstapel setzt und sich somit unter der Applikationsschicht befindet. Dieses Design hat den Vorteil, dass SSL unabhängig von dem verwendeten Applikationsprotokoll eine sichere Punkt-zu-Punkt Verbindung herstellen kann. SSL besteht nicht aus einem einzelnen Protokoll, sondern besitzt zwei Protokollschichten.

SSL-Handshake Protocol	SSL-Change Cipher Spec Protocol	SSL-Alert Protocol	Anwendungs-protokoll z.B HTTP
SSL-Record Protocol			
TCP			
IP			

Tabelle 2.4: SSL Protokollstack

Zwei wichtige Konzepte von SSL sind die SSL-Session und die SSL-Connection.

SSL-Session Eine SSL-Session ist eine Assoziation zwischen einem Client und einem Server. Sessions werden über das Handshake Protokoll aufgebaut. Eine Session definiert eine Menge von kryptologischen Sicherheitsparametern, welche gemeinsam über mehrere Verbindungen genutzt werden können. Der Sinn von Session besteht darin, nicht jedesmal eine neue zeitaufwendige Verhandlung der Sicherheitsparameter auszuführen.

SSL-Connection Eine SSL-Connection ist ein Transportkanal, der einen spezifizierten Service bietet. Bei SSL handelt es sich um eine Peer-to-Peer Verbindung, welche nur vorübergehend aufgebaut sein kann. Jede Connection muss mit einer Session assoziiert sein.

Somit können aus einer Session mehrere Connections aufgebaut werden, die auch parallel betrieben werden können.

SSL-Sessions besitzen unterschiedliche Zustände. Ein Sessionzustand ist definiert über folgende Parameter [FKK96]:

- **Session Identifier**: Eine zufällige Bytesequenz vom Server erzeugt, um eine aktive oder wiederherstellbare Session zu identifizieren.
- **Peer certificate**: Ein X509.v3 Zertifikat des Clients. Dieses Element kann Null sein.
- **Compression method**: Der Kompressionsalgorithmus, der vor einer Verschlüsselung eingesetzt wird.
- **Cipher spec**: Spezifiziert den Verschlüsselungsalgorithmus, sowie den Hashalgorithmus für den MAC. Zusätzlich werden noch Attribute, wie z.B Hashlänge, definiert.
- **Master secret**: 48 Byte Geheimnis, das Client und Server kennen müssen.
- **Is resumable**: Ein Flag, welches anzeigt, ob diese Session für neue Verbindungen genutzt werden kann.

Auch die Connections definieren unterschiedliche Zustände:

- **Server and client random**: Byte Sequenzen, die vom Server und Client für jede Verbindung erzeugt werden.
- **Server write MAC secret**: Der geheime Schlüssel, der bei MAC-Operationen von Daten, die zum Server gehen, benutzt wird.
- **Client write MAC secret**: Der geheime Schlüssel, der bei MAC-Operationen von Daten, die zum Client gehen, verwendet wird.

- **Server write key**: Der symmetrische Schlüssel, der zur Verschlüsselung beim Server und bei der Entschlüsselung beim Client angewendet wird.
- **Client write key**: Der symmetrische Schlüssel, der zur Verschlüsselung beim Client und bei der Entschlüsselung beim Server benutzt wird.
- **Initialization vectors**: Wenn ein Blockchiffre im CBC-Mode benutzt wird, so kommt ein „initialization vector" (IV) für jede Verschlüsselung zum Einsatz. Dieses Feld wird zum ersten Mal beim SSL-Handshake Protokoll initialisiert. Danach wird der letzte Ciphertextblock von jedem Record als IV für den nächsten Record benutzt.
- **Sequence numbers**: Beide Parteien verwalten eigene Sequenznummern für die gesendeten und empfangenen Nachrichten jeder Verbindung. Falls eine der Parteien ein „Change cipher spec" verschickt oder erhält, so wird die Sequenznummer wieder auf Zero gesetzt. Sequenznummern können nicht größer als $2^{64} - 1$ werden.

SSL-Record Protocol

Das *SSL-Record Protocol* stellt zwei Dienste für SSL-Verbindungen zur Verfügung:

- **Vertraulichkeit**: Das Handshake Protokoll handelt einen gemeinsamen geheimen Schlüssel aus, der für die symmetrische Verschlüsselung der SSL Nutzdaten angewendet wird.
- **Nachrichten Integrität**: Das Handschake Protokoll definiert zudem auch einen gemeinsamen geheimen Schlüssel, der für den „Message Authentication Code" (MAC) benutzt wird.

Das *SSL-Record Protocol* nimmt eine für die Übertragung vorgesehene Applikationsnachricht an und erstellt daraus ein SSL-Paket. Zunächst wird das Paket fragmentiert und in die notwendige Blockgröße zerlegt/aufgefüllt. Optional wird dann dieser Block komprimiert. Anschließend wird ein MAC angehängt. Dieser neue Block wird verschlüsselt und ein SSL-Header vorangestellt. Danach wird dieser Block dem TCP-Stack übergeben. Der Empfänger entschlüsselt, verifiziert

und dekomprimiert diesen Block. Anschließend setzt er aus den anderen Teilen wieder den ursprünglichen Block zusammen und übergibt ihn dann an die Applikationsebene.

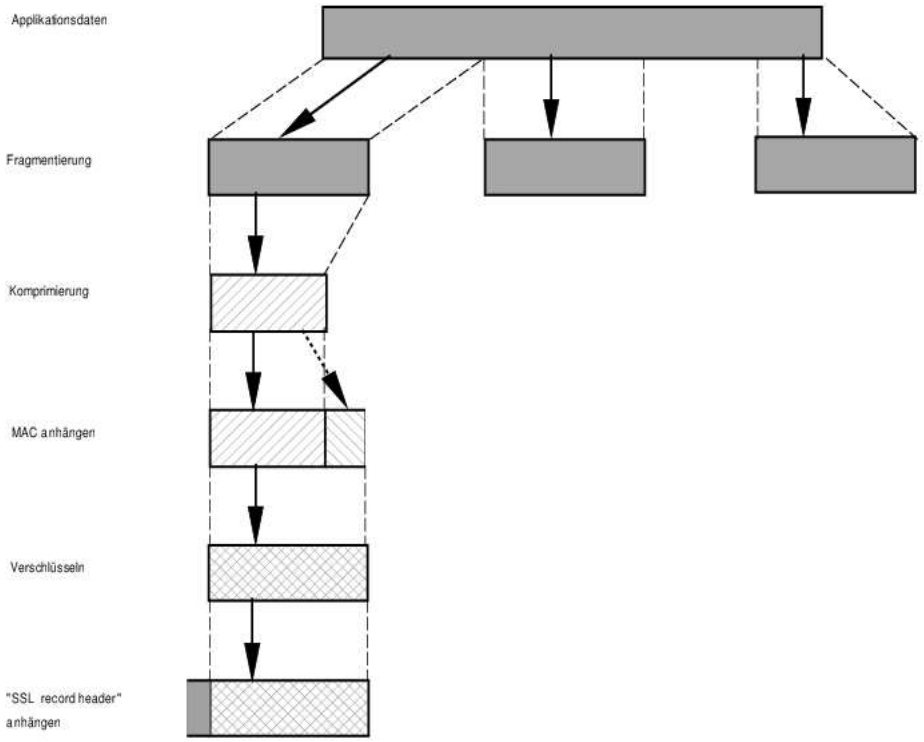

Abbildung 2.3: SSL-Paket erstellen

Der erste Schritt ist die Fragmentierung. Jedes Fragment hat eine Größe von 2^{14} Bytes oder kleiner. Optional wird dieser Block komprimiert. Der verwendete Kompressionsalgorithmus muss verlustlos sein und darf die Nachricht nicht auf mehr als 1024 Bytes vergrössern[3]. Wenn bei SSLv3 (wie auch bei TLS) kein Kompressionsalgorithmus gewählt wurde, wird der Block nicht komprimiert. Der nächste Schritt ist das Berechnen des „*message authentication code*“ (MAC) und dann das Anhängen desselben an den Datenblock. Dafür wird ein gemeinsamer geheimer Schlüssel benötigt. Der verwendete Algorithmus ist dem des HMAC

[3]Eigentlich sollte ein Komprimierungsalgorithmus den Block verkleinern. Bei sehr kleinen Blöcken, kann es wegen Formatierungskonventionen vorkommen, dass diese anschliessend grösser sind.

sehr ähnlich.

$$hash(MAC.write.secret \| pad2 \| hash(MAC.write.secret \| pad1 \| seq.num \| SSLCompressed.type \| SSLCompressed.length \| SSLCompressed.fragment))$$

$\|$ = Konkatenation
$MAC.write.secret$ = gemeinsamer geheimer Schlüssel
$hash$ = Hashalgorithms, entweder MD5 oder SHA-1
$pad1$ = Das Byte 0x36 48 mal wiederholt für MD5 oder 40 mal für SHA-1
$pad2$ = Das Byte 0x5c 48 mal wiederholt für MD5 oder 40 mal für SHA-1
$seq.num$ = Die Sequenznummer für diese Nachricht
$SSLCompressed.type$ = Das nächst höhere Protokoll,welches benutzt wurde, um dieses Fragment zu bearbeiten
$SSLCompressed.length$ = Die Länge des Fragments
$SSLCompressed.fragment$ = Das komprimierte Fragment, oder falls keine Kompression der Daten gewählt wurde, der Klartext.

Anschließend wird die komprimierte Nachricht einschließlich des MACs mit einem symmetrischen Schlüssel verschlüsselt. Wobei die verschlüsselte Nachricht nicht größer als 1024 Bytes werden darf. Somit wird die Gesamtlänge nicht größer als $2^{14} + 1024$ Bytes.

Folgende Verschlüsselungsalgorithmen dürfen hierfür verwendet werden:

	Blockchiffren		Stromchiffren
Algorithmus	Schlüsselgrösse	Algorithmus	Schlüsselgrösse
IDEA	128	RC4-40	40
RC2-40	40	RC4-128	128
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Tabelle 2.5: SSL-Verschlüsselungsalgorithmen

Zum Abschluss wird im *SSL-Record Protocol* noch ein Header der Nachricht vorangestellt, der folgende Felder hat:

- **Content Type (8 bits)**: Das höhere Protokoll, das für die Verarbeitung verwendet wird. Es handelt sich nicht um das Protokoll aus der Applikationsebene, sondern um eines aus *change_cipher_spec*-, *alert*-,

hand_shake- und *application_data-protocol*. Das Anwendungsprotokoll ist transparent für SSL.

- **Major Version (8 bits)**: Kennzeichnet die Version des verwendeten SSL-Protokolls. Bei SSLv3 ist der Wert 3.
- **Minor Version (8 bits)**: Kennzeichnet die Unterversion. Bei SSLv3 ist der Wert 0.
- **Compressed length (16 bits)**: Die Länge der Klartextnachricht in Bytes. Bei Komprimierung ist es die Länge der komprimierten Klartextnachricht. Der maximale Wert darf nicht grösser als $2^{14} + 2048$ sein.

Content Type	Major Version	Minor Version	Compressed Length
Plaintext (optional komprimiert) verschlüsselt			
MAC (0,16 oder 20 Bytes) verschlüsselt			

Tabelle 2.6: SSL Record Format

Change Cipher Spec Protocol

Das *Change Cipher Spec Protocol* ist eines von drei SSL-spezifizierten Protokollen, das das darunterliegende *SSL Record Protocol* benutzt. Dieses Protokoll besteht nur aus einer einzigen Nachricht mit nur einem Byte. Dieses Byte hat den Wert 1. Der Sinn dieses Protokolls ist, einen wartenden Zustand in einen aktuellen Zustand zu kopieren. Das hat zur Folge, dass die Chiffre Suite aktualisiert wird.

Alert Protocol

Das *Alert Protocol* dient zum Übertragen von SSL-Fehlermeldungen. Wie bei Applikationen, die SSL verwenden, werden diese Nachrichten komprimiert und verschlüsselt. Jede Nachricht aus diesem Protokoll besteht aus 2 Bytes. Das erste Byte kann den Wert warning (1) oder fatal (2) haben und gibt den Grad der Fehlermeldung an. Wird der Wert fatal übertragen, beendet SSL sofort die Verbindung. Andere Verbindungen aus der Session bleiben bestehen, aber es kann

keine neue Verbindung erzeugt werden. Das zweite Byte beschreibt den Fehler genauer.

Fatale Alertmeldungen:

- **unexpected_message**: Ungültige Nachricht erhalten.
- **bad_record_mac**: Ungültige MAC erhalten.
- **decompression_failure**: Die Kompressionsfunktion hat eine ungültige Eingabe erhalten.
- **handshake_failure**: Der Sender konnte keine Vereinbarung über die geforderten Sicherheitsparameter treffen.
- **illegal_parameter**: Ein Feld in einer *handshake* Nachricht ist ungültig oder inkonsistent mit anderen Feldern.

Warnende Alertmeldungen:

- **close_notify**: Der Sender dieser Nachricht wird keine Nachrichten mehr schicken.
- **no_certificate**: Antwort auf ein *certificate_request*, falls keines vorhanden ist.
- **bad_certificate**: Ein korruptes Zertifikat erhalten.
- **unsupported_certificate**: Typ des erhaltenen Zertifikats wird nicht unterstützt.
- **certificate_revoked**: Das Zertifikat wurde von seinem Signierer widerrufen.
- **certificate_expired**: Das Zertifikat ist abgelaufen.
- **certificate_unkown**: Sonstiger Fehler ist beim Validieren des Zertifikats aufgetreten.

SSL-Handshake Protocol

Das komplexeste Protokoll bei SSL ist das *SSL-Handshake Protocol*. Dieses Protokoll dient zum Authentisieren des Servers und des Clients, zum Aushandeln eines Verschlüsselungs- und MAC-Algorithmus, sowie zum Austauschen der kryptographischen Schlüssel, die der Sicherung der Daten dienen. Dieses Protokoll tauscht unterschiedliche Nachrichten zwischen Client und Server aus, die alle den gleichen Aufbau besitzen.

- Type: (1Byte) Zeigt eine von zehn möglichen Nachrichten an.
- Length: (3 Bytes) Enthält die Länge der Nachricht in Bytes.
- Content: ($\geq$ 1 Byte) Parameter, die mit dieser Nachricht assoziert sind.

Es folgt eine Tabelle mit den unterschiedlichen Nachrichtentypen.

Nachrichtentyp	**Parameter**
hello_request	null
client_hello	version, random, session_id, cipher suite, compression method
server_hello	version, random, session_id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Tabelle 2.7: SSL-Handshake Protocol: Nachrichtentypen

Das *SSL-Handshake Protocol* kann in vier Phasen aufgeteilt werden.

Phase 1: Erstellung der Sicherheitsfähigkeiten

Diese Phase dient zum Initiieren einer logischen Verbindung und zum Aufbau der Sicherheitsmöglichkeiten, die mit der Verbindung assoziiert werden sollen. Dieser Austausch wird vom Client gestartet, indem dieser eine *client_hello* Nachricht an den Server schickt mit folgenden Parametern:

- **Version**: Die höchste Version von SSL, die der Client versteht.
- **Random**: Eine auf dem Client generierte Zufallsstruktur, bestehend aus einem 32 Bit Zeitstempel und 28 zufällig generierten Bytes. Diese Werte dienen zum Schlüsselaustausch und verhindern Replay Attacken.
- **Session ID**: Ein Sitzungsbezeichner von variabler Länge. Wenn dieser Wert nicht 0 ist, dann kennzeichnet der Client hiermit, dass er die Parameter einer bestehenden Session verändern möchte oder eine neue Verbindung in der Session starten will. Ist der Wert 0, so wird eine neue Session gewünscht.
- **CipherSuite**: Eine Liste von vorhandenen kryptographischen Algorithmen, in absteigender favorisierter Ordnung. Jeder Eintrag definiert sowohl ein Chiffre wie auch einen Schlüsseltauschalgorithmus. Diese werden weiter unten genauer erläutert.
- **Compression Method**: Eine Liste von Kompressionsalgorithmen, die der Client unterstützt.

Nachdem der Client ein *client_hello* verschickt hat, wartet er auf die *server_hello* Nachricht. Diese vom Server verschickte Nachricht hat dieselben Felder wie die *client_hello* Nachricht. Das Version Feld enthält die höchste SSL Version, die der Server unterstützt und die auch vom Client verarbeitet werden kann. Der Inhalt des Random Feldes wird vom Server generiert und ist unabhängig vom Inhalt des Random Feldes aus der Client Nachricht. Wenn das Session ID Feld vom Client nicht 0 war, wird vom Server derselbe Wert benutzt. Ansonsten enthält dieses Feld den Wert der neuen Session. Das CipherSuite Feld enthält in der Serverantwort nur einen einzigen Algorithmus mit zugehörigem Schlüsselaustauschalgorithmus. Der Server hat diese aus den angebotenen Möglichkeiten vom Client ausgewählt. Das Compression Method Feld enthält wiederum nur einen Kompressionalgorithmus aus den Vorschlägen des Clients.
Das erste Element im CipherSuite Feld ist der Schlüsselaustauschalgorithmus. Folgende Schlüsselaustauschalgorithmen werden unterstützt:

- **RSA**: Der geheime symmetrische Schlüssel wird mit dem öffentlichen RSA Schlüssel des Empfängers verschlüsselt. Zu diesem Zweck muss ein Publickey Zertifikat verfügbar sein.

- **Fixed Diffie-Hellman**: Hier wird das Diffie-Hellman Schlüsseltauschprotokoll benutzt. Die öffentlichen Parameter des Diffie-Hellman Verfahrens müssen dazu im Server Zertifikat stehen und dieses von einer „certificate authority" (CA) signiert sein. Der Client stellt seine Diffie-Hellman öffentlichen Schlüssel Parameter entweder in einem Zertifikat zur Verfügung, falls Clientauthentisierung erforderlich ist, oder in einer Schlüsselaustauschnachricht.

- **Ephemeral Diffie-Hellman**: Hier werden temporäre Einmalschlüssel erzeugt. Die Diffie-Hellman öffentlichen Schlüssel werden mit dem privaten RSA- oder DSS-Schlüssel des Senders signiert. Zertifikate werden eingesetzt, um die öffentlichen Schlüssel zu verifizieren.

- **Anonymous Diffie-Hellman**: Der Diffie-Hellman Algorithmus wird ohne Authentikation[4] benutzt. Dazu schickt jede Seite ihre Diffie-Hellman Parameter an die andere Seite, ohne eine Authentikation des Kommunikationspartners durchzuführen. Hierbei besteht deswegen die Gefahr für eine Man in Middle Attacke.

- **Fortezza**: Dieses ist ein Verfahren der U.S. Regierung. Es ist dem Diffie-Hellman Algorithmus sehr ähnlich. Das Verfahren wird benutzt, wenn die Schlüsselberechnung auf einem Token oder einer Smartcard erfolgt. Das Ergebnis des Schlüsselaustauschprozesses ist der „token encryption key" (TEK).

Nach dem Schlüsselaustauschalgorithmus folgen die Chiffreparameter, die aus folgenden Feldern bestehen:

- **Cipher Algorithm**: Ein Algorithmus aus RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza.

[4] In dieser Arbeit werden die Begriffe Authentisierung und Authentikation synonym verwendet.

- **MAC-Algorithm**: MD5 oder SHA-1.
- **Chipher Type**: Stream oder Block Chiffre.
- **Is Exportable**: true oder false.
- **Hash Size**: 0, 16 (MD5) oder 20 (SHA-1).
- **Key Material**: Eine Sequenz von Bytes, die für die Generierung von Schreibkeys benutzt wurde.
- **IV Size**: Die Größe des Initialisierungswertes für die „Cipher Block Chaining“ Verschlüsselung.

Phase 2: Serverauthentisierung und Schlüsseltausch

Der Server beginnt die Phase mit dem Senden seines Zertifikates in einer *certificate* Nachricht, falls er authentisiert werden soll. Dies ist nicht notwendig, wenn `Anonymous Diffie-Hellman` benutzt wird. Diese Nachricht enthält ein X.509 Zertifikat oder eine Kette von X.509 Zertifikaten. Falls `Fixed Diffie-Hellman` verwendet wird, stehen in diesem Zertifikat die öffentlichen Diffie-Hellman Parameter. Anschließend kann, falls notwendig, eine *server_key_exchange* Nachricht vom Server gesendet werden. Dieses wird nicht verlangt, falls der Server `Fixed Diffie-Hellman` oder `RSA` zum Schlüsseltausch benutzt. Nun kann der Server ein Clientzertifikat anfordern, dazu schickt er eine *certificate_request* Nachricht. Diese enthält zwei Parameter: certificate_type und certificate_authorities. Der certificate_type bezeichnet den Publickey Algorithmus des Zertifikates und certificate_authorities enthält eine Liste von ausgezeichneten Namen und akzeptierten CAs. Abschließend für die 2. Phase wird die *server_done* Nachricht geschickt. Diese ist immer erforderlich. Nach dem Senden dieser Nachricht wartet der Server auf die Antwort vom Client. Diese Nachricht besitzt keine Parameter.

Phase 3: Clientauthentisierung und Schlüsseltausch

Nachdem der Client die *server_done* Nachricht empfangen hat, prüft dieser, falls notwendig, ob er ein validiertes Zertifikat erhalten hat und verifiziert ob die *server_hello* Parameter akzeptabel sind. Wenn alles zufriedenstellend ist, schickt

der Client folgende Nachrichten an den Server. Falls der Server ein Client Zertifikat fordert, so schickt der Client zunächst eine *certificate* Nachricht zum Server. Anschließend muss eine *client_key_exchange* Nachricht zum Server übertragen werden. Der Inhalt dieser Nachricht hängt von dem verwendeten Schlüsseltauschprotokoll ab:

- **RSA**: Der Client generiert ein 48 Byte „pre-master secret“ und verschlüsselt dieses mit dem öffentlichen Schlüssel des Servers. Entweder aus dem Zertifikat oder mit dem temporären RSA Schlüssel aus der *server_key* Nachricht. Dieses wird später benutzt, um einen „master secret“ zu berechnen.
- **Ephemeral oder Anonymous Diffie-Hellman**: Die öffentlichen Diffie-Hellman Parameter des Clients werden verschickt.
- **Fixed Diffie-Hellman**: Die öffentlichen Diffie-Hellman Parameter wurden schon in der *certificate* Nachricht verschickt, daher ist der Inhalt dieses Paketes null.
- **Fortezza**: Die Fortezza Parameter werden verschickt.

Anschließend kann der Client eine *certificate_verify* Nachricht verschicken. Diese dient zum expliziten Verifizieren des Clientszertifikats. Diese Nachricht wird nur gesendet, wenn das Clientzertifikat Signierfunktionalität besitzt (alle außer `Fixed Diffie-Hellman`). Diese Nachricht enthält den signierten Hashcode aller vorherigen Nachrichten. Wenn jemand das Client Zertifikat entwendet hat, aber nicht den privaten Schlüssel für dieses Zertifikat besitzt, kann er diese Nachricht nicht erstellen.

Phase 4: Finish

Diese Phase beendet den Aufbau einer gesicherten Verbindung. Der Client sendet eine *change_cipher_spec* Nachricht und kopiert die ausgehandelte ChipherSpec in die aktuelle CipherSpec. Diese Nachricht ist nicht mehr Teil des *SSL-Handshake Protocols*, sondern nutzt das *Change Cipher Spec Protocol*. Anschließend schickt der Client eine *finished* Nachricht. Diese Nachricht wird mit dem neuen Algorithmus, Schlüssel und Geheimnis erstellt. Diese Nachricht verifiziert,

dass der Schlüsseltausch und Authentisierungsprozess erfolgreich waren. Der Inhalt dieses Paketes besteht aus zwei Hashwerten:

$$MD5(master\_secret \| pad\_2 \| MD5(handshake\_messages \| Sender \| master\_secret \| pad\_1))$$

und

$$SHA(master\_secret \| pad\_2 \| SHA(handshake\_messages \| Sender \| master\_secret \| pad\_1))$$

Wobei $Sender$ ein Code ist, der den Client identifiziert und $handshake\_messages$ alle ausgetauschten Nachrichten bis zu dieser enthält.

Der Server antwortet auf diese empfangene Nachricht mit seiner *change_Cipher_Spec* Nachricht und transferiert die ausgehandelte CipherSpec in seinen aktuellen CipherSpec. Anschließend schickt auch der Server eine *finished* Nachricht.
Nun ist der Verbindungsaufbau vollständig und Client und Server können die Daten aus der Applikationsschicht austauschen. Auf der nächsten Seite findet sich eine Darstellung des zeitlichen Ablaufs der Protokollschritte.

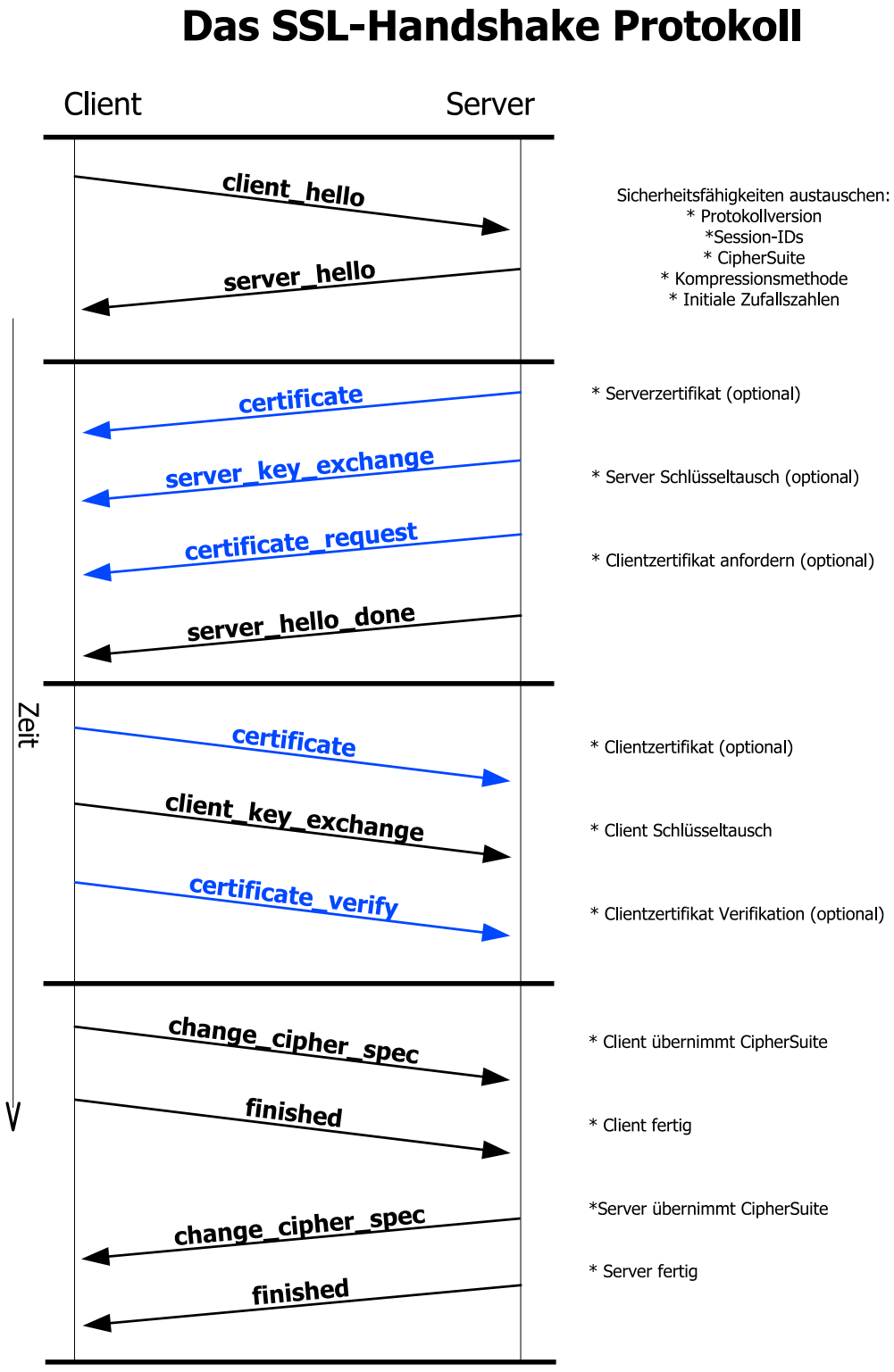

Abbildung 2.4: SSL-Handshake Protokoll

Generierung des Masterkey:

Der gemeinsam benutzte **Masterkey** ($master\_secret$) besteht aus 48 Byte und gilt nur für eine Session. Der Schlüssel wird in zwei Stufen generiert. Zuerst wird im Handshake Protokoll ein pre_master_secret ausgetauscht. Dabei existieren zwei Fälle für den Austausch des pre_master_secret:

- **RSA**: Das 48 Byte große pre_master_secret wird vom Client erzeugt, mit dem öffentlichen Schlüssel des Servers verschlüsselt und anschließend an diesen versandt. Der Server entschlüsselt die erhaltene Nachricht mit seinem privaten Schlüssel, um den pre_master_secret zu erhalten.
- **Diffie-Hellman**: Sowohl der Client als auch der Server generieren einen öffentlichen Diffie-Hellman Schlüssel. Nachdem diese Schlüssel ausgetauscht worden sind, führt jede Seite eine „Diffie-Hellman Calculation“ durch, um das gemeinsame pre_master_secret zu generieren.

In der zweiten Stufe berechnen dann beide Parteien aus diesem pre_master_secret den **Masterkey**:

master_secret := MD5(pre_master_secret || SHA('A' || pre_master_secret || ClientHello.random || ServerHello.random)) ||
MD5(pre_master_secret || SHA('BB' || pre_master_secret || ClientHello.random || ServerHello.random)) ||
MD5(pre_master_secret || SHA('CCC' || pre_master_secret || ClientHello.random || ServerHello.random)) ||

Dabei sind *ClientHello.random* und *ServerHello.random* die beiden Werte aus der ausgetauschten anfänglichen *hello_message*.

Erzeugung der CipherSpecs:

Die benutzten CipherSpecs benötigen ein Client-MAC-Geheimnis, ein Server-MAC-Geheimnis, einen Clientschlüssel, einen Serverschlüssel, einen Clientinitialisierungsvektor und einen Serverinitialisierungsvektor. Diese werden aus dem **Masterkey** erzeugt. Dazu werden mehrere Hashes des **Masterkey** für die notwendige Anzahl von Bytes für die einzelnen Parameter errechnet. Man kann somit

den **Masterkey** als Seed Wert für eine pseudozufällige Berechnung ansehen und die *ClientHello.random* und *ServerHello.random* als zusätzliche Verschleierung gegen eine Kryptoanalyse.

Unterschiede zwischen SSLv3 und TLS:
TLS, das aufbauend auf SSLv3 bei der IETF entstanden ist, ist unter [DA99] spezifiziert. Im folgenden werden kurz die Unterschiede zu SSLv3 vorgestellt:

- **Versionsnummer**: Die Versionsnummer im *Recordformat* unterscheidet sich zu SSLv3. Aktuell ist für TLS die Major Version 3 und die Minor Version 1.
- **Message Authentication Code**: Während beim MAC-Algorithmus in SSLv3 die Padding Daten mit dem geheimen Schlüssel verkettet werden, werden bei TLS die Padding Daten „XORed“[5]. Außerdem wird bei der MAC-Berechnung bei TLS noch ein zusätzlicher Parameter als bei SSLv3 benutzt, der Wert TLScompression.version.
- **Pseudozufallsfunktion**: TLS benutzt zusätzlich noch eine Pseudozufallsfunktion in der Generierung und Verifikation der Schlüssel. So können aus relativ kleinen Geheimnissen größere Byteblöcke generiert werden, welche resistenter sind bei Attacken gegen die Hashfunktionen.
- **Alert Codes**: TLS implementiert alle Alert Codes aus SSLv3 und fügt noch zusätzliche hinzu.
- **CipherSuite**: TLS unterstützt alle Schlüsselaustauschprotokolle und alle symmetrischen Verschlüsselungsalgorithmen außer Fortezza.
- **Client Zertifikat Type**: SSLv3 besitzt zusätzlich zu TLS `rsa_ephemeral_dh`, `dss_ephemeral_dh` und `fortezza_kea` als Zertifikatstypen. `Ephemeral Diffie-Hellman` benutzt entweder RSA oder DSS, um die Diffie-Hellman Parameter zu signieren. Bei TLS werden dafür die vorhandenen `rsa_sign` oder `dss_sign` benutzt. TLS beinhaltet nicht das Fortezza Schema.

[5]Die mathematische Verknüpfung XOR bezeichne ich auch als Veroderung innerhalb dieser Arbeit.

- **Certificate_Verify and Finished Messages**: Bei TLS wird in der *certificate_verify* Nachricht der Hash nur über die *handshake* Nachricht erzeugt. Die extra Parameter $master\_secret$, pad_1 und pad_2 aus SSLv3 würden die Sicherheit nicht erhöhen. Der Hash in der *finished* Nachricht wird bei TLS anders berechnet.

- **Master Secret**: Der pre_master_secret wird wie bei SSLv3 generiert. Der daraus berechnete $master\_secret$ wird aber anders berechnet. Auch die aus dem $master\_secret$ erzeugten MAC Geheimnisse, Session Schlüssel und Initialisierungs Vektoren werden anders berechnet.

- **Padding**: Bei SSLv3 werden die Applikationsdaten auf das Minimum der benötigten Chiffreblockgröße aufgefüllt. TLS erlaubt auch das Auffüllen in ein Vielfaches der benötigten Chiffreblockgröße. Dies soll Angriffe über die Länge der übermittelten Nachrichten erschweren.

2.2.4 Domain Name System

Zwar funktioniert das TCP/IP-Protokoll auch ohne das „Domain Name System“ (DNS), dennoch bildet dieses Protokoll eines der Grundbausteine des heutigen Internets. Das DNS bildet den Namensraum des Internets in einer hierarchisch verteilten Datenbank ab. Das DNS wird benutzt, um Domänennamen in IP-Adressen aufzulösen und umgekehrt. So muss sich ein Anwender nicht die IP-Adresse eines bestimmten Servers einprägen, sondern nur den leichter zu merkenden DNS-Namen. Die beiden Hauptspezifikationen zu DNS finden sich unter [Moc87b] und [Moc87a]. Mehrere darauf basierende Erweiterungen sind inzwischen erschienen, wie die „Domain Name System Security Extensions“ (DNSSec).

Das „Domain Name System“ kann in drei Bestandteile unterteilt werden: Der Domänennamensraum, der Nameserver und der Resolver.

Domänennamensraum:

DNS-Namen sind aus alphanumerischen Zeichen und den beiden Sonderzeichen „-“ und „.“ aufgebaut und dürfen nicht mehr als 255 Zeichen enthalten. Dabei kommt dem „.“ eine besondere Bedeutung zu. Als Beispiel verwenden wir

im Folgenden den DNS-Namen WWW.BEI.SPIEL.DE.. Die Punkte fungieren als Trenner innerhalb des Namens, um unterschiedliche Namensräume zu klassifizieren. Die Zeichenkette zwischen den Punkten wird als `Label` bezeichnet. Soll ein DNS-Name zu einer IP Adresse aufgelöst werden, so wird der DNS-Name von rechts nach links durchlaufen. Der oberste Namensraum wird durch den ganz rechten[6] „." bezeichnet. Dieser Namensraum wird von einigen wenigen Nameservern (Rootservern) verwaltet und bezeichnet die Wurzel des DNS. Diese Rootserver wissen, welche anderen Nameserver für die unterschiedlichen „Top Level Domains" (TLDs) zuständig sind. Die TLD ist das ganz rechts stehende Label. TLDs können unterschieden werden in „generic TLDs"[7] (gTLDs) und in „country-code TLDs"[8] (ccTLDs). Die Nameserver für den Namensraum .de. aus unserem Beispiel sind zuständig für die Verwaltung dieses Namensraums und führen eine Liste von weiteren Nameservern, die die nächsten Unternamensräume in .de. verwalten[9]. In unserem Beispiel wäre das der Namensraum SPIEL.DE. mit einem eigenen Nameserver, der die Domäne SPIEL.DE. verwaltet und einen Eintrag enthält, welcher Nameserver für die Domäne BEI.SPIEL.DE zuständig ist. Dieses Auflösungsschema wird rekursiv weiter durchlaufen bis man an das ganz linke Label gelangt. Dieses ganz linke Label bezeichnet meist einen konkreten Rechner innerhalb eines durch den Rest des DNS-Namens definierten Namensraumes. Somit spannt das DNS einen hierarchischen Baum mit verteilten dezentralen Datenbanken auf.

Nameserver:

Ein Nameserver ist der Dienst, den Clients anfragen können, um Informationen aus seiner DNS-Datenbank zu erhalten. Dabei wird unterschieden zwischen **autorisiertem** Nameserver und **nicht-autorisiertem** Nameserver.

- **Autorisierter Nameserver**: Dieser Nameserver ist zuständig für einen

[6] Häufig wird dieser Punkt nicht mit angegeben. Die Software fügt dann bei einer Anfrage automatisch diesen Punkt hinzu.

[7] Dies sind allgemeine TLDs, wie com., net., info., name.

[8] Dies sind Länderspezifische TLDs, wie de., us., fr., uk.

[9] Der de. Namensraum wird von der Registrierungstelle „DENIC Domain Verwaltungs- und Betriebsgesellschaft eG" verwaltet. Bei dieser Institution können neue Unternamensräume in de. registriert werden und die Nameserver der Denic führen dann diese Unterdomäne mit zugehörigen Nameserververweisen.

bestimmten Domänennamensraum. Er verwaltet eine Liste von Rechnernamen mit IP-Adressen und weitere Informationen zu seinem Domänennamensraum, darunter eventuell auch, welcher Nameserver zuständig ist für weitere Unternamensräume. Zu jedem Domänennamensraum existiert zumindet ein autorisierter Nameserver. Dieser kann aber aus Redundanz- und Lastverteilungsgründen geclustert vorkommen, wobei die Datensätze des primären Nameservers repliziert auf den secondary Nameservern vorliegen.

- **Nicht-autorisierter Nameserver**: Da Nameserver auch DNS-Informationen von anderen Nameservern ausgeben können, werden solche Informationen dann von einem nicht-autorisierten Nameserver erstellt. Dies bedeutet nicht, dass diese Informationen nicht verarbeitet werden sollten, sondern es kennzeichnet den Sachverhalt, dass der antwortende Nameserver nicht wirklich für diesen Namensraum zuständig ist und seine Informationen eigentlich von einem anderen Nameserver kommen. DNS-Antworten an einen Anfragenden kommen häufig von nicht-autorisierten Nameservern, da Nameserver in der Regel Antworten zwischenspeichern, um nicht bei jeder neuen Anfrage den Abfragebaum erneut durchlaufen zu müssen.

Resolver:

Als Resolver bezeichnet man die Software, die beim Client eine Auflösung des DNS-Namen in die IP-Adresse bewerkstelligen soll. Dazu kontaktiert der Resolver einen voreingestellten Nameserver im Internet, um die Auflösung zu beginnen. Dies geschieht in der Regel mittels UDP über Port 53, kann aber auch mit TCP über Port 53 erfolgen. Bei Resolvern kann zwischen iterativen und rekursiven Verfahren unterschieden werden. Beim iteraktiven Verfahren durchläuft der Resolver selbst den Domänenbaum und führt die Anfragen jeweils selbst durch. Bei rekursiven Anfragen fragt der Resolver nur einen Nameserver an und überlässt dem Nameserver anschließend das Durchwandern des Domänenbaumes. Dieser primär angefragte Nameserver liefert anschließend dem Resolver die gewünschte IP und speichert die Informationen selbst zwischen oder meldet, dass dieser DNS-Name nicht aufgelöst werden konnte. Die meisten Clients nutzen das rekursive

Verfahren für DNS-Anfragen.

2.2.5 HTTP

Das „Hypertext Transfer Protocol“ (HTTP) dient zum Abfragen von Daten im Internet. 1989 wurde dieses Protokoll von Tim Berners-Lee am CERN[10] zusammen mit der URL[11] und HTML[12] entwickelt. Diese drei Elemente bilden den Kern des „World Wide Web“ (WWW). HTTP ist dabei ein zustandloses Protokoll, d.h. es bietet keine Korrelation bei Mehrfachanfragen. Wurde es zunächst rein für die Abfrage von Webseiten im WWW benutzt, ist es durch Erweiterungen inzwischen nicht mehr nur auf die einseitige Anforderung von Hypertextdokumenten beschränkt.

HTTP teilt sich in Header- und Bodydaten auf. Im Header können zusätzliche Informationen übertragen werden, wie *User-Agent, gewünschte Sprache, Statuscodes, usw.*. Im Body werden die tatsächlich angeforderten Informationen übermittelt. Zusätzlich können an eine URL noch *Variablen=Werte* Argumente übergeben werden. Dies geschieht durch das Anhängen eines *?* und anschließender Aufzählung der *Variablen=Werte*, getrennt durch ein *&*. Eine andere Möglichkeit, Daten an einen Server mittels HTTP zu übermitteln, bietet die POST-Methode. Dabei werden die zu übermittelnden Daten nicht in der URL kodiert, sondern nach dem Anfrageheader im *Entity Body* übertragen. Die POST-Methode hebt bestimmte Längenbeschränkungen wie bei der GET-Methode (URL-Übertragung) auf und wird üblicherweise über ein eingebettetes Formularfeld angestossen.

Wurde bei HTTP 1.0 (spezifiziert in [BLFF96]) noch für jede Anfrage ein eigener Kommunikationskanal aufgebaut, so erlaubt HTTP 1.1 (spezifiziert in [FGM+99]) auch das Anfordern von mehreren Elementen über einen Kommunikationskanal.

[10] Die „European Organization for Nuclear Research“ ist eine europäische Kernforschungseinrichtung.

[11] „Uniform Resource Locator“ (URL) ist eine Kennzeichnung, die eindeutig eine Ressource im Internet identifiziert. Eine URL baut sich aus den Komponenten *Protokoll://Rechner/Ressource* auf. Eine URL ist eine Ausprägung der „Uniform Resource Identifier“ (URI). URI ist spezifiziert in [BLuLM98]

[12] „Hypertext Markup Language“ (HTML) ist ein Dokumentenformat. HTML dient als Auszeichnungssprache zur Beschreibung von Hypertextdokumenten.

Ein besonderer Statuscode von HTTP findet in den später vorgestellten Systemen besonders häufig eine Verwendung. Es handelt sich dabei um den Statuscode 302. Dieser Statuscode in der Antwort eines Servers kann den Client veranlassen, die angeforderte Ressource unter einer anderen URL zu erfragen. In der Regel findet ein solches Umlenken ohne Benutzerinteraktion statt.

Da HTTP nur sehr rudimentäre Sicherheitseigenschaften aufweist, besteht die Möglichkeit, HTTP über einen SSL/TLS Tunnel zu betreiben. Dieses Verfahren ist in [Res00] spezifiziert.

2.2.6 Cookies

Da HTTP ein zustandloses Protokoll ist, wurde von Netscape das Cookie Verfahren[13] eingeführt. Inzwischen ist es als RFC spezifiziert unter [uLM00]. Cookies sind kleine Informationspakete aus ASCII-Zeichen, die in den Header Teilen von HTTP-Anfragen und HTTP-Antworten übertragen werden. Der Webbrowser speichert diese Daten für eine spätere Verwendung zwischen. Bei der Speicherung von Cookies wird unterschieden zwischen *Persistenten Cookies* und *Session Cookies*.

- *Session Cookies*: Die erhaltenen Cookiedaten werden beim Webbrowser nur für eine Benutzersitzung zwischengespeichert. Beendet der Benutzer den Webbrowser und startet diesen anschließend neu, so sind die Daten der *Session Cookies* gelöscht.

- *Persistente Cookies*: Die Cookiedaten werden vom Webbrowser an einem Ort auf dem Rechner abgespeichert. Somit bleiben die Daten auch nach einer Beendigung und Neustart des Webbrowsers erhalten.

Ein Cookie besitzt einen wahlfreien Inhalt und darf nicht größer als 4 KB sein. Ein Server darf mehrere Cookies je Antwort beim Benutzer setzen und ein Client sollte für mindestens zwanzig Cookies pro Server (Domänennamen) Platz vorhalten. Der Benutzer kann auch mehrere Cookies bei einer Anfrage zum Server

[13] Die Spezifikation von Netscape kann unter `http://wp.netscape.com/newsref/std/cookie_spec.html` eingesehen werden.

übertragen. Cookies sollten mit einer Lebensdauer versehen werden, die angibt, wie lange ein solcher Cookie gültig ist.

Durch diese Möglichkeit der Datenspeicherung beim Benutzer werden Zustandsinformationen realisiert, da bei jeder Anfrage des Clients diese Cookiedaten wieder zum Server übermittelt werden. Damit nicht jeder Server alle gespeicherten Cookiedaten beim Benutzer lesen kann, werden Cookies nur an den Server übertragen, der diese Daten zuvor gesetzt hat. Dabei wird aber nicht zwingend die IP-Adresse oder der ganze DNS-Name des Servers benutzt, sondern es wird nur ein Teil des DNS-Namens verwendet. Es wird nur der Domänenteil abgespeichert und nicht der zugehörige Rechnername. Somit hat jeder Rechner aus der Domäne Zugriff auf diese Cookiedaten. Bei manchen Webbrowsern wird auch nur ein kleinerer Teil des Domänennamens für eine Legitimierung der Übertragung benutzt.

2.2.7 SOAP

Das „Simple Object Access Protocol“ (SOAP) wurde 1999 von Dave Winer und Don Box in der Version 0.9 veröffentlicht. Kurz darauf erschien die Version 1.0. Im Jahre 2000 wurde dann die Version 1.1 als Note beim „World Wide Web Consortium“ (W3C) eingereicht, mit dem Ziel, dass sich dort eine Arbeitsgruppe um die Weiterentwicklung kümmern sollte. Im Juni 2003 verabschiedete diese Arbeitsgruppe dann die Version 1.2 mit den beiden Spezifikationen [GHM+03a] und [GHM+03b].

Mit Hilfe von SOAP lassen sich Dateien austauschen und „Remote Procedure Calls“ (RPC) realisieren. Ziel von SOAP ist die Erstellung eines leichtgewichtigen und gut erweiterbaren Protokolls für diese Zwecke. Dabei nutzt SOAP schon vorhandene und etablierte Standards. Das Austauschformat basiert auf der „Extensible Markup Language“ (XML) und es können unterschiedliche Protokolle für den Austausch der Nachrichten verwendet werden. Am häufigsten wird SOAP-over-HTTP verwendet, das sich für den Austausch der Informationen des HTTP-Protokolls bedient.

Eine SOAP-Nachricht besteht aus einem Header und einem Body. Im Header stehen Metainformationen, im Body die Nutzdaten. So braucht nur der Body

vom Empfänger ausgewertet zu werden, alle anderen Zwischenstationen müssen nur den Header betrachten.

SOAP bildet einen der Grundbausteine für den Aufbau von Webservices und wird immer häufiger für diesen Zweck eingesetzt.

2.2.8 SAML

Die „Security Assertion Markup Language“ (SAML) ist eine Auszeichnungssprache für Sicherheitsbestätigungen. Anhand von SAML können sicherheitsbezogene Informationen beschrieben und ausgetauscht werden. SAML nutzt für den Dokumentenaufbau das XML-Format. Die Spezifikation von SAML erfolgt durch die „Organization for the Advancement of Structured Information Standards“ (OASIS). Die Version 1.1 von SAML ist in [OAS03a] und [OAS03b] spezifiziert.

Die SAML-Spezifikation hat drei Hauptbestandteile:

- **Assertions**: Hier werden Informationen zu Authentisierung, Autorisierung sowie weitere Sessionattribute hinterlegt.
- **Protokoll**: Hier wird definiert, wie SAML-Assertions angefordert und übermittelt werden.
- **Bindings und Profile**: Hier wird festgelegt, wie SAML-Dokumente (Assertions) in Standard,Transport und Messaging Frameworks eingebunden werden.

SAML Assertions sollen vor allem drei Schwerpunkte abdecken:

- Authentikation: SAML-Dokumente können Aussagen über den Status einer Authentikation machen.
- Attribute: SAML-Dokumente können Anwender- und Sitzungsattribute enthalten.
- Autorisierungen: SAML-Dokumente können Autorisierungsinformationen beinhalten.

SAML-Assertions können digital signiert werden. Ihre Ausgabe erfolgt durch vertrauenswürdige Instanzen innerhalb eines Systems, den sogenannten „Assertion Issuing Autorities“.

Somit können diese Sicherheitsinformationen mittels SAML-Dokumenten zwischen Systemen ausgetauscht werden. Dabei legt SAML die Struktur des Austauschs von Sicherheitsinformationen fest, jedoch nicht die konkreten Mechanismen.

2.2.9 XML-Encryption und XML-Signature

Die Standards XML-Encryption (xmlenc) und XML-Signature (xmlsig) erläutern, wie man Daten in XML-Dokumenten verschlüsselt oder signiert.

XML-Encryption

XML-Encryption ist eine Spezifikation des „World Wide Web Consortium“ (W3C). Die Spezifikation [IDS02] beschreibt, die Verschlüsselung von Daten und die Darstellung des Ergebnisses in XML. Dabei können die Daten aus einem XML-Dokument, einem XML-Element, einem XML-Element Inhalt bestehen oder sonstiger Art sein. Das Ergebnis ist ein „XML Encryption Element“ (`EncryptedData`), das die verschlüsselten Daten oder eine Referenz darauf enthält. Das Element `EncryptedData` kann das Rootelement des XML-Dokumentes sein oder nur eines von vielen Elementen innerhalb eines XML-Dokuments.

Zusätzlich kann noch angegeben werden, welches Verschlüsselungsverfahren benutzt wurde und erweiterte Schlüsselinformationen. Bei diesen Schlüsselinformationen kann es sich um Referenzen auf den Schlüssel handeln oder bei asymmetrischen Verfahren kann gleich der öffentliche Schlüssel mit in das XML-Dokument eingebunden werden. Außerdem werden „Key-Wrap“ Verfahren angeboten, um auch symmetrische Schlüssel zu übertragen. Für die Übertragung wird der Schlüssel selbst verschlüsselt.

XML-Encryption verlangt, dass bestimmte Verschlüsselungsverfahren mit bestimmten Schlüssellängen implementiert sein müssen. Andere Verfahren sind dabei optional anzubieten. Welche Verfahren angeboten werden müssen und welche optional sind, kann der Spezifikation entnommen werden.

XML-Signature

XML-Signature ist eine Spezifikation des „World Wide Web Consortium“ (W3C). Die Spezifikation [BBF+02] beschreibt, wie man Daten signiert und die Signatur in XML darstellt. XML-Signature stellt Verfahren für Integritätschutz, Nachrichtenauthentikation und Signiererauthentikation bereit. Dabei können die zu verarbeitenden Daten schon als XML vorliegen oder in einer anderen Art. Zudem kann die Signatur in dem XML-Dokument eingebettet sein oder extern vom Dokument verwaltet werden.

Der berechnete Hashwert der Daten sowie weitere Zusatzinformationen werden in ein eigenes XML-Element verpackt. Anschließend wird über dieses XML-Element erneut ein Hashwert berechnet und dann kryptografisch anhand einer digitalen Signatur abgesichert. XML-Signaturen werden durch das `Signature` Element innerhalb eines XML-Dokuments gekennzeichnet. Ein `Signature` Element enthält seinerseits wieder verschachtelte weitere signaturbezogene XML-Elemente. Der Verweis der Signatur auf die signierten Ausgangsdaten findet über *URI*s statt.

Wie bei XML-Encryption können in den Zusatzinformationen der verwendete Algorithmus und Schlüsselinformationen enthalten sein. Auch Verweise auf Zertifikate sind bei beiden Verfahren möglich. XML-Signature verlangt, dass bestimmte Signierverfahren implementiert sein müssen, während andere optional sind. Um welche Verfahren es sich handelt, kann der Spezifikation entnommen werden.

2.2.10 Publickey Infrastruktur und x.509 Zertifikate

„Publickey Infrastruktur“ (PKI) sind Systeme, die es ermöglichen, digitale Zertifikate auszustellen, zu verteilen und zu überprüfen.

Es entsteht eine hierarchisch gegliederte Baumstruktur mit Instanzen an den Knoten, die Zertifikate ausgeben. Mit Hilfe dieser Zertifikate können anschliessend Signaturen überprüft werden. Denn die Gewährleistung der Echtheit von Schlüsseln für Signaturen oder Verschlüsselungen in asymmetrischen Kryptoverfahren ist problematisch. Um eine Signatur zu verifizieren, benötigt man den öffentlichen Schlüssel der Person, die die Signatur erstellt hat. Wird dieser Schlüssel

über unsichere Kanäle übertragen, so könnte der Schlüssel (und natürlich zuvor auch die Signatur) von einer betrügenden Person stammen.

Deswegen wird der öffentliche Schlüssel selbst von einer Instanz signiert und somit gewährleistet, dass dieser Schlüssel einem Objekt (Person oder Rechner) zugeordnet werden kann. Der Verifizierungsschlüssel dieser Instanz kann erneut von einer höheren Instanz signiert sein. Somit erhält man eine Baumstruktur von Vertrauensbeziehungen. Natürlich muss man der Echtheit des obersten Zertifikates , also der „Root CA“, inklusive des Verifikationsschlüssels, vertrauen.

Eine minimale PKI sollte folgende Bestandteile vorhalten:

- Digitale Zertifikate: Dies sind Objekte, die wiederum selbst digital signiert sind und benutzt werden, um die Echtheit von Schlüsseln zu gewährleisten.
- Certification Authority (CA): Diese Organisationen stellen Zertifikate bereit.
- Registration Authority (RA): Bei diesen Organisationen können Zertifikate für Personen oder Rechner beantragt werden.
- Certificate Revocation Lists: Dies sind Sperrlisten von zurückgezogenen, abgelaufenen oder für ungültig erklärten Zertifikaten.
- Verzeichnisdienst: Um an Zertifikate für eine Verifizierung zu gelangen, sollte ein durchsuchbares Verzeichnis existieren, bei dem der Anwender für das gewünschte Zertifikat anfragen kann.
- Validierungsdienst: Dies ist der Dienst, zumeist auf dem Anwenderrechner, der eine Signatur überprüft und dabei auch die Vertrauenskette von Zertifikaten einer PKI beachten sollte.

Zusätzlich ist es natürlich erforderlich, Regeln und Abläufe für das Ausstellen von Zertifikaten sowie für deren gesicherte Speicherung zu definieren. Damit man einer Certification Authority vertrauen kann, muss ein Anwender deren Hauptzertifikat vertrauen sowie ihrer Verantwortlichkeit bezüglich der Bereitstellung, der Zuweisung und der Integritätssicherung der von ihr herausgegebenen Zertifikate.

Dies ist zumindest schon bei Webbrowsern ein Problem. Aktuelle Webbrowser werden mit einer Vielzahl von Zertifikaten von ganz unterschiedlichen Certification Authorities ausgeliefert. Jedes Zertifikat, welches von einer dieser mitgelieferten Certification Authority ausgestellt wurde, wird somit vom Webbrowser ohne Nachfrage beim Benutzer akzeptiert.

Zertifikate werden im X.509 Schema verwaltet und ausgetauscht. x.509 entstand ursprünglich im Zusammenhang mit dem X.500 Standard der ITU-T[14]. Inzwischen bezieht man sich aber meist bei X.509 auf Version 3 (X.509v3) der „Internet Engineering Task Force“ (IETF), das unter [HPFS02] spezifiziert ist.

Ein X.509v3 Zertifikat besteht aus unterschiedlichen Feldern:

- Version: Enhält die Versionsnummer des benutzten X.509 Standards, also entweder den Wert 1, 2 oder 3.
- Seriennummer: Hier steht eine Seriennummer, welche die Zertifizierungstelle vergibt. Diese Seriennummer dient u.a. zum Widerruf von Zertifikaten.
- Algorithmen ID: Hier wird der Algorithmus bezeichnet, der verwendet wurde, um die Signatur zu erstellen.
- Aussteller: Dieses Feld enthält den eindeutigen Bezeichner des Ausstellers im X.500 Format.
- Gültigkeit: Hier wird die Gültigkeitsdauer des Zertifikates notiert.
- Subject: Hier steht der Name des Zertifikatsinhabers, also der Person, für die das Zertifikat ausgestellt wurde, im X.500 Format. Ein Webbrowser wertet innerhalb dieses Feldes nur das *CN*-Feld aus, welches meist den Servernamen im DNS-Format enthält.
- Subject Publickey Info: Dieses Feld enthält den öffentlichen Schlüssel des Zertifikatsinhabers, sowie einen Identifikator für den Algorithmus, mit dem der Schlüssel verwendbar ist.

[14]Das „Telecommunication Standardization Bureau“ (ITU-T) ist ein Teil der „International Telecommunication Union“ (ITU) und beschäftigt sich hauptsächlich mit Standardisierungen rund um die Telekommunikation.

- Eindeutige ID des Ausstellers: Dieses Feld ist optional und kann einen eindeutigen Bezeichner der austellenden Zertifizierungstelle enthalten. Dies kann nötig sein, wenn der X.500 Name mehrmals verwendet wurde.
- Eindeutige ID des Inhabers: Dieses Feld ist optional und kann einen eindeutigen Bezeichner des Inhabers des Zertifikates enthalten. Sowohl dieses Feld, wie auch das vorherige, werden heute kaum noch verwendet.
- Erweiterungen: In diesem Feld können zusätzliche Informationen untergebracht werden.
- Zertifikat Signaturalgorithmus: Hier wird der verwendete Algorithmus zur Erstellung der Signatur bezeichnet. Es enthält den gleichen Wert wie das „Algorithmen ID“ Feld.
- Zertifikat Signatur: Hier steht der Hashwert aller vorherigen Felder verschlüsselt mit dem privaten Schlüssel der Zertifizierungsstelle.

2.3 Grundlagen der Kryptographie

Die Kryptographie bietet mächtige und vielseitige Hilfsmittel, die es ermöglichen, die Sicherheit in vernetzten Systemen zu erhöhen. Im Folgenden werden die wichtigsten kryptographischen Prinzipien und Algorithmen vorgestellt. Dabei liegt der Schwerpunkt auf den Chiffren (sowohl den symmetrischen wie auch den asymmetrischen), aber auch andere notwendige Aspekte (wie Einweg-Hashfunktionen oder digitale Signaturen) werden erläutert. Die genauen Algorithmen werden nicht erklärt. Für eine tiefere Betrachtung der Algorithmen verweise ich auf die Bücher „Handbook of Applied Cryptography“ von A. Menezes, P. van Oorschot und S. Vanstone [MVO96], „Angewandte Kryptographie“ von Bruce Schneier [Sch96] oder „Cryptography and Network Security“ von William Stallings [Sta99a].

2.3.1 Grundlegende Begriffe

Zum besseren Verständnis hier zuerst einige Definitionen und Abkürzungen von Begriffen, die im Folgenden häufiger Verwendung finden werden:

- *Kryptologie*: Wissenschaft der Kryptographie und Kryptoanalyse.
- *Kryptographie*: Lehre vom geheimen Schreiben.
- *Kryptoanalyse*: Lehre vom "Brechen" von Chiffren.
- *Chiffre (cipher)* : Methode des Verschlüsselns.
- *Klartext (plaintext)*: unverschlüsselter Text (*M* für "Message" oder *P* für "Plaintext").
- *Chiffretext (ciphertext)*: verschlüsselter Text (*C* für "Chiffretext").
- *Chiffrieren (encrypt)*: verschlüsseln ($E_K(M) = C$).
- *Dechiffrieren (decrypt)*: entschlüsseln ($D_K(C) = M$).

Weiterhin wird später von sogenannten unsicheren Kanälen zu lesen sein. Darunter sind Datenwege zu verstehen, bei denen nicht ausgeschlossen werden kann, dass unberechtigte Personen Zugriff auf die über diese Datenwege übermittelten Daten erhalten. Zur Erläuterung von einigen Vorgehensweisen wird in diesem Text außerdem mit den Namen „Alice“ und „Bob“ gearbeitet. Dabei handelt es sich natürlich nicht um real existierende Personen, sondern vielmehr um Statthalter für jede Art von handelnden Akteuren der digitalen Welt (also sowohl Menschen als auch Maschinen).

2.3.2 Ziele

Der Einsatz von kryptographischen Methoden in der Praxis verfolgt i.A. das Ziel der Durchsetzung einer oder mehrerer Anforderungen:

- *Vertraulichkeit, Geheimhaltung*: Darunter ist zu verstehen, dass die betreffenden Dokumente bzw. die betreffende Kommunikation vor den neugierigen Blicken nicht autorisierter Personen zu schützen sind. Dieses wird normalerweise mittels Verschlüsselung der Dokumente erreicht.
- *Authentisierung*: Besonders auf elektronischen Kommunikationswegen ist es sehr einfach, sich falsche Identitäten zuzulegen. Durch das Fehlen von

persönlichem Kontakt ist man darauf angewiesen, den digitalen Informationen zu vertrauen. Ein vergleichbares Problem ist auch das Feststellen der Identitäten von Kommunikationspartnern. Was benötigt wird, sind Methoden zur zweifelsfreien Identifikation von Akteuren in der digitalen Welt (wobei hier Akteure selbstverständlich nicht nur Menschen, sondern z.B. auch Rechner oder Prozesse sein können).

- *Integrität*: Es muss möglich sein festzustellen, ob Daten durch Dritte manipuliert wurden. Dieses bezieht sich sowohl auf Manipulation, die auf unsicheren Kommunkationswegen erfolgt sein mag, wie auch auf Datenmanipulation, die ein Eindringling eventuell im System durchgeführt hat (um z.B. Systemsoftware durch eigene Programme zu ersetzen, die Hintertüren enthalten). In diesem Kontext ist unter Datenmanipulation Modifikation, Vernichtung und Ersetzung von Daten zu verstehen.

- *Unwiderrufbarkeit*: Es sollte Methoden geben, die verhindern, dass ein Teilnehmer an einem Kommunikationsaustausch dessen Stattfinden im Nachhinein leugnen kann.

2.3.3 Der Begriff der Chiffre

Verschlüsselung von Daten erfolgt mittels eines Algorithmus, der Chiffre genannt wird. Dieser Algorithmus teilt sich auf in zwei Transformationen, eine zum Verschlüsseln der Daten (to encrypt, in Zeichen E) und einer zum Entschlüsseln (to decrypt, in Zeichen D). Sowohl die Ver- wie auch die Entschlüsselung wird unter Verwendung von Schlüsseln (in Zeichen K) durchgeführt. Dabei können die Schlüssel zum Ver- und Entschlüsseln gleich (symmetrische Chiffren) oder unterschiedlich (asymmetrische Chiffren) sein. Um praktikabel zu sein, gibt es weitere Anforderungen an Chiffrieralgorithmen. Sie sollten einfach zu benutzen sein, d.h. die Generierung von Schlüsseln sollte unproblematisch sein, und anhand der Kenntnis des Schlüssels sollte es einfach sein, aus der Chiffriertransformation die Dechiffriertransformation zu berechnen.

Weiterhin sollte die Sicherheit des Algorithmus ausschließlich von der Geheimhaltung des Schlüssels abhängen und nicht auf der Geheimhaltung des Al-

gorithmus beruhen (keine „security through obscurity“). Denn man kann davon ausgehen, dass jeder noch so gut geschützte Algorithmus eines Tages aufgedeckt wird. Und schließlich sollten die Chiffrier- und Dechiffriertransformationen für alle Schlüssel effizient berechnet werden können, denn niemandem ist mit einem Algorithmus geholfen, der zwar nahezu unbrechbar, aber unverhältnismäßig aufwendig in Zeit und Systemressourcen ist (man kann sich Szenarien vorstellen, wo solch ein Algorithmus akzeptabel wäre, aber im Mittelpunkt dieser Arbeit stehen kryptographische Methoden, die in erster Linie Anwendung in der Kommunikation in Datennetzen finden).

2.3.4 Symmetrische Chiffren

Definition: Bei Symmetrischen Chiffren trifft die Analogie des Schlüssels zu. Wie bei der Tür, die vom gleichen Schlüssel zu- und auch wieder aufgeschlossen werden kann, wird auch der Text bei symmetrischen Chiffren mit dem gleichen Schlüssel ver- und wieder entschlüsselt.

Bei einem sicheren Nachrichtenaustausch zwischen verschiedenen Parteien muss dieser Schlüssel allen Beteiligten bekannt sein. Und hier liegt auch das Hauptproblem, auf das man bei der Benutzung von symmetrischen Chiffren stößt: Der sichere Schlüsselaustausch. Bevor man sicher vor unerwünschten Lauschern mittels verschlüsselter Botschaften kommunizieren kann, muss man den Schlüssel an alle Gesprächspartner verteilen, falls er ihnen noch nicht bekannt ist. Doch dieses ist kein triviales Problem, da ja offensichtlich kein sicherer Kanal zu den Gesprächspartnern vorhanden ist (sonst müsste man den Nachrichtenaustausch schließlich nicht verschlüsseln).

In der Familie der symmetrischen Chiffren gibt es zwei Kategorien: Die *Stromchiffren*, welche die Daten bitweise verschlüsseln, und die *Blockchiffren*, die den Text in Blöcken fester Größe verschlüsseln.

Stromchiffren

Prinzip der Stromchiffren: Eine Stromchiffre verschlüsselt den Klartext i.A. bitweise (manchmal aber auch in größeren Einheiten). Dieses geschieht in den meisten Fällen durch eine bitweise exklusive Veroderung (XOR) mit einem

Schlüsselstrom ($C_i = K_i \oplus P_i$, wobei C_i, K_i, P_i jeweils das *i*-te Bit des Chiffretextes, Schlüsselstroms und Klartextes sind). Die Entschlüsselung findet ebenfalls durch bitweises XOR mit dem gleichen Schlüsselstrom statt. ($C_i \oplus K_i = K_i \oplus P_i \oplus K_i = P_i \oplus 0 = P_i$). Der Schlüsselstrom wird durch einen Algorithmus (dem sogenannten Schlüsselstromgenerator) erzeugt, der als Eingabe zur Initialisierung einen Wert (den eigentlichen Schlüssel) bekommt. Der gleiche Schlüssel erzeugt immer den gleichen Schlüsselstrom, so dass den einzelnen Kommunikationspartnern lediglich der Schlüssel bekannt sein muss, um den Strom an Chiffretextbits zu entschlüsseln. Die Sicherheit einer Stromchiffre beruht auf der Güte des Schlüsselstromgenerators. Der Schlüsselstromgenerator produziert eine Folge von Bits, deren Auftreten möglichst zufällig ist. Je besser der Generator den Zufall approximiert, desto sicherer wird die Chiffrierung. Bei der Verwendung von Stromchiffren sollte man darauf achten, niemals den gleichen Schlüssel (und somit den gleichen Chiffrestrom) zweimal zu verwenden. Falls man dieses tut, könnte eine dritte Person, die die beiden Chiffretexte abgefangen hat, diese beiden miteinander exklusiv verodern. Als Ergebnis erhält sie eine exklusive Veroderung der beiden Klartexte (die Schlüsselstrombits fallen bei diesem Vorgehen weg). Dies ist leicht zu entschlüsseln, da das Vorkommen der einzelnen Zeichen in einer Sprache nicht zufällig ist. Durch die Kenntnis eines der Klartexte kann man nun den Schlüsselstrom berechnen (durch XOR des Klar- und des Chiffretextes) und hat dadurch die Möglichkeit, alle weiteren Nachrichten, die mit diesem Schlüssel chiffriert wurden, problemlos zu lesen.

Wenn man einen Schlüsselstromgenerator mit einem periodischen Schlüsselstrom verwendet, sollte man ebenfalls darauf achten, dass die Größe der zu verschlüsselnden Daten kleiner als die Länge der Schlüsselperiode ist. Ist dies nicht der Fall, gibt man eventuellen Lauschern einen Ansatzpunkt zum erfolgreichen Entschlüsseln des Textes: Es besteht so die Möglichkeit, den Chiffretext in Blöcke der Größe der Periode aufzuteilen und diese, ähnlich zum oben erwähnten Angriff, miteinander exklusiv zu verodern. Da sich der Schlüssel nach dem Durchlaufen der Periode wiederholt, fallen auch bei diesem Vorgehen die Schlüsselbits weg (s.o.).

Durch die oben beschriebenen Eigenschaften (Strom von Chiffretextbits durch bitweise Verschlüsselung; keine Notwendigkeit, spezielle Blockgrößen der zu ver-

schlüsselnden Daten einzuhalten) eignen sich Stromchiffren besonders für den Einsatz an Orten, wo es auf die Verschlüsselung kontinuierlicher Datenströme ankommt (z.B. Video/Audio).

Arten von Stromchiffren: Man kann zwischen zwei unterschiedlichen Arten von Stromchiffren unterscheiden: den *synchronen* und den *selbstsynchronisierenden* Stromchiffren.

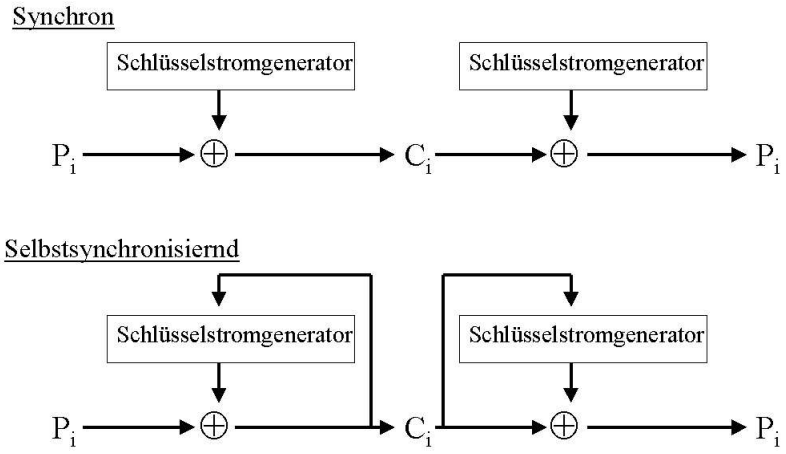

Abbildung 2.5: Synchrone und selbstsynchronisierende Stromchiffren

Bei den **synchronen Stromchiffren** ist der Schlüsselstrom unabhängig vom Datenstrom, d.h. der generierte Strom von Schlüsselbits wird allein bestimmt durch den verwendeten Algorithmus und den Schlüssel. Der Vorteil einer solchen Verfahrensweise ist, dass die Berechnung des Schlüsselstroms im Voraus erledigt werden kann, und beim eigentlichen Nachrichtenaustausch nur noch die XOR-Verknüpfung mit den Daten stattfinden muss. Besonders bei rechenintensiven Algorithmen kann man so den Datendurchsatz der Kommunikation erheblich erhöhen. Der Nachteil von synchronen Stromchiffren ist, dass bei nur einem verlorengegangenen Bit der gesamte folgende Chiffretext unbrauchbar ist. Dies ist der Fall, weil sowohl auf Sender- wie auch auf Empfängerseite die Schlüsselstromgeneratoren die jeweiligen Datenbits mit demselben Schlüsselstrombit (de)kodieren müssen, sie müssen also synchron vorgehen (daher auch

der Name). Falls ein Bit verloren geht, geraten die beiden in Asynchronismus. Während der Sender schon bei Bit Nummer *i+1* ist, ist der Empfänger noch bei Bit Nummer *i* und versucht somit eine Dechiffrierung anhand eines falschen Schlüsselstrombits. Dieser Fehler setzt sich im folgenden Chiffretext fort. Fehlerhaft übertragene Bits wiederum machen keine größeren Probleme. Die fehlerhaften Bits werden zwar falsch dechiffriert, aber der Rest der Nachricht bleibt entschlüsselbar.

Bei **selbstsynchronisierenden Stromchiffren** ist jedes Schlüsselstrombit eine Funktion einer festen Anzahl (n) vorhergehender Chiffretextbits. Dadurch bekommt die Chiffre die Eigenschaft, dass auch unter Verwendung des gleichen Schlüssels unterschiedliche Daten mit unterschiedlichen Schlüsselströmen kodiert werden. Somit fällt der Angriffsansatz aus 2.3.4 weg. Fehlerhaft übertragene oder fehlende Bits führen dazu, dass die nächsten n Bits falsch dechiffriert werden, da die Funktion zur Berechnung des Schlüsselstroms auf Sender- und Empfängerseite mit unterschiedlichen Chiffretextbits arbeitet. Nach n Bits haben sich die beiden aber wieder synchronisiert (deswegen selbstsynchronisierend) und arbeiten wieder korrekt (die Kommunikationspartner haben allerdings keine Möglichkeit zu entscheiden, ab wann wieder synchron gearbeitet wird, dieses kann bestenfalls aus den empfangenen Daten geschlossen werden). Dieses eigentlich sehr wünschenswerte Verhalten ermöglicht aber einen Angriff auf die Kommunikation durch Wiedereinspielung. Eine dritte Partei, die aus einem früher stattgefundenen Nachrichtenaustausch Chiffretextbits gespeichert hat, kann diese später in einen Nachrichtenaustausch wieder einschleusen. Auf Empfängerseite wird dann für n Bits unbrauchbarer Datenmüll entschlüsselt, danach hat sich der Schlüsselstromgenerator aber mit den alten Daten synchronisiert und entschlüsselt nun einwandfrei die eingeschleusten Daten. Dies funktioniert natürlich nur, wenn der Schlüssel in der Zwischenzeit nicht geändert wurde.

Ein Beispiel für eine häufig verwendete Stromchiffre ist RC4 (von Ron Rivest, 1987). RC4 ist eine synchrone Stromchiffre mit einer variablen Schlüssellänge und findet unter anderem Einsatz in Lotus Notes und Oracle Secure SQL.

Blockchiffren

Blockchiffren sind wie Stromchiffren symmetrische Chiffrieralgorithmen. Im Unterschied zu den Stromchiffren findet die Verschlüsselung aber nicht bitweise statt, sondern in Blöcken fester Größe (üblich sind im Allgemeinen 64 Bits). Bei einer Blocklänge von 64 Bits werden also immer jeweils 64 Bits Klartext in 64 Bits Chiffretext transformiert. Bei einem Klartext, dessen Größe ungleich einem Vielfachen der Blockgröße ist, muss der letzte Block mit Bits aufgefüllt werden ("padding"). Dieses kann durch einfaches Auffüllen mit Nullen oder Einsen stattfinden. Somit hat der Chiffretext nicht unbedingt dieselbe Größe wie der Klartext, was auch ein weiterer Unterschied zu den Stromchiffren ist, wo Klartext- und Chiffretextgröße in jedem Fall gleich sind. Die unterschiedliche Größe von Chiffre- und Klartext ist eine wünschenswerte Eigenschaft, da in manchen Fällen die alleinige Kenntnis über die genaue Größe einer Nachricht Hinweise auf ihren Inhalt liefern kann.

Betriebsmodi von Blockchiffren: Blockchiffren können in verschiedenen kryptographischen Modi betrieben werden. Die verschiedenen Modi ermöglichen eine der jeweiligen Situation angepasste Nutzung der Chiffre, je nachdem ob es z.B. wichtiger ist, dass die Verschlüsselung schnell abläuft oder ob der Gefahr des Entfernens oder Einschleusens von fremden Daten begegnet werden soll. Es ist auch möglich, einen Blockchiffrieralgorithmus so zu benutzen, dass er wie eine Stromchiffre arbeitet, z.B. in Fällen, in denen es wichtig ist, dass Klar- und Chiffretext dieselbe Größe haben. Kryptographische Modi fügen einer Chiffre keine weitere Sicherheit zu; die Sicherheit beruht weiterhin ausschließlich auf dem verwendeten Algorithmus. Im Folgenden werden vier häufig verwendete Modi vorgestellt:

- ECB (Electronic Code Book): Jeder 64 Bits breite Klartextblock wird unabhängig voneinander mit dem gleichen Schlüssel verschlüsselt und entschlüsselt.
- CBC (Chipher Block Chaining): Der aktuelle Klartextblock wird vor der Verschlüsselung „XORed“ mit dem vorausgegangenen Ciphertextblock.

- CFB (Cipher Feedback): Dieses Verfahren erlaubt eine variable Blockgröße bei der Verarbeitung. Der vorausgegangene Ciphertext wird erneut verschlüsselt und die Ausgabe wird mit dem Plaintext in der gewünschten Länge „XORed“.

- OFB (Output Feedback): Funktioniert wie der Cipher Feedback Mode, benutzt aber statt des Ciphertextes nur die Ausgabedaten der Blockverschlüsselung. Die Folge der Daten des Schlüsselstroms für das Verodern ist hier unabhängig vom Plaintext.

Ausgewählte Blockchiffren: Es gibt eine Vielzahl an Blockchiffren, die wohl bekanntesten und zum Teil häufig eingesetzten sind:

- DES (Data Encryption Standard (1977)): Der DES-Algorithmus ist ein klassischer Vertreter der monoalphabetischen, polygraphischen Blockchiffrierungen. Das „National Bureau of Standards“ der U.S.A. entwickelte diesen Algorithmus in Zusammenarbeit mit IBM. DES ist eine 16-Runden Feistel-Chiffre, die mit Blockgrössen von 64 Bits arbeitet und eine Schlüssellänge von 64 Bits benutzt, wobei effektiv nur 56 Bits wichtig sind, da nach jedem siebten Bit ein Parity Information Bit folgt. Da im Laufe der Zeit, die 56 Bits als Schlüssel nicht mehr der Sicherheit genügten, verwendet man heute meist 3-fach DES (3DES). Hierbei wird der zu veschlüsselnde Block dreimal durch den Algorithmus geschickt und mit unterschiedlichen Schlüsseln abwechselnd ver- und entschlüsselt.

- AES (Advanced Encryption Standard (2000)): 1997 rief das „National Institute of Standards and Technology“ (NIST) dazu auf, kryptographische Algorithmen einzureichen, um einen Nachfolger für DES zu bestimmen, den AES. 15 Algorithmen wurden eingereicht. Auswahlkriterien waren vor allem Sicherheit, Performanz und Flexibilität. Im Jahre 2000 wurde dann ein Algorithmus von Joan Daemen und Vincent Rijmen, der Rijndael, ausgewählt. Der Rijndael-Algorithmus arbeitet mit variablen Blockgrößen und Schlüssellängen. Die beiden Parameter müssen ein Vielfaches von 32 Bits

sein. Zur Zeit wird die Sicherheit des AES-Algorithmus in Kryptologie Fachkreisen diskutiert[15].

- IDEA (International Data Encryption Algorithm (1992)): Der IDEA-Algorithmus ist eine Weiterentwicklung des PES (Proposed Encryption Standard). IDEA verschlüsselt den Klartext in Blöcken von 64 Bits und benutzt dabei einen 128 Bits großen Schlüssel. Die Verschlüsselung erfolgt in acht Runden und ist insgesamt etwa doppelt so schnell wie der DES. IDEA findet unter anderem in PGP (Pretty Good Privacy) Verwendung.

2.3.5 Asymmetrische Chiffren

Definition: Asymmetrische Chiffren verfolgen einen anderen Ansatz als die oben erläuterten symmetrischen Chiffren. Es wird nicht der gleiche Schlüssel sowohl zur Ver- wie auch zur Entschlüsselung verwendet, sondern es gibt bei asymmetrischen Chiffren zwei oder mehr verschiedene Schlüssel[16]. Im Fall von asymmetrischen Chiffren mit zwei Schlüsseln spricht man oft auch von *Publickey*

[15] Wurden Blockchiffren in der Kryptoanalyse bisher durch statistische Angriffe (differenzielle/lineare Kryptoanalyse) untersucht, so wird im Moment eine völlig neue Möglichkeit der Kryptoanalyse diskutiert.

Basierend auf dem XL-Algorithmus von N. Courtois, A. Shamir, J. Patarin und A. Klimov [CKPS00] , entwickelten die Mathematiker N. Courtois und J. Pieprzyk einen erweiterten Algorithmus XLS [CP02]. Das Besondere an diesem Verfahren ist, dass es sich um einen algebraischen Ansatz handelt. Zunächst wird der Blockalgorithmus als geschlossene Formel dargestellt. Dies hatten N. Ferguson, R. Schroeppel und D. Whiting 2001 für AES bewerkstelligt [FSW01]. Die geschlossene Formel beinhaltet eine Billiarde Summanden. Damit wird anschliessend ein quadratisches Gleichungssystem erstellt. Bei AES besteht dieses aus 8000 Gleichungen mit 1600 Variablen. Dieses Gleichungssystem hat bestimmte Eigenschaften:

- Es ist überbestimmt.
- Es ist schwach besetzt (die meisten Koeffizienten sind null)
- Es hat eine annähernd reguläre Struktur

Der Streitpunkt dieses Verfahrens ist die Möglichkeit des Lösens dieses Gleichungssystems. Sind nämlich z.B. nicht genug Koeffizenten des Gleichungssystem unabhängig, scheint das Verfahren nicht zu funktionieren. Und selbst wenn das Verfahren funktionieren sollte, ist es zur Zeit immer noch zu umfangreich, um diesen Angriff in der Praxis zu benutzen. Die Frage bleibt aber, ob durch diese neue Herangehensweise nicht noch effektivere Verfahren gefunden werden.

[16] In dieser Arbeit werde ich mich allerdings auf die Beschreibung asymmetrischer Chiffren mit zwei Schlüsseln beschränken.

Algorithmen, da üblicherweise ein Schlüssel veröffentlicht wird und ein Schlüssel geheim bleibt. Bei einer Kommunikation dient der öffentliche Schlüssel (*public key*) zum Chiffrieren der Daten und der geheime Schlüssel (*private key*) zum Dechiffrieren. So ist es möglich, dass jeder dem Besitzer des geheimen Schlüssels verschlüsselte Nachrichten zukommen lassen kann, ohne dass vorher ein gemeinsamer Schlüssel vereinbart werden musste. Man verwendet einfach den öffentlichen Schlüssel der betreffenden Person, der auch auf unsicheren Kanälen übermittelt werden kann und verschlüsselt die zu sendenden Daten mit diesem. Der so entstandene Chiffretext kann nur mit dem geheimen Schlüssel dechiffriert werden, der nur dem Empfänger bekannt sein darf. Damit dies auch funktioniert, muss gewährleistet sein, dass durch Kenntnis des öffentlichen Schlüssels und des Algorithmus keine Möglichkeit entsteht, den geheimen Schlüssel zu ermitteln. Mit anderen Worten: Es muss unmöglich (bzw. unverhältnismäßig schwer) sein, unter Kenntnis des öffentlichen Schlüssels den geheimen Schlüssel zu berechnen.

Die Rollen (öffentlich/privat) der Schlüssel sind nicht bei allen Algorithmen von vornherein festgelegt, häufig ist es egal, mit welchem Schlüssel ver- und mit welchem entschlüsselt wird.

RSA: Der wohl am weitesten verbreitete Publickey Algorithmus ist RSA[17]. Anders als bei symmetrischen Chiffren wie dem DES, die auf geschickter Bit-Ersetzung und Permutation beruhen, ist das Funktionsprinzip des RSA ein rein zahlentheoretisches. Seine Schlüssellänge ist variabel, wobei Größen zwischen 512 und 2048 Bits üblich sind. Der RSA ermöglicht sowohl eine Verschlüsselung wie auch eine Generierung von digitalen Signaturen (siehe 2.3.6) und hat sich weltweit zu einem Quasi-Standard für asymmetrische Verschlüsselung entwickelt.

Die Funktionsweise von RSA: Der Algorithmus des RSA ist verblüffend einfach, deswegen soll er kurz skizziert werden.

Schlüsselgenerierung:

1. Wähle zwei etwa gleich große Primzahlen p,q.

[17]Benannt nach seinen Entwicklern Ron Rivest, Adi Shamir und Leonard Adleman (1977).

2. Diese bestimmen den Modulus $n = p * q$.

3. Wähle Zahl e mit ggt$(e, (p-1)(q-1)) = 1$.

4. Wähle Zahl d mit $e * d \bmod (p-1)(q-1) = 1$.

5. e und n werden öffentlich gemacht und bilden den *public key*.

6. d, p und q bleiben geheim (p und q werden üblicherweise aus Sicherheitsgründen nach der Schlüsselerzeugung gelöscht). d bildet den *private key*.

Die Ver- und Entschlüsselung funktionieren wie folgt:
Verschlüsselung: $C = M^e \bmod n$
Entschlüsselung: $M = C^d \bmod n$
Die Reihenfolge der Anwendung der Schlüssel ist beliebig, was aus der Kommutativität der Multiplikation folgt ($M = C^d = (M^e)^d = M^{ed} = M^{de} = (M^d)^e = C'^e = M$). Dies ist jedoch eine spezifische Eigenschaft des RSA.

Andere Ansätze für asymmetrische Chiffren: Neben dem oben beschriebenen Ansatz für asymmetrische Chiffren, deren Sicherheit auf der Schwierigkeit des Faktorisierens großer Zahlen beruht, existieren weitere Ansätze für asymmetrische Chiffren, in denen andere, nur mit großem Aufwand zu lösende Probleme als Grundlage benutzt wurden:

- Berechnen des diskreten Logarithmus über endlichen Gruppen: Ähnlich wie das Problem des Faktorisierens ist das Berechnen des diskreten Logarithmus über endlichen Gruppen ein hartes Problem. Asymmetrische Chiffren, wie z.B. der ElGamal-Algorithmus, benutzen als Gruppe die ganzen Zahlen modulo einer Primzahl. Für diese Gruppe hat das Problem subexponentiellen Aufwand.

- Lösen von NP-Vollständigen Problemen: Es existieren mehrere Ansätze, NP-Vollständige Probleme für asymmetrische Chiffren zu verwenden. Dabei versucht man das Problem so zu modifizieren, dass es mit Hilfe einer zusätzlichen Information einfach zu lösen ist. Ohne die Information bleibt der Aufwand aber weiterhin NP-Vollständig. Man spricht in solchen Fällen

davon, dass man eine Hintertür in das Problem einbaut (trapdoor). Das modifizierte Problem bildet den öffentlichen Schlüssel, die Zusatzinformation den geheimen. Die zu verschlüsselnden Daten werden in eine Instanz (bzw. in eine Aufgabestellung) des NP-Vollständigen Problems transformiert. Um die Daten zurückzugewinnen, muss man das Problem lösen. Das Lösen des Problems ist ohne die Kenntnis der Zusatzinformation (also des geheimen Schlüssels) ab einer gewissen Größe des Problems unmöglich, bei Kenntnis der Information aber im Idealfall mit linearem Zeitaufwand zu erledigen. Leider wurde bis jetzt noch kein überzeugender Algorithmus dieses Ansatzes gefunden.

- Elliptische Kurven: Asymmetrische Chiffren, deren Sicherheit auf der Schwierigkeit des Lösens des diskreten Logarithmus über endlichen Gruppen beruht, benutzen als Gruppe i.A. die ganzen Zahlen modulo einer Primzahl. Diese Gruppe hat leider einige Besonderheiten, durch die das Berechnen des diskreten Logarithmus nur subexponentiellen Aufwand hat. Um ausreichende Sicherheit zu bieten, ist es nötig, relativ große Schlüssel zu verwenden. Dies wiederum erschwert den Einsatz solcher Chiffren in Anwendungen, die nur über beschränkten Speicherplatz verfügen, wie es z.B. bei Chipkarten der Fall ist. Darüber hinaus wird ein Zusammenhang zwischen dem Problem der Faktorisierung und dem des diskreten Logarithmus vermutet. Falls sich dieses als wahr herausstellt und außerdem ein Durchbruch im Lösen eines der beiden Probleme erzielt wird, werden mit einem Schlag fast alle verwendeten asymmetrischen Chiffren nutzlos.

  Mit den elliptischen Kurven, deren Punkte eine "Abelsche Gruppe" bilden, glaubt man, einen potenten Ersatz für die Gruppe der ganzen Zahlen modulo einer Primzahl gefunden zu haben. Das so gewonnene Problem hat exponentiellen Aufwand und kommt somit mit kleineren Schlüsseln aus. Darüber hinaus ist das Problem des Lösens des Logarithmus über einer elliptischen Kurve unabhängig vom Problem des Faktorisierens großer Zahlen. Weiter kommt positiv hinzu, dass die bekannten und bereits als sicher bewerteten Algorithmen, die momentan noch die Gruppe der ganzen Zahlen modulo einer Primzahl verwenden, auch mit der Gruppe der

Punkte einer elliptischen Kurve funktionieren. Der verbreiteten Benutzung dieses Ansatzes steht aber noch im Weg, dass die Schlüsselgenerierung für derartige Algorithmen ein noch nicht befriedigend gelöstes Problem ist.

Als Beispiel für einen Algorithmus, der erfolgreich mit elliptischen Kurven als Gruppe arbeitet, sei hier der *Elliptic Curve Digital Signature Algorithm (ECDSA)* (1992) genannt. Wie der Name schon vermuten lässt, handelt es sich hierbei um eine Variante des DSA (siehe 2.3.6). ECDSA wurde 1999 als ANSI-Standard (ANSI x9.62) akzeptiert.

2.3.6 Weitere notwendige Algorithmen

Einweg-Hashfunktionen: Ein wichtiges Werkzeug, das in vielen kryptographischen Protokollen Verwendung findet, sind *Einweg-Hashfunktionen*. Eine Hashfunktion H erzeugt aus einem beliebig großen Datenblock D einen eindeutigen Hashwert $h = H(D)$ mit fester Länge m. Hashfunktionen finden auch in anderen Bereichen der Informatik Verwendung. Um für kryptographische Anwendungen zweckmäßig zu sein, muss aber noch mehr von der Funktion gefordert werden: Zu einem gegebenen Datenblock D muss es einfach sein, den Hashwert h zu berechnen, aber die Umkehrung, aus einem Hashwert h' einen Datenblock D' zu generieren mit $h' = H(D')$, sollte „unmöglich“ zu berechnen sein. Daher kommt auch die Bezeichnung Einweg-Hashfunktion. Ebenfalls sollte es nicht möglich sein, zwei Datenblöcke D und D' mit demselben Hashwert $h = H(D) = H(D')$ zu finden. Daraus resultiert, dass auch kleinste Änderungen am Text zu unterschiedlichen Hashwerten führen müssen (Diffusion).

Authentisierung ohne Verschlüsselung mittels Einweg-Hashfunktionen: Einweg-Hashfunktionen ergeben eine einfache Möglichkeit zur Authentisierung von Kommunikationspartnern. Die Voraussetzung dafür ist, dass die an der Kommunikation beteiligten Parteien ein Geheimnis G teilen. Zur Erläuterung des Vorgehens betrachte man folgende Situation: Alice will Bob überzeugen, dass eine Nachricht M von ihr stammt. Um dieses zu tun, bildet Alice $h = H(M, G)$ also den Hashwert der Konkatenation der Nachricht und des Geheimnisses, wobei H eine Einweg-Hashfunktion ist, auf die sich Alice und Bob im Vorfeld geeinigt

haben und G das gemeinsame Geheimnis. Alice sendet h zusammen mit M an Bob. Bob, der G kennt, kann ebenfalls $H(M, G)$ bilden und das Ergebnis mit h vergleichen. Wenn die Werte übereinstimmen, weiß Bob, dass die Nachricht M wirklich von Alice geschickt wurde. Da nur er und Alice das Geheimnis G kennen, hätte niemand außer Alice den Hashwert h generieren können, und da H eine Einweg-Hashfunktion ist, ist es für Dritte auch trotz der Kenntnis von h und M unmöglich, G zu berechnen. Bob weiß außerdem, dass der Text M unverändert angekommen ist. Hätte eine dritte Partei die Nachricht M durch eine andere Nachricht M' ersetzt, wäre der resultierende Hashwert $h' = H(M', G)$ unterschiedlich zu dem mitgeschickten $h = H(M, G)$. Diese Art der Authentisierung findet u.a. bei IPSec Verwendung.

Der Nachteil dieses Verfahrens ist, dass es Dritten gegenüber keine Authentisierung bietet („non-repudiation"). Alice kann sich zwar von der Identität Bobs überzeugen, hat aber keine Möglichkeit einer dritten Person Clara zu beweisen, dass die Nachricht tatsächlich von Bob stammt, da Clara weder das Geheimnis G kennt, noch, bei Offenlegung des Geheimnisses, aus G auf Bob schließen kann.

Beispiele von Einweg-Hashfunktionen: Hier sollen zwei verbreitete Einweg-Hashfunktionen kurz erwähnt werden, die als weitgehend sicher gelten[18]:

- Secure Hash Algorithm, SHA1 (NIST, 1994): SHA1 ist als Teil des DSA (siehe 2.3.6) von der NSA[19] entwickelt worden und berechnet einen Hashwert von 160 Bits.

- RIPE-MD 160, Europäische Union: RIPE-MD 160 ist eine sichere Weiterentwicklung von MD4 und berechnet 160 Bits große Hashwerte.

[18]Eine noch nicht veröffentlichte Arbeit [WYY05] von chinesischen Kryptologen soll ein Verfahren beschreiben, das Kollisionen bei SHA-1 in nur 2^{69} Operationen findet, anstelle von 2^{80}. Diese Arbeit ist schon von mehreren bekannten Kryptoanalytiker, wie z.B. Bruce Schneier (`http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html`), bestätigt worden und die Forschungergebnisse in diesem Bereich sollten weiter verfolgt werden.

[19]Die „National Security Agency" (NSA) ist eine Unterabteilung des „Department of Defense" (DoD). Die NSA ist Teil des amerikanischen Geheimdienstes und wurde in den 1940ern geschaffen. (siehe auch: `http://de.wikipedia.org/wiki/NSA`

Digitale Signaturen: Für viele Kommunikationsprotokolle hätte man gerne ein elektronisches Äquivalent zur menschlichen Unterschrift, also eine Möglichkeit, digitale Dokumente zu unterzeichnen. Die exakten Anforderungen an eine sogenannte digitale Signatur lassen sich wie folgt auflisten:

- Authentizität: Der Empfänger kann sich von der Identität des Unterzeichners überzeugen (Es muss für jede Person problemlos möglich sein festzustellen, von wem die digitale Signatur erstellt wurde. Die digitale Signatur identifiziert ihren Ersteller eindeutig).
- Fälschungssicherheit: Nur dem Unterzeichner ist es möglich, die Signatur zu erzeugen.
- Überprüfbarkeit: Eine dritte Partei kann jederzeit die Signatur verifizieren.
- Keine Wiederverwendbarkeit: Die Signatur bezieht sich nur auf das unterzeichnete Dokument und kann keinesfalls für andere Dokumente verwendet werden.
- Keine Veränderbarkeit, Integrität: Nachdem das Dokument unterzeichnet ist, kann es nicht mehr verändert werden.

Es ist ersichtlich, dass, wenn alle diese Forderungen erfüllt sind, Vorgänge wie digitales Unterzeichnen von Verträgen und ähnliches möglich ist. Digitale Signaturen kann man weiterhin zur Authentisierung von Kommunikationspartnern oder Autoren digitaler Dokumente einsetzen.

Digitale Signaturen mittels asymmetrischer Chiffren: Vorausgesetzt die Reihenfolge der Anwendungen des geheimen und öffentlichen Schlüssels ist unerheblich, eignen sich asymmetrische Chiffren gut, um digitale Signaturen zu erstellen. Dabei wird wie folgt vorgegangen: Alice chiffriert das betreffende Dokument mit ihrem geheimen Schlüssel. Danach sendet sie das so unterzeichnete Dokument an Bob. Bob dechiffriert das Dokument mit Alices öffentlichem Schlüssel. Betrachten wir kurz, ob alle unsere Anforderungen an digitale Signaturen bei einem derartigen Vorgehen erfüllt werden:

- *Authentizität*: Da Alices öffentlicher Schlüssel den Klartext ergibt, weiß Bob, dass das Dokument von Alice stammt.
- *Fälschungssicherheit*: Nur Alice kennt ihren geheimen Schlüssel. Niemand anderes hätte die Signatur erstellen können.
- *Überprüfbarkeit*: Jeder, der Alices öffentlichen Schlüssel kennt, kann die Signatur bestätigen. Alice kann sie nicht zurücknehmen.
- *Keine Wiederverwendbarkeit*: Die Unterschrift bezieht sich nur auf dieses Dokument, andere Dokumente ergeben andere Signaturen.
- *Keine Veränderbarkeit, Integrität*: Bei Veränderung des Dokuments ergibt die Dechiffrierung mit Alices öffentlichem Schlüssel kein sinnvolles Ergebnis.

In der Praxis wird oftmals nicht das gesamte Dokument unterzeichnet, (also chiffriert) sondern nur der Hashwert des Dokuments. Dieses geschieht vor allem aus Gründen der Effizienz (außerdem bleibt so das Dokument selbst, auch ohne Überprüfung der digitalen Signatur, lesbar). Man kann sich davon überzeugen, dass bei Benutzung einer Einweg-Hashfunktion mit einem ausreichend großem Hashwert sich die Sicherheit der digitalen Signatur dabei nicht ändert.

Digitale Signaturen mittels DSA: Bei digitalen Signaturen mit asymmetrischen Chiffren geht mit der Signierung eine Verschlüsselung des Dokuments einher (falls das ganze Dokument und nicht nur der Hashwert signiert wird). Dies ist nicht in allen Fällen wünschenswert. Weiterhin muss man beachten, dass in manchen Ländern Benutzung oder Export von Software, die Verschlüsselung ermöglicht, restringiert ist. Also gibt es einen Bedarf nach Algorithmen, die digitale Signaturen ohne einhergehende Verschlüsselung ermöglichen. Der Digital Signature Algorithm (DSA), der 1991 von der NSA entwickelt wurde und Teil des DSS (Digital Signature Standard) ist, verfolgt einen derartigen Ansatz. Bei korrekter Implementation eignet er sich nur zum Signieren, aber nicht zum Verschlüsseln. Auch beim DSA wird mit öffentlichen und geheimen Schlüsseln gearbeitet und die Signatur wird auch hier unter Verwendung des öffentlichen Schlüssels überprüft.

Die Sicherheit des DSA beruht auf der Schwierigkeit, den diskreten Logarithmus über endlichen Körpern zu berechnen. Die Schlüssellänge ist variabel zwischen 512 und 1024 Bits. Wie auch schon beim DES wurde dem DSA bei seiner Einführung mit Misstrauen begegnet, da an seiner Entwicklung die NSA beteiligt war. Tatsächlich besteht bei maliziöser Implementation des Algorithmus die Möglichkeit, einen verdeckten Kanal in den DSA einzubauen, der nach und nach den geheimen Schlüssel des Anwenders nach außen schleust. Da ein solcher Kanal unmöglich zu entdecken ist, sollte man nur vertrauenswürdige Implementationen des DSA verwenden.

Weiteres wichtiges Zubehör

Für viele kryptographische Anwendungen und Protokolle benötigt man neben den bereits vorgestellten Werkzeugen wie Chiffren und Einweg-Hashfunktionen noch weiteres Handwerkszeug.

- *Zufallszahlengeneratoren*: Als erstes seien hier die kryptographisch starken Zufallszahlengeneratoren genannt. In vielen Anwendungen und Protokollen werden Schlüssel zur Datensicherung oder Kommunikation aus Zufallszahlen gewonnen. Für diese Fälle benötigt man einen Zufallszahlengenerator, dessen Ausgaben nicht vorhersehbar sind. Genauer gesagt, muss es unmöglich sein, ohne Kenntnis des Initialwertes des Generators, ausschließlich aus Kenntnis der vorherigen Zufallszahlen und des Algorithmus, den nächsten Zufallswert vorherzusagen. Falls ein Algorithmus Verwendung findet, der dieses nicht leistet, erschließt sich, z.B. in einer Situation, in der die Zufallszahl zur Schlüsselgenerierung benutzt wird, ein möglicher Angriffspunkt auf die Verschlüsselung. Anstatt die Chiffre oder den Schlüssel angreifen zu müssen, können potentielle Angreifer versuchen, den verwendeten Schlüssel zu „erraten“. Falls sie in Besitz von vorher verwendeten Schlüsselinformationen sind und der Zufallszahlengenerator noch nicht neu initialisiert wurde, ist ein derartiges Vorgehen äußerst vielversprechend.

- *Synchronisierte Uhren*: Viele kryptographische Protokolle benutzen Zeitstempel, um Angriffe, die durch das Wiedereinspielen von abgefangenen

Daten stattfinden, abzuwehren. Die einzelnen Nachrichten eines Protokolls werden mit einem Zeitstempel, der Informationen über Datum und Uhrzeit enthält, versehen. Wenn eine Nachricht eintrifft, deren Zeitstempel älter als eine festgelegte Schranke ist, wird diese Nachricht nicht mehr akzeptiert. Ein solches Verfahren ist nur möglich, wenn die Teilnehmer an dem Protokoll über synchronisierte Uhren verfügen. Doch dieses ist kein triviales Unterfangen, besonders bei Kommunikation, die über die Grenzen eines kleineren Netzwerkes hinausgeht.

- *Hochauflösende Zeitstempel*: Die Notwendigkeit für hochauflösende Zeitstempel hängt eng mit dem Punkt der synchronisierten Uhren zusammen. Viele Betriebssysteme liefern nur sehr grob gemessene Zeitangaben. Für manche Protokolle ist es aber nötig, Unterschiede in Zeitstempeln, die in den Bereich von Zeiträumen unter einer Nanosekunde fallen, zu erkennen. Dafür reichen dann oft die vom Betriebssystem zur Verfügung gestellten Hilfsmittel nicht aus.

2.3.7 Kryptographische Protokolle

Ein kryptographisches Protokoll ist ein Kommunikationsprotokoll, das mindestens ein Sicherheitsziel wie beispielsweise Authenzität, Vertraulichkeit oder Integrität der übertragenen Daten gewährleisten kann. Es definiert eindeutig eine festgelegte Abfolge von Handlungen der Kommunikationspartner. Dabei wird, je nach Aufgabe des Protokolls, bestimmt, welches Format die verschickten Nachrichten haben, welche Algorithmen verwendet werden, wie die Schlüssel ausgetauscht werden und ähnliches. Dabei finden üblicherweise mehrere kryptographische Algorithmen Verwendung. Man findet kryptographische Protokolle in allen Schichten der TCP/IP-Familie. Es seien als Beispiele genannt, *TLS* und *SSL* in der Transportschicht und *SSH*[20], *Kerberos* und *PGP*[21] in der Anwendungsschicht.

[20] „Secure shell“ (SSH) ist sowohl ein Programm als auch ein Netzwerkprotokoll für das „remote“ Einloggen und Verwalten von Rechnern.

[21] „Pretty Good Privacy“ ist ein Programm zum Ver- und Entschlüsseln von Daten. Es wird vor allem bei der Emailkommunikation eingesetzt.

Kapitel 3

Darstellung der verschiedenen Single Sign-On Lösungen

Unabhängig, ob verteilte Systeme im Internet oder in einem Intranet benutzt werden, benötigt im Allgemeinen der Benutzer eine Menge von Authentisierungszeugnissen[1]. Diese bestehen meist aus Benutzernamen mit zugehörigem Passwort, können aber auch andere Authentisierungsinformationen enthalten. In der Regel wird für jeden Dienst[2] eine erneute Anmeldung mit einem eigenen Authentisierungszeugnis notwendig. Die ansteigende Zahl von Identifikationsmerkmalen und zugehörigen Authentisierungsinformationen soll **Single Sign-On** (SSO) verwaltbar machen. Gewünscht wird aus Benutzersicht, eine einmalige Anmeldung auszuführen und anschliessend, die weiteren Authentisierungen transparent für den Benutzer ablaufen zu lassen. Dieser Ansatz ist nicht neu und wird schon seit Jahren innerhalb von Firmen verfolgt. Waren die ersten SSO-Systeme noch auf eine homogene IT-Landschaft mit einem exklusiven Betriebsystem und eingeschränkter Auswahl von Anwendungsapplikationen begrenzt, so sollen die neueren Systeme sogar Firmengrenzen überwinden und die unterschiedlichsten Anforderungen erfüllen.

[1] engl.: authentication credentials

[2] Als Dienst wird in diesem Zusammenhang die Bereitstellung von Serviceleistungen oder Content angesehen.

3.1 Klassifizierung von Single Sign-On Systemen

Basierend auf den Arbeiten von Jan De Clercq [Cle02] sowie Andreas Pashalidis und Chris J. Mitchell [PM03] wird zunächst eine Klassifikation der unterschiedlichen Ansätze von Single Sign-on vorgestellt. Die Grundidee der Single Sign-On Systeme, besteht darin, dass sich jemand einmalig an einer bestimmten Stelle anmeldet. Nutzt er anschliessend Dienste, welche eine Zugangsberechtigung aufweisen, so sollen diese mit einer Authentisierungstelle kommunizieren, um festzustellen, ob der Benutzer eine Zugangsberechtigung besitzt.

Die „Open Group“ definiert Single Sign-On in ihrer Spezifikation "X/Open Single Sign-On Service (XSSO) Pluggable Authentication" [The97] als:

> „Single sign-on (SSO) is mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where he has access permission, without the need to enter multiple passwords. Single sign-on reduces human error, a major component of systems failure and is therefore highly desirable but difficult to implement.“[3]

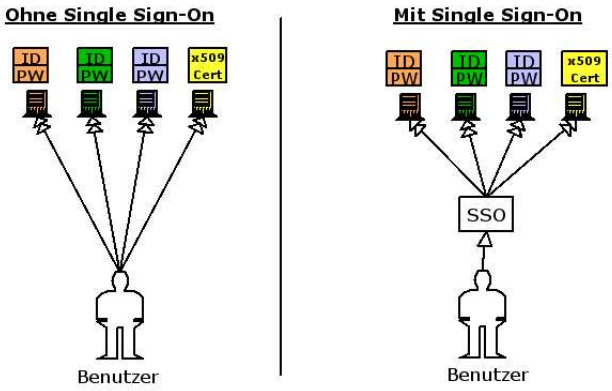

Abbildung 3.1: Single Sign-On

[3] http://www.opengroup.org/security/sso/

3.1.1 SSO Akteure

Zunächst werden die Teilnehmer einer solchen Authentisierungsinfrastruktur bestimmt. Ein Single Sign-On Szenario besteht hauptsächlich aus drei Teilnehmergruppen. Anhand deren Anzahl und Interaktionen lassen sich abstrakte Verbunde beschreiben.

User/Benutzer [4] Betitelt den Teilnehmer, der einen oder mehrere Dienste nutzen möchte. Dienste benötigen nicht zwingend eine Authentikation. Im Sinne von Single Sign-On sind aber gerade diese Dienste von Interesse für diesen Teilnehmer. Da Single Sign-On ihm in der Ausübung seiner Netzwerkidentitäten unterstützen soll.

Service Provider (SP) An diesen Stellen werden für den Benutzer Dienste angeboten. Diese Dienste können eine Authentisierung erfordern.

Authentication Service Provider (ASP) Dies sind die konkreten physischen Rechner, welche die Authentisierungsverwaltung betreiben.

Authentication authority (AA) Dies ist ein logischer Verbund von **ASP** und **SP**, die zusammenarbeiten. Natürlich kann eine **AA** zur Steigerung der Verfügbarkeit mehrere **ASP**s inklusive redundanter Authentisierungszeugnisdatenbanken benutzen. **AA** bilden somit logische Einheiten in den Single Sign-On Systemen.

Authentication Infrastructure (AI) Ein Verbund der unterschiedlichen „Authentication Authorities“ zu einer erweiterten Infrastruktur, wird „Authentication Infrastructure“ genannt. Somit kann eine **AI** mehrere dieser **AA**s beinhalten. Obwohl die einzelnen **AA**s unabhängige Verwaltungsobjekte und Hardwareausstattungen abbilden, besteht innerhalb der **AI** für den Benutzer der Eindruck, als ob er sich in einer einzigen **AA** bewegt. Dies wird durch zusätzliche Vertrauensbeziehungen oder dem Austausch von Authentikationszeugnissen zwischen den **ASP**s in den unterschiedlichen **AA**s einer **AI** erreicht.

[4] Es werden teilweise die englischen Begriffe in dieser Arbeit benutzt, damit keine Missverständnisse zwischen den Bezeichnungen in dieser und der angegeben Literatur entstehen.

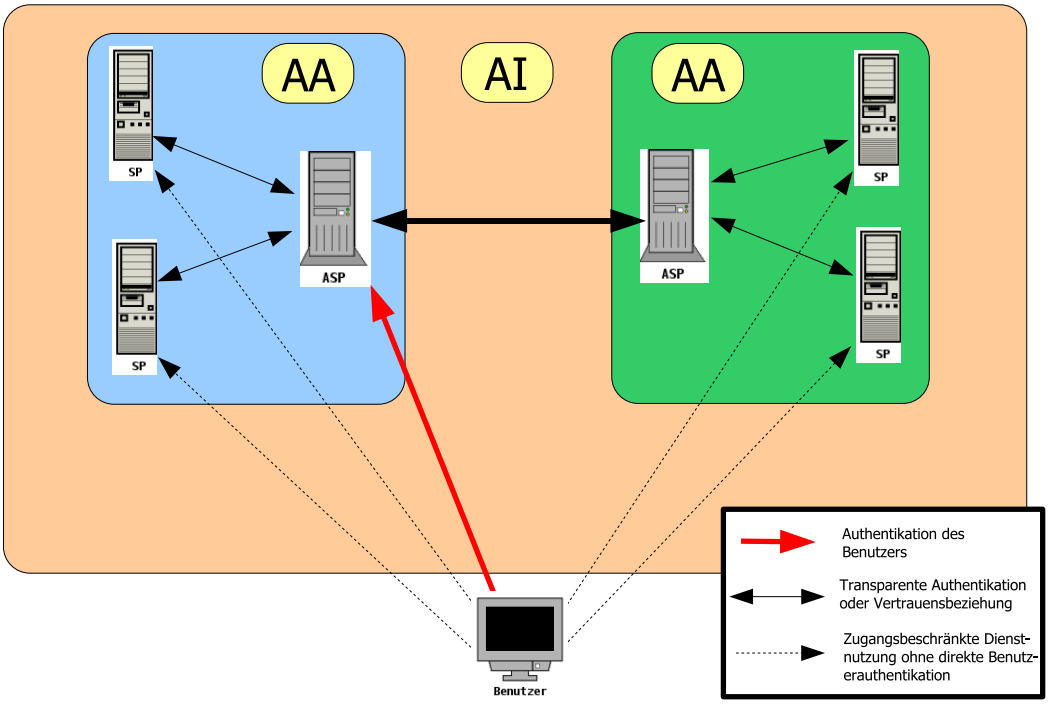

Abbildung 3.2: Verbunde bei Single Sign-On

Es folgt eine Einteilung in vier unterschiedliche Ausprägungen von Single Sign-On Systemen. Anhand dieser Einteilung können später die zu untersuchenden Systeme klassifiziert werden.

3.1.2 Local Pseudo Single Sign-On

Als einfachste Lösung ergeben sich sogenannte Local Pseudo Single Sign-On Systeme. Bei diesen verwaltet eine Software auf dem lokalen Rechner die unterschiedlichen Authentisierungszeugnisse. Der Benutzer authentisiert sich einmalig (primäre Authentikation) bei der lokalen Verwaltungssoftware und anschließend veranlasst die Verwaltungssoftware das Heraussuchen und Übertragen der Anmeldungsdaten. In diese Kategorie fallen z.B. die Passwortverwalter[5] in Webbrowsern. Es versteht sich von selbst, dass diese Verwaltungssoftware eine Möglichkeit besitzen muss, in die Kommunikation zwischen Benutzer und **SP** interagieren zu können, um Single Sign-on zu realisieren. Es handelt sich hierbei um

[5]Abhängig von der URL füllt der Webbrowser automatisch die Formularfelder mit den notwendigen Daten aus, um eine Authentisierung zu bewerkstelligen.

ein *n:1* System. Für jeden zu benutzenden Dienst wird ein Authentisierungszeugnis in der Vewaltungssoftware hinterlegt. Da es vorkommen kann, dass ein Benutzer unterschiedliche Accounts für einen Dienst hat, können unterschiedliche Authentisierungszeugnisse für denselben Dienst hinterlegt werden. Zwingende Voraussetzung für dieses Verfahren ist, dass der Benutzer dem Rechner vertrauen muss, auf dem die Verwaltungssoftware läuft. Ausserdem ist der Benutzer in der Regel auf den immer gleichen Rechner angewiesen, da der Rechner die Verwaltung und Speicherung seiner Authentisierungszeugnisse betreibt. Um eine erhöhte Mobilität zu erreichen, könnte man die Verwaltungssoftware inklusive der Authentisierungszeugnissen auf eine Smartcard oder ähnliches auslagern.

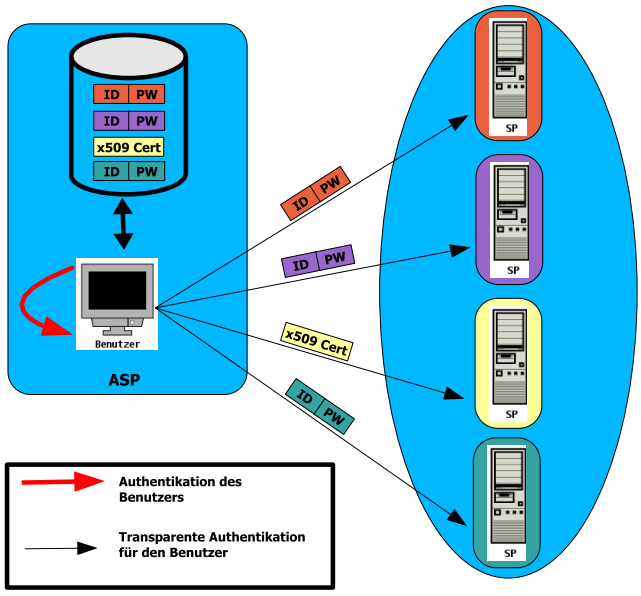

Abbildung 3.3: Local Pseudo Single Sign-On

3.1.3 Proxy-based Pseudo Single Sign-On

Bei dieser Variante erfolgt das Authentisierungsverfahren wie bei den Local Pseudo Single Sign-On Systeme, mit dem Unterschied, dass die Verwaltungssoftware nicht mehr lokal auf dem Rechner des Benutzers läuft, sondern auf einen Server im Netzwerk ausgelagert ist. Durch die Auslagerung der Authentisierungszeugnisse mit der Verwaltungssoftware, besteht für den Benutzer die Möglichkeit, seine Anmeldedaten von unterschiedlichen Rechnern zu benutzen. Der Benutzer muss dem Server vertrauen, und die primäre Authentikation findet zwischen dem Benutzer und dem Proxy Servers statt. Will der Benutzer einen **SP** nutzen, so muss die Authentikationskommunikation vom Proxy abgefangen oder zu ihm umgeleitet werden. Entscheidend bei dieser Variante ist, dass die Authentisierungszeugnisse zu keiner Zeit auf dem Benutzer-PC liegen. Die Authentikation bei einem **SP** findet zwischen Proxy und **SP** statt. Zu bemerken ist, dass aus Sicht eines **SP** nicht erkennbar ist, dass eine solche Single Sign-On Lösung verwendet wird. Es besteht kein Vertrauensverhältnis zwischen Proxy und **SP**.

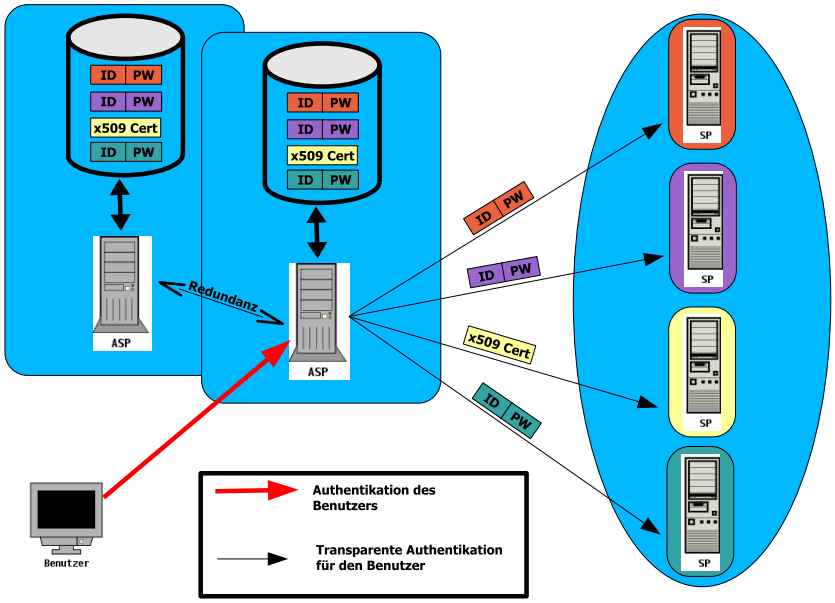

Abbildung 3.4: Proxy-based Pseudo Single Sign-On

3.1.4 Local True Single Sign-On

Im Unterschied zu den vorherigen beiden Single Sign-On Architekturen wird bei den nachfolgenden Architekturen mit wirklichen **Vertrauensbeziehungen** zwischen den Service Providern (**SP**) und dem Authentication Service Provider (**ASP**) gearbeitet. Diese müssen vor der Benutzung von Anwendern des Single-Sign-On Dienstes definiert werden und eventuell sogar vertraglich niedergeschrieben sein. Der hauptsächliche Unterschied ist nun, dass eine direkte Authentisierung nur zwischen dem Benutzer und dem **ASP** erfolgt. Die **SP** werden über den Status der Anmeldung über sogenannte Authentication Assertions informiert. Die Authentication Assertions beinhalten die Single Sign-On Benutzeridentität und den Authentisierungstatus an dem **ASP**. Im Gegensatz zum PSEUDO SINGLE SIGN-ON haben wir hier eine *n:m* Beziehung. Der Benutzer kann nicht nur unterschiedliche SSO Identitäten für einen Dienst wählen, sondern auch eine konkrete SSO-Identität mit mehreren **SP** teilen. Daraus ergibt sich die Möglichkeit für den Benutzer unterschiedliche **Rollen** zu benutzen. Da eine echte SSO-Lösung nicht eine erneute Authentikation mit dem **SP** bewerkstelligt, sondern der **SP** nur über den Status der Authentikation zwischen dem Benutzer und dem **ASP** informiert wird, muss ein Vertrauensverhältnis zwischen dem **ASP** und den **SP**s bestehen. Somit muss bei einem lokalen Betreiben eines **ASP** auf dem Rechner des Benutzers dafür gesorgt werden, dass die **SP**s ein Vertrauensverhältnis mit diesem **ASP** besitzen. Um dies zu erreichen, bedarf es einer gesicherten Komponente auf dem Rechner des Benutzers. Pashalidis und Mitchell beschreiben eine Möglichkeit mit Hilfe der Trusted Computing Platform Alliance[6] (TCPA) in [PM03].

[6] TCPA ist eine Spezifikation für einen Kryptoprozessor inklusive Speicherbereich, auf den der Benutzer keinen direkten Zugriff besitzt, siehe `http://www.trustedcomputing.org`

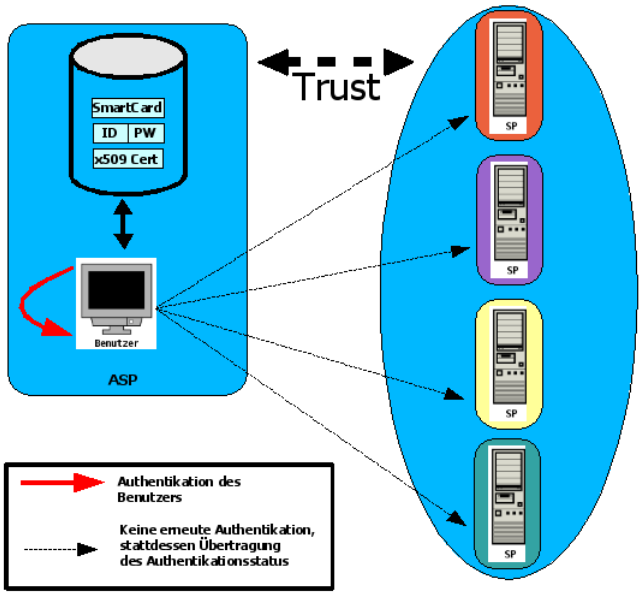

Abbildung 3.5: Local True Single Sign-On

3.1.5 Proxy-based True Single Sign-On

Das Vertrauenskonzept bleibt wie bei den LOCAL TRUE SINGLE SIGN-ON Systemen bestehen, nur der **ASP** wird jedoch nun ausgelagert. Erneut authentisiert sich der Benutzer nur an einer Stelle und zwar beim **ASP**. Alle nachfolgenden notwendigen Autorisierungen finden nicht mehr explizit mit dem Benutzer statt. Es werden Authentication Assertions zwischen **ASP** und **SP** ausgetauscht, um die Legitimität für die Verwendung des Dienstes seitens des Benutzers nachzuweisen. Durch das Auslagern des **ASP** ensteht die Möglichkeit, mehrere **ASP**s zur Steigerung der Verfügbarkeit vorzuhalten oder sogar mit unterschiedlichen **AA**s eine zusätzliche Infrastruktur aufzubauen. Durch die Bildung von Kommunikationskanälen zwischen den unterschiedlichen **AA**s erhält man eine **AI**.

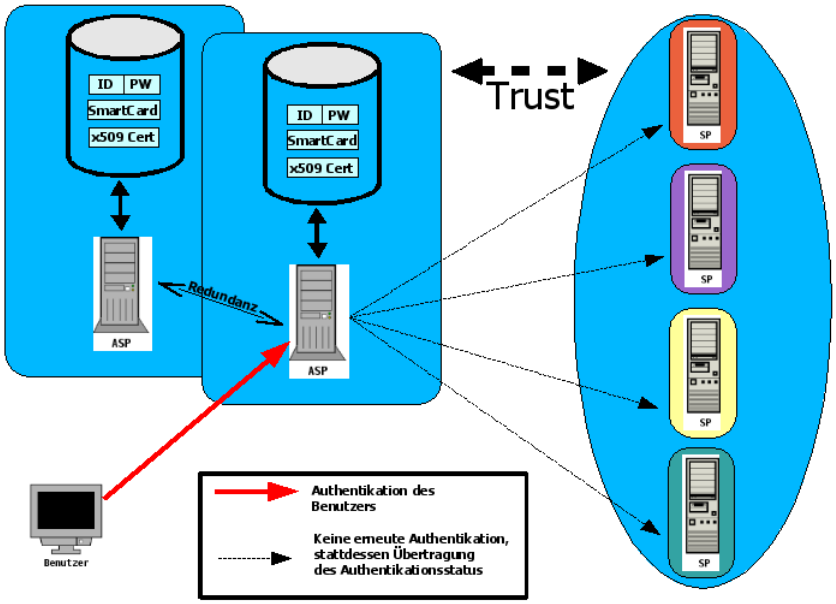

Abbildung 3.6: Proxy-based True Single Sign-On

3.1.6 Sonstige Eigenschaften von SSO

In der Literatur findet sich ein weiteres System, welches teilweise als Single Sign-On klassifiziert wird. Hier handelt es sich um Passwort-Synchronisationssysteme[7]. Diese System habe ich hier nicht aufgeführt, da es nach meinem Verständnis keine Single Sign-On Lösung ist. Denn eine explizite Anmeldung muss erneut bei jedem Dienst von dem Benutzer erfolgen. Die Vereinfachung für den Benutzer besteht darin, dass er nicht mehrere Authentisierungsinformationen vorhalten muss, sondern mit einer Authentisierungsinformation Zugang bekommt. Der Authentisierungsvorgang bleibt bei diesem System identisch mit den nicht Single Sign-On Systemen.

Auch wenn sich Single Sign-On hauptsächlich mit der Authentisierung befasst, überschneidet es sich mit dem Bereich des Identity Managments [JtM01]. Neben Authentisierungsinformationen werden nämlich unter Umständen noch zu-

[7]Bei Passwort-Synchronisationssystemen werden die Zugangsdaten bei unterschiedlichen **SP**s mit einem Master-Zugangsdatum abgeglichen. Ändert sich das Master-Zugangsdatum, so wird diese neue Zugangsinformation auf alle beteiligten, ansonsten autonomen **SP**s repliziert.

sätzlich Profil- und Identitätsinformationen übermittelt. Hierbei spielt es gerade in der Risikoabschätzung eine Rolle, wie beherrschbar und transparent die Daten des Benutzers verwaltet und übertragen werden.

Single Sign-On lässt sich zudem einteilen in zentralisierte und föderale Systeme.

Zentralisierte Single Sign-On Systeme: Hierbei gibt es einen zentralen **ASP**. Dieser **ASP** kann zwar repliziert redundant vorkommen in der **AA**, entscheidend ist aber, dass es nur eine **AA** gibt. Es fehlt bei solchen Systemen die Möglichkeit, eine **AI** aufzubauen.

Föderale Single Sign-On Systeme: Durch Erweiterung von Vertrauensbeziehungen zwischen den **AA**s lassen sich komplexe **AI**s bilden. Es besteht die Möglichkeit, Benutzerinformationen auf die unterschiedlichen **AA**s zu verteilen und Regeln zu definieren, wie der Austausch von Informationen erfolgen soll. Auch der Informationsinhalt, der zwischen den **AA**s ausgetauscht wird, kann reguliert werden. Diese Systeme führen eine zusätzliche Komplexität in Single Sign-On Systemen ein.

Eine weitere Unterscheidung lässt sich anhand der Authentication Assertions vornehmen

Tokenbasierende Authentication Assertions: Hier werden für den Nachweis der Autorisierungtokens benutzt. Also ein Merkmal, welches den Kommunikationspartnern ein Vertrauensverhältnis bestätigt und zwischen diesen abgesprochen sein muss. Im Allgemeinen bedarf es bei den tokenbasierten Authentication Assertions einer gemeinsamen Verwaltung zwischen den **AA**s und den **ASP**s.

Zertifikatsbasierende Authentication Assertions: Bei diesen Authentication Assertions kommt die Publickey Kryptographie zum Einsatz. Dadurch wird es möglich die **AA**s und **ASP**s autonomer zu verwalten. Es bedarf keines gemeinsamen Geheimnisses mehr, um den Nachweis der Authentizität zu gewährleisten.

3.2 Kerberos v5

Kerberos[8] wurde ursprünglich am Massachusetts Institute of Technology (MIT) für das Projekt Athena, siehe [Bry88] und [SNS88], entwickelt. Ziel war es, ein Authentisierungsprotokoll mit einer vertrauenswürdigen dritten Partei für TCP/IP-Netze zu entwickeln. Kerberos beruht auf symmetrischer Kryptographie und teilt sich mit jeder Einheit im Netz einen anderen geheimen Schlüssel. Die Kenntnis dieses Schlüssels dient als Identitätsbeweis. Kerberos Version 4 war die erste Veröffentlichung. Nachdem ein paar Sicherheitsprobleme bekannt wurden und der Dienst auch auf andere Umgebungen als bei Athena eingesetzt werden sollte, flossen ein paar Nachbesserungen in die Version 5 [KN93]. Kerberos ist besonders in geschlossenen Netzsystemen im Einsatz, ist also vor allem für die firmeninterne Verwendung entwickelt worden.

3.2.1 Aufbau von Kerberos V5

Kerberos basiert auf dem Protokoll von Needham-Schroeder [NS78], mit vertrauenswürdiger dritter Partei. Die zugrundeliegende **Authentication Authority** besteht bei Kerberos aus folgenden Teilnehmern:

Benutzer: Hier handelt es sich um eine Instanz, welche einen Service im Netzwerk nutzen möchte. Diese Instanz muss nicht zwingend eine Person sein, sondern es kann sich auch um einen Service handeln, der einen anderen Service konsultieren muss, um seine Aufgabe zu erledigen. Ein Beispiel wäre ein ausgelöster Druckauftrag, bei dem der Druckservice Zugang zu einem Fileserver benötigt, um die erforderlichen Daten zu bekommen, damit er den Auftrag an den Drucker leiten kann.

Service Provider Hierbei handelt es sich um die Services, welche in dem Netzwerk angeboten werden und eine voherige Authentisierung benötigen, bevor diese angefordert werden dürfen. **SP** können ihrerseits wiederum **Benutzer** von anderen Diensten in einem Kerberos Netzverbund sein.

[8] Aus Wikipedia.de: "Kerberos ist in der griechischen Mythologie der dreiköpfige Höllenhund, der den Eingang zur Unterwelt bewacht". Das Kerberos Protokoll sollte ursprünglich drei Komponenten beinhalten um ein Netzwerk zu beschützen: Authentication, Accounting und Audit. Die letzten beiden wurden nie implementiert.

Authentication Service Provider werden in Kerberos aus zwei unterschiedlichen Diensten aufgebaut. Der eine Dienst ist für die primäre Authentisierung zuständig, der andere stellt nach erfolgreicher Authentisierung die notwendigen Tickets (Tokens) aus, damit ein Benutzer verschiedene Dienste im Netzwerk verwenden kann. Die **ASP**[9] besteht somit aus folgenden Diensten:

Authentication Server (AS): Dieser Dienst verwaltet die Benutzerpasswörter in einer Datenbank und überprüft zu Beginn einer Kerberossitzung die Identität des Benutzers. Zusätzlich kennt dieser Dienst die notwendigen Daten für die Kommunikation mit den **TGS** und kann bei erfolgreicher Verifizierung dem Benutzer den Zugang zum **TGS** ermöglichen. Bei einer Kerberossession braucht ein Benutzer sich nur einmalig mit dem **AS** zu unterhalten.

Ticket Granting Service (TGS): Der **TGS** übernimmt die Ausgabe von Servicetickets für den Benutzer und ermöglicht somit die Nutzung von Diensten in einem Kerberosverbund. Der **TGS** authentisiert den Benutzer nicht erneut, sondern verlässt sich hierbei auf den **AS**. Die rechtmäßige Benutzung des **TGS** wird über ein Token realisiert, welches der **AS** dem Benutzer übermittelt.

Kerberos Realm: Eine vollständige Kerberosumgebung besteht aus den oben genannten Servern (**AS** und **TGS**), Clients, sowie Service Provider Servern. Der **AS** muss die User-ID und dass gehashte Passwort von allen teilnehmenden Benutzern in seiner Datensammlung vorhalten. Jeder neue Benutzer muss zunächst am **AS** registriert werden. Ausserdem braucht der **AS** die geheimen Schlüssel der **TGS**, um die Ticket Granting Tickets ausstellen zu können. Die **TGS** wiederum müssen sämtliche geheimen Schlüssel der Service Provider besitzen, damit sie die Servicetickets austellen können. Diese Umgebung wird bei Kerberos **Realm** genannt. Netzwerke von Clients und Server unter unterschiedlichen Administrationen haben meist unterschiedliche Realms.

[9] Diese beiden Dienste werden in der Literatur häufig zusammengefasst als Kerberosserver

Der schematische Ablauf einer Kerberos Session besteht somit aus dem Anfordern eines Tickets für den **Ticket Granting Service**. Nachdem der Benutzer dieses erhalten hat, kann er beim **TGS** ein Ticket für einen Dienst erbitten. Sobald er das Ticket für den gewünschten Dienst besitzt, hat der Benutzer die Möglichkeit, den gewünschten Dienst zu nutzen.

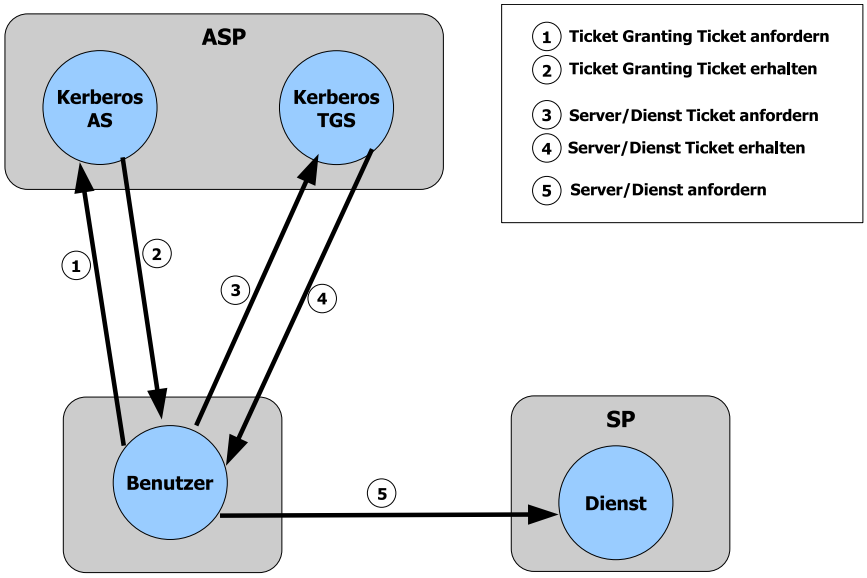

Abbildung 3.7: Kerberos Aufbau

3.2.2 Authentisieren und Anfordern des Ticket Granting Tickets

Zunächst muss der Benutzer eine Kommunikation mit dem **Authentication Server** eingehen. Hierbei wird die primäre Authentikation vorgenommen. Die Verifizierung der Identität eines Benutzers, der an einem Kerberosumgebung teilnehmen möchte, erfolgt über ein Passwort, welches dem Benutzer bekannt ist und in der Regel am Anfang einer Sitzung an der Workstation über ein Eingabefenster abegfragt wird.

Der Benutzer sendet eine Nachricht mit seinem Namen und dem Namen des **TGS**-Servers, sowie weiteren Informationen an den Kerberosauthentikationsserver **AS**.

C → AS:

$$Options||ID_C||Realm_C||ID_{TGS}||Times||Nonce_1$$

- $Options$: Hiermit können bestimmte Optionen für das zu erwartende Ticket beantragt werden. Die möglichen Optionen werden im Unterkapitel 3.2.5 bei der Flagsbeschreibung erläutert.
- ID_C: Die Benutzeridentität, in der Regel ein Benutzername.
- $Realm_C$: Gibt den Realm des Benutzers an.
- ID_{TGS}: Bestimmt, für welchen **TGS** ein Ticket benötigt wird.
- $Times$: Beantragte Zeitfenster. Unterteilt sich in:
  1. Start: Zeitpunkt, ab wann das Ticket gültig sein soll.
  2. Ende: Zeitpunkt, ab wann das Ticket ungültig werden soll.
  3. Intervall: Zeitpunkt, ab wann das Ticket erneuert werden soll.
- $Nonce_1$: Ein Zufallswert, der in der Antwort vom **AS** wiederholt werden muss, um sicherzustellen, dass die Antwort aktuell ist.

Der **AS** sendet ein Ticket Granting Ticket, Identifikationsinformationen und einen Informationsblock verschlüsselt mit dem Hashwert des Benutzerpasswortes. Somit braucht das Benutzerpasswort nicht über dass Netzwerk übertragen zu werden. Nur wenn der Benutzer das richtige Passwort kennt, kann er die notwendigen Daten entschlüsseln, welche Ihm einen Zugang zum **TGS** ermöglichen.

AS → C:

$$Realm_C||ID_C||Ticket_{TGS}||E_{K_C}[K_{C,TGS}||Times||Nonce_1||Realm_{TGS}||ID_{TGS}]$$

wobei $Ticket_{TGS}$ aus folgenden Komponenten besteht

$$Ticket_{TGS} = E_{K_{TGS}}[Flags||K_{C,TGS}||Realm_C||ID_C||AS_C||Times]$$

- $E_{K_C}[...]$ bezeichnet die Verschlüsselung der Daten in den eckigen Klammern mit dem Schlüssel K_C. In diesem Falle ist der Schlüssel, der Hashwert des Benutzerpasswortes.

- $K_{C,TGS}$ ist der neu generierte Sitzungsschlüssel für die Kommunikation zwischen dem Benutzer und dem **TGS**.
- $Times$ sind die übernommenen Zeitwerte aus der Anfrage des Clients, oder aber, falls diese nicht in einem gültigen Bereich liegen, neu vorgegebene Zeitwerte vom **AS**.
- $Realm_{TGS}$ ist der zugehörige Realm für den **TGS**.
- $E_{K_{TGS}}[...]$ bezeichnet die Verschlüsselung der Daten in den eckigen Klammern mit dem Schlüssel K_{TGS}. Dieser Schlüssel wurde zuvor beim **AS** hinterlegt und ändert sich nicht.
- $Flags$ Die Erklärung zu diesem Feld erfolgt später im Abschnitt 3.2.5.
- AD_C enthält die Netzwerkadresse des Benutzers bei der initialen Anfrage an den **AS**. Dies dient zum Schutz vor Einspielungen der Tickets von anderen Rechnern aus.

Somit ist die Kommunikation mit dem **AS** abgeschlossen und wird auch erst bei einem erneuten Einloggen in das Kerberossystem oder nach Ablauf der vorgegebenen Zeitwerte wieder notwendig. Hat der Benutzer das richtige Passwort, so ist er im Besitz des Sitzungsschlüssels für die Kommunikation mit dem **TGS**. Zudem besitzt er ein Token, welches er zwar nicht entschlüsseln kann, dass ihn gegenüber dem **TGS** aber ausweisen soll. Somit folgt nun die Kommunikation mit dem **TGS**.

3.2.3 Kommunikation mit dem Ticket Granting Server und Ausstellung der Service Granting Tickets

Will der Benutzer nun einen Serverdienst benutzen, so braucht er ein Service Granting Ticket für den gewünschten **SP** vom **TGS**. Dazu schickt er folgendes Paket an den **TGS**:

C → TGS:

$$Options||ID_{SP}||Times||Nonce_2||Ticket_{TGS}||Authenticator_C$$

wobei $Authenticator_C$ folgendermassen aufgebaut ist

$$Authenticator_C = E_{K_{C,TGS}}[ID_C||Realm_C||TS_1]$$

- ID_{SP} gibt den gewünschten **SP** an, bei dem der Benutzer eine Serviceleistung nutzen möchte.
- $Nonce_2$ ist wie bei der Kommunikation mit dem **AS** ein Zufallswert, der bei der Antwort vom **TGS** wiederholt wird um Replay Attacken zu verhindern.
- TS_1 ist die aktuelle Zeit beim Benutzer, damit der **TGS** erkennen kann, dass die Anfrage nicht veraltet ist.

Der **TGS** entschlüsselt das $Ticket_{TGS}$ und kommt somit in den Besitz des geheimen Schlüssels $K_{C,TGS}$. Mit diesem Schlüssel kann er den $Authenticator_C$ dechiffrieren. Der **TGS** überprüft die notwendigen Daten und Verbindungsinformationen und stellt bei erfolgreicher Verifizierung ein Antwortpaket zusammen.

TGS → C:

$$Realm_C||ID_C||Ticket_{SP}||E_{K_{C,TGS}}[K_{C,SP}||Times||Nonce_2||Realm_{SP}||ID_{SP}]$$

dabei ist $Ticket_{SP}$ von folgender Gestalt

$$Ticket_{SP} = E_{K_{SP}}[Flags||K_{C,SP}||Realm_C||ID_C||AD_C||Times]$$

- $K_{C,SP}$ ist der geheime Schlüssel für die Kommunikation zwischen Benutzer und **SP**.
- $Realm_{SP}$ ist der zugehörige Realm des gewünschten **SP**.
- K_{SP} ist der beim **TGS** hinterlegte geheime Schlüssel des **SP**.

3.2.4 Nutzen eines Service Dienstes

Hat der Benutzer ein Service Granting Ticket vom **TGS** für den gewünschten Service Dienst erhalten, kann er die Kommunikation mit dem **SP** beginnen. Dazu weist er sich mit nachfolgendem Paket beim **SP** aus:

C → SP:

$$Options||Ticket_{SP}||Authenticator_C$$

$Authenticator_C$ ist aufgebaut aus

$$Authenticator_C = E_{K_{C,SP}}[ID_C||Realm_C||TS_2||Subkey||Seqnr]$$

- TS_2 ist die aktuelle Zeit beim Benutzer. Dieser Wert dient dem **SP** zur Überprüfung, ob die Anfrage zeitnah eintrifft und erschwert Replay Attacken wie bei TS_1.

- $Subkey$ kann vom Benutzer verwendet werden, um einen neuen geheimen Sitzungsschlüssel zu definieren, für die gewünschte Kommunikation mit dem **SP**. Wird dieses Feld nicht verwendet, wird der vorgegebene Schlüssel $K_{C,SP}$ für die weitere Unterhaltung mit dem **SP** benutzt.

- $Seqnr$ ist ein weiteres optionales Feld, bei der der Benutzer eine Startsequenznummer vorgeben kann, welche der **SP** für seinen Antwortpaketen innerhalb der gewünschten Sitzung benutzen soll. Dieses Feature dient dem Benutzer zum Schutz vor Replay Attacken.

Hat der **SP** das Paket entschlüsselt und gewährt dem Benutzer die Nutzung seiner Dienste, so schickt er, falls der Benutzer eine gegenseitige Authentisierung fordert, das nachfolgende Paket zurück. Dabei wiederholt er die Zeitmarke TS_2, um zu belegen, dass er die Anfrage dekodieren konnte. Zudem kann auch der **SP** einen neuen $Subkey$ für die weitere Kommunikation angeben. Ausserdem hat er auch die Möglichkeit, für die weiteren Nachrichten des Benutzers eine Startsequenznummer vorzuschreiben.

SP $\rightarrow$ **C**:

$$E_{K_{C,SP}}[TS_2||Subkey||Seqnr]$$

3.2.5 Kerberosoptionen und Nachrichtenflags Erläuterungen

In diesem Abschnitt werden die unterschiedlichen Optionen mit ihren zugehörigen Flags vorgestellt.

- INITIAL: Das Ticket wurde anhand eines **AS** ausgegeben und nicht vom **TGS**. Version 5 von Kerberos bietet die Option, ein Service Granting Ticket für bestimmte Dienste direkt vom **AS** zu beziehen. Dies kann z.B. für einen Passwortänderungsdienst benutzt werden, wo man sicherstellen möchte, dass das Benutzerpasswort erst kürzlich verifiziert wurde.

- PRE-AUTHENT: Dieses Feld gibt an, ob der Benutzer vor der Ausstellung des Ticket Granting Tickets beim **AS** authentisiert wurde. Für diese Vorauthentisierung wird bei Kerberos kein genaues Verfahren definiert. Es kann aber z.B. benutzt werden, um mit Smartcards oder biometrischen Verfahren zu arbeiten.

- HW-AUTHENT: Wenn eine Smartcard oder sonstige zusätzliche Hardware für die PRE-AUTHENT benutzt wird, so wird die Art und Weise in diesem Feld klassifiziert.

- RENEWABLE: Durch dieses Flag kann das Problem eines sehr lang gültigen oder auch sehr kurz gültigen Ticket Granting Tickets entschärft werden. Ist das Ticket Granting Ticket nur sehr kurz gültig, so muss der Benutzer sich immer wieder neu am **AS** anmelden. Ist dagegen das Ticket sehr lange gültig, erhöht sich die Gefahr eines Diebstahls oder sonstigen Missbrauchs. Deswegen besitzt dieses Feld zwei Werte. Der erste Wert gibt die Zeitmarke an, wie lange das aktuelle Ticket gültig ist. Der zweite Zeitwert gilt für die gesamte erlaubte Sitzung mit dem **TGS**. Wird dieses Feld benutzt, kann der Benutzer den **TGS** bitten, ein neues Ticket Granting Ticket auszustellen, solange die zweite Zeitmarke nicht überschritten ist.

- MAY-POSTDATE: Der Benutzer kann beim **AS** ein Ticket Granting Ticket anfordern, bei dem dieses Feld gesetzt ist. Ist dieses Feld gesetzt, so hat der Benutzer die Möglichkeit, beim **TGS** gleich mehrere Service Granting Tickets anzuforden, welche aber bis auf eines POSTDATED oder INVALID sind und somit ungültig sind. Genutzt werden können solche Tickets bei sehr lang laufenden Batchjobs auf einem Diensteserver, welcher in bestimmten Abständen ein erneutes Ticket des Benutzers benötigt. Die Bedeutung des POSTDATED und INVALID Feldes wird unter den nächsten beiden Punkten beschrieben.

- POSTDATED: Der Benutzer braucht nicht ständig sein Ticket Granting Ticket zu benutzen, um neue Tickets zu erhalten, sondern schickt dem Diensteserver eines der schon zuvor erhaltenen Tickets. Der Diensteserver kann überprüfen, wann die ursprüngliche Austellung erfolgte.

- INVALID: Dieses Flag kennzeichnet, dass dieses Ticket ungültig ist und vom **AS** noch validiert werden muss.

- PROXIABLE: Bei Kerberos Version 5 können Server als Proxy für eine Clientanfrage dienen. Sie übernehmen dann die notwendigen Privilegien und Zeugnisse des Clients um statt seiner einen anderen Dienst Server zu kontaktieren. Wenn ein Client diesen Mechanismus wünscht, so beantragt er beim **AS** ein Ticket Granting Ticket mit diesem Flag gesetzt. Ein typischer Einsatz eines solchen Proxys ist, wenn der Client einen Druckservice nutzen will, und der Druckservice das Recht bekommen soll, auf den Fileserver zuzugreifen, um die benötigte Datei zu bekommen.

- PROXY: Stellt der **TGS** ein Service Granting Ticket aus, das der Benutzer für Proxyzwecke nutzen kann, so ist dies Service Granting Ticket mit diesem Flag markiert. Der **TGS** erlaubt dabei, dass die Netzwerkadresse der Service Anfragenden Instanz eine andere ist, als die des Benutzers. Der zuständige Diensteserver kann dann diese Anfrage akzeptieren oder zusätzliche Authentikationsschritte fordern.

- FORWARDABLE: Dieses Konzept erweitert das Proxykonzept auf die Interrealm Kommunikation. Ein Benutzer kann ein spezielles Ticket Granting Ticket beantragen, bei dem dieses Flag gesetzt ist. Der **TGS** kann dann dem anfragenden Benutzer ein Ticket Granting Ticket mit einer unterschiedlichen Netzwerkadresse ausstellen, welches der Benutzer einem **TGS** in einem anderen Realm präsentieren kann.

- FORWARDED: Stellt der **TGS** ein solches Ticket Granting Ticket aus, so wird dieses Flag gesetzt und der Benutzer kann das Ticket für einen **TGS** in einem anderen Realm verwenden. Somit kann der Benutzer Dienste in einem anderen Realm benutzen, ohne dass die Kerberossysteme sich geheime Schlüssel teilen müssen. Es lassen sich mit dieser Methode hierarchische Strukturen von Realms aufbauen, die der Benutzer durchlaufen kann, um Dienste in dem gewünschten Realm zu nutzen.

3.2.6 Single Logout bei Kerberos

Die Kerberos Spezifikation für Version 5 [KN93], geht nicht auf den Prozess des Single Logouts ein. Ich werde deswegen hier beschreiben, wie die Beendigung in der Regel bei vielen Implementierungen des Protokolls abläuft. Zu betonen ist aber, dass bei Kerberos für die Benachrichtigung einer Abmeldung des Benutzers bei den Kommunikationspartnern, keine Vorgaben existieren.

In der Regel hält der Rechner des Benutzers die notwendigen Daten für die sichere Kommunikation in einem Kerberosverbund nur im Arbeitsspeicher vor. Zusätzlich läuft auf dem Benutzercomputer ein Prozess (Kerberosdaemon), der für die Schlüsselverwaltung und die Kommunikationsschritte mit den **AS** und **TGS** zuständig ist. Die Diensteapplikationen haben dann eine Schicht implementiert, die die Interaktion mit diesem Kerberosdaemon ermöglicht. Bei der Beendigung einer Benutzer Session werden dann nur die sensiblen Daten beim Kerberosdaemon aus dem Arbeitsspeicher entfernt.

Keiner der verteilten Kommunikationspartner bekommt aber diese Beendigung der Benutzer Session mit. Bei den **AS**, **TGS** und **SP** läuft die Session erst ab, wenn diese Dienste auf das Timeout der vorhandenen Tickets treffen.

3.3 Microsoft Passport

Seit Windows2000 implementiert Microsoft ein eigenes Single Sign-On System für den firmeninternen Einsatz. Dieses Verfahren basiert auf Kerberos Version 5, erweitert das Modell aber für den Umgang mit asymmetrischen Kryptoverfahren. Dieses System ist integraler Bestandteil aller neueren Windows Betriebssystemversionen und ist Teil des *Active Directory Systems*.

Zusätzlich hat Microsoft in den letzten Jahren ein zweites System vorgestellt, welches speziell für das Internet und Webservices entwickelt worden ist. Dieses zusätzliche Single Sign-On System firmiert unter dem Namen *.NET Passport* und soll in dieser Arbeit genauer untersucht werden. Microsoft bezeichnet dieses System als *Internet authentication service* [Mic04a].

Das Passport Framework wird zur Zeit in einigen grundlegenden Konzepten von Microsoft überarbeitet, nachdem verschiedene Gruppen und Institutionen gravierende Mängel aufgezeigt haben. Unter anderem fordert die Europäische Union eine Überarbeitung des Konzeptes, da es nicht konform mit der Europäischen Datenschutzrichtlinie ist [Art02]. Die aktuelle Version von Microsoft .Net Passport ist 2.5.

3.3.1 Aufbau von Microsoft Passport

In seinen Dokumenten verwendet Microsoft andere Bezeichnungen für die unterschiedlichen Akteure. Zudem bezeichnet Microsoft sein Verfahren als „Single Sign-In“. Ich werde in dieser Arbeit jedoch wegen der besseren Vergleichbarkeit die zuvor definierten Begriffe verwenden.

- the client = der Benutzer
- the merchant = der **SP**
- the Passport Sign-On server = der **ASP**

Das Passport System dient nicht nur der Authentikation von Benutzern, sondern erweitert es um die Übertragung von Profilinformationen. Die Grundidee ist, dass der Benutzer an einer zentralen Stelle bei Microsoft im Internet Zugangsdaten hinterlegt und dadurch auf alle weiteren angeschlossenen Dienste der **SP**

ohne weitere Abfrage seiner Identität zugreifen kann. Microsoft übernimmt in dieser Infrastruktur die Rolle des **ASP**.

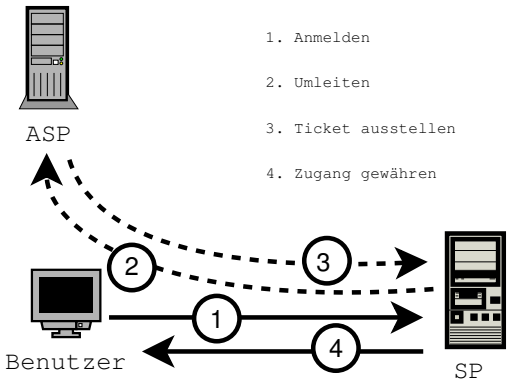

Abbildung 3.8: Passport: Zugang zum **SP**

3.3.2 Systemvoraussetzungen

Der **ASP** wird von Microsoft betrieben und wird als fertiger Dienst zur Verfügung gestellt. Ansonsten benötigt man für die Teilnahme an Passport folgende Komponenten [Mic04k]:

Servervoraussetzungen für den SP:

- Als Rechnerarchitektur kommen nur X86-Computer in Frage.
- Das eingesetzte Betriebssystem muss Microsoft Windows 2000 Server, Microsoft Windows XP Professional oder Microsoft Windows 2003 Server sein.
- Der Webserver muss ein Microsoft Internet Information Services (IIS) Version 5.0 oder höher sein.

Desweiteren setzt Microsoft auf standardisierte Webtechniken, wie HTTP, HTTPS, HTML-Formulare, Java Skript[10], Cookies[11]und XML. Bestimmte Seiten müs-

[10]Java Skript ist nicht zwingend notwendig, erleichtert aber die Kommunikation.

[11]Auch Cookies sollen nicht zwingend sein. Mir ist aber kein **SP** bekannt, der ohne Cookies funktioniert. Microsoft selbst empfiehlt die Benutzung von Cookies bei den **SP**s [Mic04o].

sen beim **SP** mit ASP[12] und einem Microsoft COM-Objekt, dem *Passport Manager*, erstellt werden.

Benutzervoraussetzungen: Es werden auf Benutzerseite nur ein Webbrowser mit der Unterstützung von SSL und Cookies benötigt. Zur Darstellung des Co-Branding und zur Perfomancesteigerung wird Java Script empfohlen. Es sollte dennoch darauf hingewiesen werden, dass Microsoft andere Browser als den IE nicht offiziell supportet.

3.3.3 Registrierungsprozess

Da Passport nicht für ein geschlossenes User System gedacht ist, besteht die Registrierung von neuen Benutzern aus einem stetigen dynamischen Prozess. Es steht jedem Benutzer frei, sich jederzeit einen Passportaccount einzurichten, der dann automatisiert freigeschaltet wird. Ein Benutzeraccount ist gebührenfrei bei Microsoft zu erhalten.

Die **SP** müssen sich auch bei Passport registrieren lassen und bestimmte Auflagen erfüllen.

Service Provider Registrierung

Ein **SP** der an der .NET Passport Infrastruktur teilnehmen möchte, muss gegenüber Microsoft zunächst Angaben zur Datensicherheit, zum Webangebot und zum Umgang mit Nutzerdaten machen [Mic04h]. Erfüllt der **SP** die Anforderungen, so erhält er von dem Passport System eine eindeutige Kennung (Site-ID[13]), sowie einen geheimen symmetrischen 3DES Schlüssel, den *Microsoft .NET Passport participant key* für die Kommunikation mit dem **ASP**. Als teilnehmender **SP** muss man Gebühren an Microsoft bezahlen. Beim Registrieren eines **SP** am Passport System wird vom **SP** seine Hauptdomäne angegeben. Alle Server innerhalb dieses Domänenbereichs können an dem Passport Verfahren teilnehmen

[12]Das Akronym ASP hat hier eine andere Bedeutung! **Active Server Pages** ist Microsofts Alternative zu dynamisch erstellten HTML-Seiten, vergleichbar zu *Java Server Pages* oder *PHP*.

[13]Für die Generierung werden hardwarespezifische Daten wie MAC-Adresse, Festplattendaten oder CPU-IDs verwendet, um Eindeutigkeit zu gewährleisten.[May04]

[Mic04n].

.NET Passport bietet dem **SP** 3 Ausprägungen für die Kommunikation an:

- Standard Single Sign-On: Hier wird nur bei der Übertragung der Anmeldedaten verlangt, dass diese über einen verschlüsselten Kanal (SSL) erfolgen. Die restliche Kommunikation kann über HTTP erfolgen. Diese Variante ist das Standardverfahren.
- Channel Single Sign-On: Sämtliche Kommunikation mit dem Passport System und den **SP** muss über SSL erfolgen.
- Strong Credential Single Sign-On: Zusätzlich zur Channel Single Sign-On Authentikation wird der Benutzer beim **SP** zur Eingabe einer 4-stelligen PIN gebeten. Erst wenn diese zusätzliche Kennung angegeben wird, gilt der Benutzer als authentisiert.

Benutzerregistrierung

Der Benutzer hat zwei Möglichkeiten, einen Account bei Passport einzurichten. Er muss dies entweder direkt über Microsoft oder über einen der partizipierenden Serviceprovider tun.

Direkte Benutzer Registrierung über Microsoft: Ein Benutzer kann sich einen Passportaccount direkt unter `http://www.passport.com` einrichten. Alternativ wird bei jeder Registrierung für einen Emailaccount bei Hotmail[14] oder bei der Registrierung bei einem anderen Dienst von MSN[15] automatisch ein Passportaccount eingerichtet. Zudem kann der Benutzer unter der Voraussetzung, dass er das Betriebssystem Microsoft WindowsXP verwendet, den „Passport Registration Wizard“ benutzen und wird sogar bei der Registrierung von WindowsXP dazu aufgefordert.

Die Passport Benutzer Account Daten teilen sich in zwei Kategorien auf [Mic04g]:

[14] `http://www.hotmail.com` ist ein kostenloser Email Service von Microsoft.

[15] Microsoft Network Services ist sozusagen das Internetportal mit den von Microsoft betriebenen Servicediensten, u.a. Messenging, Internetzugang, Suchmaschine, Foren und Nachrichten.

Credentials: Hierunter fallen alle Account Informationen, die nur von dem Passport System benutzt werden.

Profile data: Umfasst die Benutzerdaten, welche mit den SPs geteilt werden.

Die unterschiedlichen Benutzeraccountdaten werden in keinem Microsoft Dokument vollständig dokumentiert. Den Umstand einer undurchsichtigen Dokumentation hat auch die Datenschutzgruppe der Europäischen Union bemängelt:

> „Bei der Untersuchung der Arbeitsweise von .NET Passport stieß die Datenschutzgruppe zuallererst auf das Problem, klare und transparente Informationen über dieses System zu finden. Das verfügbare Informationsmaterial war zum Teil unpräzise und konnte keine Aufklärung zu den wesentlichen Datenschutzfragen bieten (Identität des für die Verarbeitung Verantwortlichen, Zweck der Verarbeitung, Rechte der Betroffenen, Empfänger der Daten, Aspekte, die zwecks fairer Verarbeitung unbedingt geklärt werden müssen); manchmal erhielt das Material sogar widersprüchliche Angaben.“ [Art02]

Basierend auf [Mic04g] und [Mic04d] sowie [Mic04h] wird nun versucht, eine vollständige Liste der möglichen zu erfassenden Daten zu erstellen.

Liste von Benutzerdaten bei .NET Passport

Profil Datum	Benötigt für ein Passportkonto	Sichtbar für die **SP**s	Je **SP** veränderbar	Kategorie **C**ore/ **P**rofile
Emailadresse[1]	ja	ja[2]	nein	C/P
Alternative Email[3]	nein	nein	nein	C
Passwort	ja	nein	nein	C
Mobilnummer	ja[4]	nein	nein	C
PIN	ja[4]	nein	nein	C
Frage und Antwort[5]	nein	nein	nein	C
Vorname	nein	opt-in[6]	nein	P
Nachname	nein	opt-in[6]	nein	P
Geburtstag	nein	opt-in[6]	nein	P
Geschlecht	nein	opt-in[6]	nein	P
Stadt	nein	opt-in[6]	nein	P
Postleitzahl	nein	opt-in[6]	nein	P
Land	nein	opt-in[6]	nein	P
Sprache	nein	opt-in[6]	nein	P
Zeitzone	nein	opt-in[6]	nein	P
Berufstätigkeit	nein	opt-in[6]	nein	P
PUID	ja	ja	nein	P
GEOID	nein	opt-in[6]	nein	P
Security Key[7]	nein	ja	nein	P

[1]Microsoft Passport verlangt eine gültige Emailadresse. Erst durch das Zurücksenden einer speziellen Email des Benutzers an das Passport System wird der Account aktiviert.

[2]Es gibt **SP** innerhalb des Passport Systems, denen die Email Adresse übermittelt wird, unabhängig von den Benutzereinstellungen [Mic04h].

[3]Eine Alternative Emailadresse dient allein dem Zweck bei einem Verlust des Anmeldepasswortes, eine abweichende Email angegeben zu haben, so dass eine Kommunikation mit dem Passport System weiterhin möglich ist. Hat der Benutzer z.B. eine Emailadresse bei Hotmail.com eingetragen, so hat er bei Verlust des Passwortes keine Möglichkeit, sein Passwort zurücksetzen zu lassen, da er nicht mehr an sein Emailkonto gelangen kann.

[4]*Mobilnummer* und *PIN* werden benötigt, wenn .NET Passport über Mobilfunk Netzwerke benutzt werden soll.

[5]Falls der Benutzer sein Passwort vergisst, kann er mit Hilfe dieser Daten ein neues Passwort beantragen.

[6]Wenn der Benutzer sein Einverständnis gibt. Standardmässig hat der Benutzer die Wahl *Emailadresse* und/oder *Vor- und Zuname* und/oder die restlichen Profildaten einem **SP** zu übertragen.

[7]Dieser vierstellige Wert wird für das *Strong Credential Single Sign-On* benutzt.

Tabelle 3.1: Passport Benutzerdaten

Der Benutzer hat nach Eingabe seiner Daten die Möglichkeit, bestimmte Einschränkungen bei den zu übertragenden Profildaten zum **SP** einzustellen. Er kann erlauben, dass nur die Emailadresse und/oder Vor- und Zuname und/oder alle anderen Profildaten gesendet werden sollen [Mic04b].

Benutzerregistrierung über einen Serviceprovider:

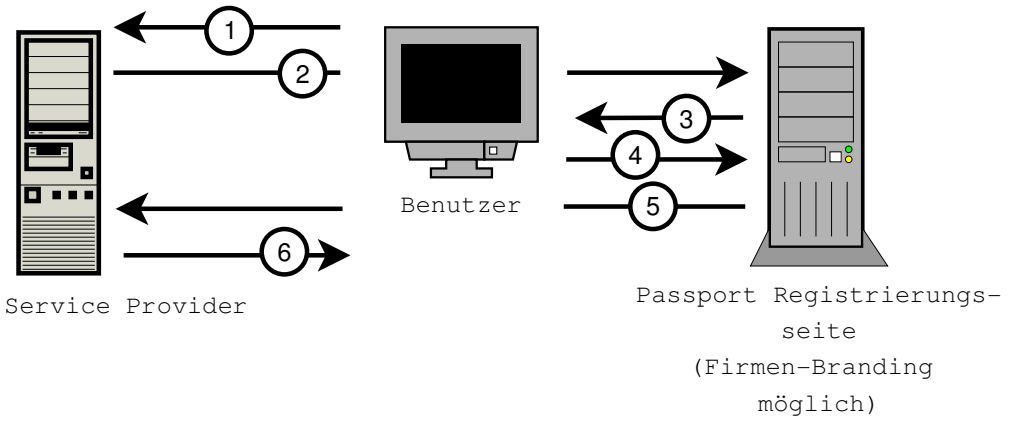

Abbildung 3.9: Passport: Benutzerregistrierung über den **SP**

Der **SP** bindet ein vorgegebenes Element in seiner Seite ein, welches kennzeichnet, dass er bei der .NET Passport Infrastruktur eingebunden ist. Klickt der Benutzer bei einem teilnehmenden **SP** auf den *Sign-In Button* 

(1), so wird er weitergeleitet[16] zu dem Passportserver (**ASP**) von Microsoft (2). Dieser bietet jetzt eine, unter Umständen nach dem Aussehen der Firma angepasste, Formularseite an, die auch nur eine Untermenge der Profilfelder beinhalten kann (3). Der Benutzer gibt die gewünschten Informationen ein und entscheidet sich, ob andere **SP**s Zugriff auf diese Daten bekommen sollen (4). Anschliessend wird der Benutzer wieder zurück an den **SP** geleitet (5). Die Profildaten, die der Benutzer angegeben hat, werden explizit freigeschaltet für die Übermittlung an den **SP**. Jetzt überträgt der **SP** dem Benutzer seine eigene Webseite (6).

[16] redirected

3.3.4 Benutzer authentisieren

Ein Einloggen in das Passportsystem kann direkt bei Microsoft[17] oder über eine Webseite beim **SP** beginnen. Da die direkte Authentikation bei Microsoft ein Unterprozess der allgemeinen Form beim **SP** ist, werde ich nur den Vorgang über einen **SP** beschreiben.

Authentikation über einen Serviceprovider: Die Anforderung einer reglementierten Webseite bei einem **SP** mit anschließender Authentikation und abschließender Auslieferung des gewünschten Webcontents lässt sich in mehrere Schritte unterteilen.

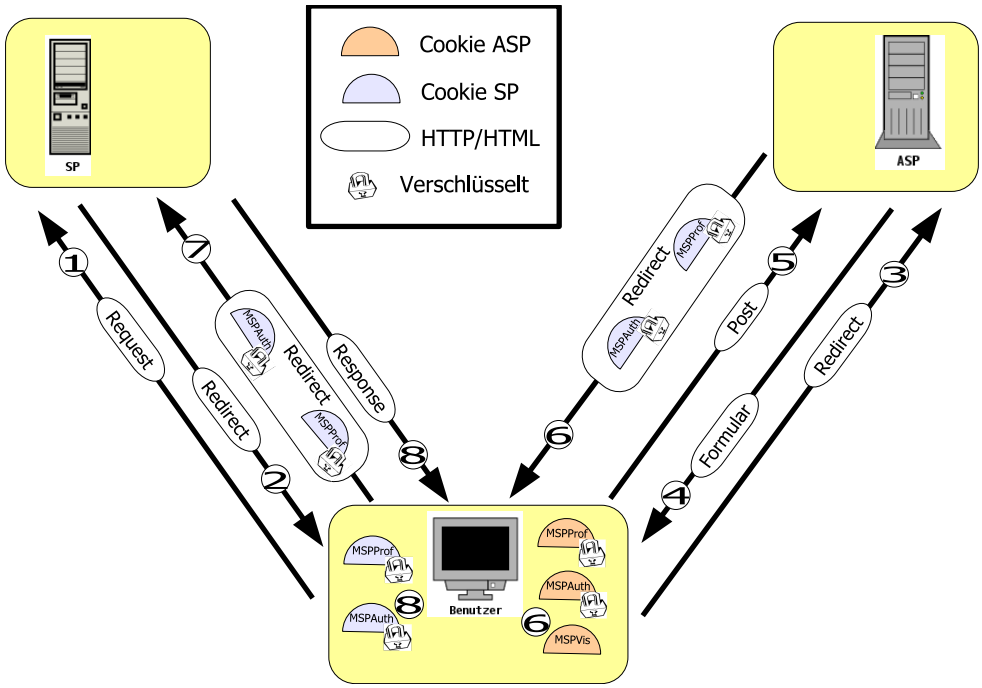

Abbildung 3.10: Passport Single Sign-On

[17] Unter `http://www.passport.com` oder `http://www.passport.net`.

1. Der **SP** benutzt für seine Anmeldungsseite das *ASP*[18]-Modul *Passport Manager*. Dieses Modul stellt die notwendigen Objekte und Methoden für die Anmeldung an die Passport Infrastruktur bereit [Mic04e]. Der **SP** hat die Alternative zwischen dem Einbinden eines Bildes über die Methode *LogoTag* oder einer URL mit *AuthURL*. Klickt ein Benutzer auf dieses *Sign-In* Objekt, wird zunächst beim **SP** ein Verwaltungsobjekt für den neuen Benutzer innerhalb des *Passport Manager* erstellt. Anschließend wird mit der Methode *IsAuthenticated* überprüft, ob der Benutzer schon angemeldet ist und ein gültiges Serviceticket besitzt. Zu diesem Zweck werden die lesbaren Cookies oder der Querystring durchsucht [Mic04m].

2. Ist der Benutzer noch nicht angemeldet, so wird der Benutzer über einen Aufruf von *LoginUser* zu einer Webseite auf dem **ASP** weitergeleitet. Dazu schickt der **SP** die HTTP-Meldung 302 *move* an den Benutzer. Der *Passport Manager* beim **SP** kann eine feste Ziel-URL benutzen oder den Nexus[19] kontaktieren um die **ASP**-URL zu erfahren. In der neuen *Move*-URL sind die *Site-ID* und die Rücksprung-URL des **SP** als Argumente enthalten.

3. Der Browser des Benutzers folgt dem Redirect und sendet eine *GET* Anfrage an den **ASP**.

4. Der **ASP** checkt die *Site-ID* und generiert eine Anmeldeseite für den Benutzer. Diese Anmeldeseite beinhaltet zwei Formularfelder EMAIL und PASSWORT, und der Benutzer wird aufgefordert, sich durch die Eingabe der Daten zu authentisieren.

[18]Das Akronym ASP hat hier eine andere Bedeutung! **Active Server Pages** ist Microsofts Alternative zu dynamisch erstellten HTML-Seiten, vergleichbar zu *Java Server Pages* oder *PHP*.

[19]Die Passport Server bilden eine Serverfarm, die aus Webservern, Datenbankservern und Anmeldeservern bestehen, die Profildaten speichern, Benutzer und Partnersites authentifizieren und Daten aktualisieren. Jeder dieser Dienste kontaktiert regelmässig einen zentralen Dienst namens Nexus, welcher für die Koordination sowie Synchronisation der Server untereinander zuständig ist. Zusätzlich gleicht der *Passport Manager* beim **SP** seine Einstellungen mit dem Nexus ab. Somit lässt sich ein redundantes System anbieten, das für Fehlertoleranz, Skalierbarkeit, Sicherheit, Leistung und eine maximale Betriebszeit erstellt wurde [Mic04c].

5. Der Benutzer füllt die Formularfelder aus und klickt auf den *Sign-In* Button dieser Webseite. Somit wird eine HTTPS-Verbindung zwischen Benutzer und **ASP** aufgebaut und die Daten werden anhand des HTTP *POST*-Verfahrens übertragen. Nur dieser Kommunikationsschritt zwischen Benutzer und **ASP** wird in jeden Fall über einen verschlüsselte SSL-Verbindung vollzogen. Alle anderen Kommunikationsschritte können verschlüsselt erfolgen, müssen aber nicht.

6. Der **ASP** überprüft die empfangenen Daten und ermittelt bei Korrektheit die *PUID* des Benutzers. Anhand der *PUID* kann auf den zugehörigen Datensatz des Benutzers referenziert werden. Der **ASP** sendet nun an den Browser des Benutzers die zuvor vom **SP** übergebene Rücksprung-URL. Zusätzlich setzt der **ASP** ein Reihe von Cookies, die in Kapitel 3.3.5 genauer aufgelistet werden. Die gesendete URL wird um zusätzliche Parameter ergänzt, die verschlüsselte Daten zum Setzen von Cookies zur Verwendung beim **SP**[20] enthalten.

7. Der Browser des Benutzers führt den Redirect aus. Dieser setzt aber zuvor die übermittelten Cookies. Diese Cookies werden unter der Domäne `passport.com` abgelegt und sind nur vom Passport System abrufbar. Es wird somit u. a. dem Benutzer ein Ticket Granting Ticket für das Passport System ausgestellt.

8. Der **SP** extrahiert die Daten aus der URL mit seinem *Participant Key*. Anschließend erkennt er, ob der Benutzer authentisiert wurde und erhält dann die *PUID* des Benutzers. Ausserdem können weitere von Passport übertragene Profildaten entschlüsselt werden. Der **SP** hat nun die Möglichkeit, anhand der eindeutigen *PUID* den Benutzer innerhalb seiner Datensammlung zu identifizieren und kann somit auch auf zusätzliche eigene Profildaten zurückgreifen. In der Regel wird der **SP** nun die erhaltenen Daten, sowie auch eigene Authentisierungs- und Profildaten beim Benutzerbrowser in Cookies ablegen. Abschließend sendet er den HTTP-Code 200 *OK* und eine Webseite aus seinem Angebot (z.B. eine Willkommensseite).

[20] Der **ASP** kann keine Cookies setzen, welche der **SP** auslesen kann, da diese Server in unterschiedlichen Domänen betrieben werden.

Ist ein Benutzer schon im Passport System angemeldet, besucht aber einen **SP**, bei dem er noch nicht eingeloggt ist, so benötigt er vom **ASP** sozusagen ein Service Granting Ticket. Der Ablauf dieses Prozesses ist fast identisch mit dem eben beschriebenen. Einzig die Schritte 4. und 5. laufen jetzt ohne Benutzerinteraktion ab. Der **ASP** überprüft die *Site-ID*, liest das Ticket Granting Cookie beim Benutzerbrowser aus und erfährt somit die notwendigen Daten, um mit Schritt 6. fortzufahren.

3.3.5 Vom Passport System verwendete Cookies

Beim Einloggen des Benutzers in das Passport System legt der **ASP** bis zu fünf Cookies beim Webbrowser des Benutzers ab [Mic04i], [Mic04g]. Diese Cookies werden unter der Domäne `passport.com` abgelegt und sind nur für den **ASP** lesbar. Kein **SP** hat Zugriff auf diese Daten. Zusätzlich sind diese Cookies verschlüsselt, wobei Microsoft keine Angaben zu dem benutzten Verschlüsselungsverfahren macht. Es ist aber zu vermuten, dass 3DES benutzt wird, da es ansonsten überall sonst verwendet wird.

Domain Authority Cookies:

- MSPProf (Profile)[21]: Enthält die Profilattribute des Benutzers.
- MSPAuth (Ticket): Enthält Zeitmarken (letzte Aktualisierung sowie der letzte manuelle Anmeldevorgang beim **ASP**), ob das Benutzerpasswort abgespeichert werden soll, „key version verification“ und zusätzliche Flags, die vom **ASP** gesetzt werden.
- MSPSecAuth (Secure): Wird bei den *Channel Single Sign-On* und *Strong Credential Single Sign-On* verwendet. Beinhaltet ein Token, das dem *Passport Manager* erlaubt, zu überprüfen ob eine SSL-Verbindung genutzt wird.

[21] In den Klammern steht der von Microsoft verwendete Common Name.

- MSPSec (Ticket Granting): Wird bei Browsern benutzt, die das Setzen von Cookies über HTTPS unterstützen. Beinhaltet die PUID und das Benutzerpasswort. Die Daten werden für einen *Silent Sign-In*[22] verwendet.

- MSPVis (Visited Sites): Enhält eine Liste der *Site-IDs* der **SP**s, bei denen der Benutzer aktuell eingeloggt ist. Diese Information wird beim Abmelden des Benutzers aus dem Passport System benutzt. Dieser Cookie wird nicht verschlüsselt abgelegt.

Der *Passport Manager* beim **SP** legt auch Cookies beim Browser des Benutzers ab. Den Dateninhalt erhält er über einen Querystring vom **ASP**. Diese Cookies werden in der Domäne des **SP** abgelegt. In der Regel werden sie in der Rootdomäne des **SP** abgelegt. Der **SP** kann aber auch vorgeben, dass diese in Subdomänen abgelegt werden. Alle vier Cookies werden mit dem *Passport participant key* des **SP** verschlüsselt.

Participating Site Cookies:

- MSPConsent (Consent): Dieser Cookie kann benutzt werden, wenn sich das Angebot des **SP** in seinen Subdomänen unterscheidet. Dann hat der **SP** jeweils die Möglichkeit, pro Subdomäne dieses Cookie zu setzen und kann den Benutzer differenzierter verfolgen.

- MSPProf (Profile): Hier sind die für den **SP** freigegebenen Profildaten abgelegt.

- MSPSecAuth (Secure): Wird bei den *Channel Single Sign-On* und *Strong Credential Single Sign-On* verwendet. Beinhaltet ein Token, das dem *Passport Manager* erlaubt zu überprüfen, ob eine SSL-Verbindung genutzt wird.

- MSPAuth (Ticket): Enthält Zeitmarken (letzte Aktualisierung sowie der letzte manuelle Anmeldevorgang beim **ASP**), ein Flag, ob das Benutzerpasswort abgespeichert werden soll, „key version verification“ und zusätzliche Flags, die vom **ASP** gesetzt werden.

[22]Beim Einloggen in das Passport System hat der Benutzer die Möglichkeit anzugeben, dass sein Loginname und Passwort permanent in ein Cookie geschrieben werden. Dadurch braucht sich der Benutzer auch nach einem Beenden des Browsers nicht erst manuell anzumelden. MSPSec wird erst beim expliziten Ausloggen vom Passport System gelöscht.

Die meisten Cookies bei Passport sind temporäre Cookies, d.h. sie werden gelöscht, wenn der Browser geschlossen wird. Wählt der Benutzer beim Anmelden „sign me in automatically“, so ändert sich dieser Sachverhalt. Dann werden mehrere Cookies beim Beenden des Browsers auf die Festplatte beim Benutzer geschrieben.

3.3.6 Single Logout bei Passport

Der *Passport Manager* beim **SP** benutzt Cookies innerhalb seiner Domäne, um den Authentikationsstatus des Benutzers zu verfolgen. Jeder **SP** ist angewiesen, beim Ausloggen des Benutzers die Passport sowie seine eigenen Cookies zu löschen[23]. Beim Abmelden des Benutzers kann der **ASP** diese Cookies nicht löschen. Deswegen muss jeder **SP** ein Skript beim **ASP** anmelden, das sich um die Löschung dieser Cookies kümmert [Mic04f].

Klickt ein Benutzer bei irgendeinem **SP** auf den *Sign Out* Button 

, so wird er zu dem **ASP** weitergeleitet. Der **ASP** kann anhand des *MSPVis*-Cookies feststellen, bei welchen **SP**s der Benutzer zur Zeit eingeloggt ist. Zunächst löscht der **ASP** alle Cookies in der `passport.com` Domäne. Abschliessend generiert der **ASP** eine Webseite für den Benutzer, auf der der Benutzer erkennen kann, ob der Abmeldevorgang bei den **SP**s erfolgreich war. Dazu bettet er in diese Webseite die Aufrufe für die Löschskripte der **SP**s ein[24].

[23] Dies geschieht durch die Zuweisung des Wertes „“ mit einer abgelaufenen Gültigskeitsdauer.

[24] Der **ASP** ruft eine Bild-URL beim **SP** auf. Dort ist aber zunächst kein Bild, sondern das Skript hinterlegt. Ist das Skript ausgeführt, wird anhand der Rückgabe eines Bildes bestimmt, ob der Vorgang erfolgreich war oder nicht.

3.4 Liberty Alliance

Die Liberty Alliance (LA) wurde im Jahre 2001 gegründet. Ziel dieses Konsortiums ist es, offene Standards zu entwickeln, für eine föderale Netzwerk-Identität-Managment-Infrastruktur mit Single Sign-On.

> „The members of the Liberty Alliance envision a networked world across which individuals and businesses can engage in virtually any transaction without compromising the privacy and security of vital identity information." [CHKT03]

Aus den 20 Gründungsmitgliedern sind inzwischen über 150 geworden. Darunter viele Firmen aus dem IT-, Mobilfunk- und Banken-Sektor, sowie Bildungseinrichtungen und staatliche Verbände [Lib04a]. Der Standardisierungprozess bei der Liberty Alliance ist noch nicht abgeschlossen. Dennoch sind schon für bestimmte Bereiche des Frameworks Standards veröffentlicht worden.

3.4.1 Aufbau von Liberty Alliance

Ein Benutzer hat z.Zt. schon viele verteilte Anmeldungszeugnisse und Profildaten im Internet. Ziel des LA-Framework ist die Verknüpfung der verstreuten Informationen (Identity federation), statt eines erneuten Aufbaus dieser Daten an einem zentralen Speicherort. Deswegen muss die Definition des **ASP** hier erweitert werden zu einer Instanz, welche auch eine Untermenge der Profildaten eines Benutzers verwaltet und den Zugriff darauf autorisiert. Die Spezifikation lässt sehr viele Ausprägungen von Liberty Alliance Implementationen zu, versucht aber trotzdem präzise zu sein[25]. Dieses gelingt aber nicht immer, da versucht wird sehr unterschiedliche Anwendungsszenarien abzubilden. Abhilfe soll das Konzept von `Liberty Profiles` bringen. Dennoch muss man in vielen unterschiedlichen Dokumentationen nachschauen, um sich ein genauen Überblick einzelner Profile zu verschaffen und es ist nicht immer eine klare Abgrenzung zu erkennen. Dies kann leicht zu Widersprüchen bei Implementierungen führen.

[25] Z.B. wird das RFC-2119 [Bra97] benutzt, welches die sprachlichen Formulierungen präzise definiert.

„The combination of message content specification and message transport mechanisms for a single client type (that is, user agent) is termed a Liberty profile. “ [AKW03, S.11]

Teilnehmer bei Liberty Alliance:

- Participant/User = der Benutzer
- Service Provider = der **SP**
- Identity Provider = der **ASP**

Es sei noch darauf hingewiesen, dass es keine klare Objekttrennung bei Liberty Alliance zwischen dem **SP** und dem **ASP** geben muss, da sowohl die Anmeldedaten als auch die Profildaten des Benutzers verteilt im System vorkommen können. Somit kann ein **SP** zugleich ein **ASP** sein oder auch umgekehrt.

Eines der zentralen Konzepte bei Liberty Alliance ist das Bilden von Circle-of-Trust.

Circle-of-Trust: „A federation of service providers and identity providers that have business relationships based on Liberty architecture and operational agreements and with whom users can transact business in a secure and apparently seamless environment.“ [Lib03d]

Das bedeutet, dass innerhalb eines Circle-of-Trust ein Benutzer seine verteilten Profildaten miteinander verknüpfen kann und sehr granular Zugriffsrechte auf diese definieren kann. Außerdem hat der Benutzer die Möglichkeit, Single Sign-On innerhalb dieses Circle-of-Trust zu betreiben.

Als nächstes wird ein Überlick und eine Einordnung der verschiedenen Teile der Liberty Alliance Spezifikation gegeben und in unterschiedliche Module eingeordnet.

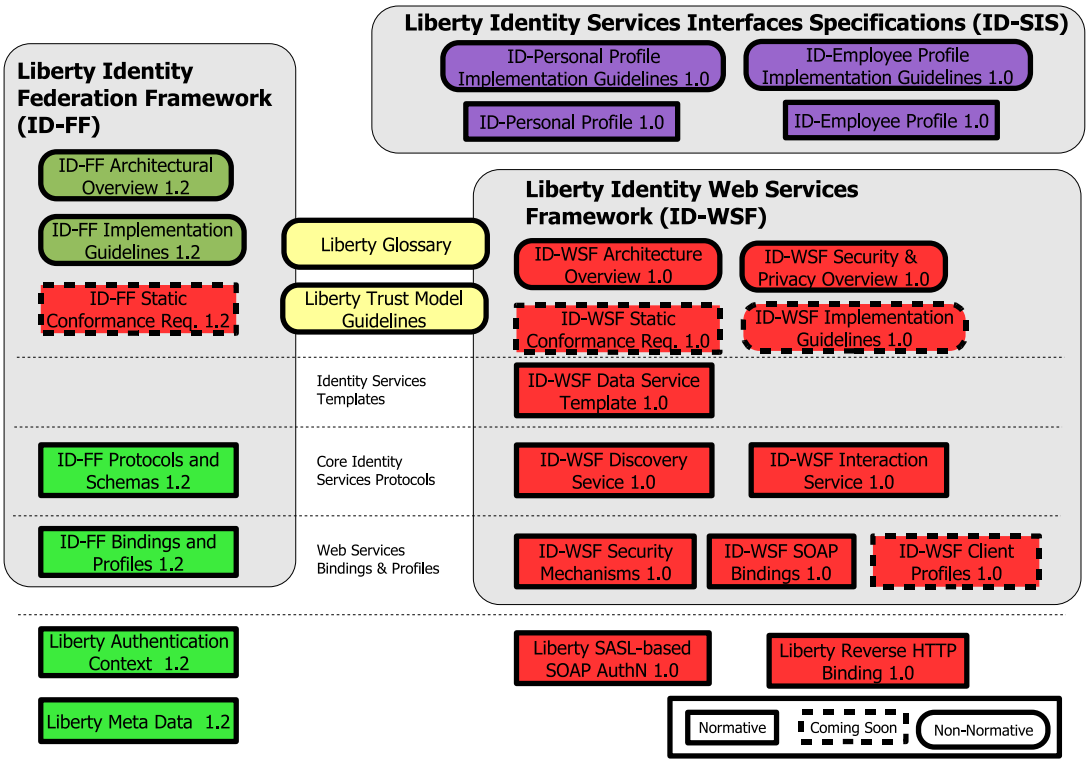

Abbildung 3.11: LA - Spezifikationen Übersicht

Modul 1 - Liberty Identity Federation Framework (ID-FF): Dieser Abschnitt behandelt die föderale Identität und deren Verwaltung. Diese Spezifikationssammlung deckt folgende Bereiche ab:

- Opt-in Account Linking: Erlaubt dem Benutzer, seine unterschiedlichen Profile bei Liberty-fähigen Teilnehmern zu verknüpfen.
- Simplified Sign-On: Beschreibt die Möglichkeit, dass der Benutzer sich einmalig bei einem Liberty Teilnehmer anmeldet und anschließend weitere Liberty Provider nutzen kann, ohne sich erneut zu authentisieren.
- Fundamental Session Management: Bietet den verknüpften **SP**s die Möglichkeit vorzuschreiben, wie der Benutzer beim Single Sign-On authentisiert werden soll. Außerdem wird der Vorgang des Single Logout behandelt.

- Anonymity: Bietet einem **SP** die Möglichkeit spezielle Profildaten abzufragen, ohne die Identität des Benutzers zu erfahren. So können z.B. Wetterdaten anhand der Postleitzahl dem anonymen Benutzer angezeigt werden.

- Protocol for the Real-time Discovery and Exchange of Meta Data: Damit Liberty Teilnehmer Informationen austauschen können, müssen zuvor Metadaten ausgetauscht werden (Zertifikate und Service-Endpunkte).

Modul 2 - Verwendete Basisprotokolle und deren Erweiterungen: Die Liberty Alliance versucht soweit wie möglich, existierende Standards in ihrem Framework zu integrieren. Dabei sind besonders Arbeiten von der „Organization for the Advancement of Structured Information Standards“ (OASIS), dem „World Wide Web Consortium“ (W3C) und der „Internet Engineering Task Force“ (IETF) eingeflossen. Die zugrundeliegenden Techniken beinhalten SAML, WS-Security, HTTP, WSDL, XML, SOAP, XML-ENC, XML-SIG, SSL/TLS und WAP.

Modul 3 - Liberty Identity Web Services Framework (ID-WSF): Das ID-WSF ist das fundamentale Gerüst für den föderalen Austausch von Profildaten und die Nutzung von Webservices innerhalb eines Liberty Alliance Verbundes . Es beschreibt Verfahren, um Identitätsmanagment-Dienste aufzubauen, im Netz aufzufinden und zu benutzen.

- Permission Based Attribute Sharing: Stellt Verfahren zur Verfügung, damit ein **SP** und Identity-Provider vom Benutzer freigegebene Informationen austauschen können. Beinhaltet auch Protokolle für die Erlaubnisabfrage beim Benutzer, seine Profildaten auszutauschen und die Austauschautorisierungen festzulegen.

- Identity Service Discovery: Dieser Dienst ermöglicht es einem **SP** die verteilt gespeicherten Profildaten zu lokalisieren. Somit können Profilinformationen über mehrere **ASP** verteilt sein.

- Interaction Service: Protokolle und Profile, falls ein Benutzer um eine Erlaubnis gefragt werden muss, bestimmte Profilinformationen dynamisch mit anderen Instanzen auszutauschen.

- Security Profiles: Legt Profile und Voraussetzungen für die sichere Lokalisierung und Benutzung von Identity Services fest. Behandelt sowohl den sicheren und vertrauenswürdigen Austausch der Daten zwischen den Teilnehmern als auch Datenschutzaspekte des Benutzers.
- Simple Object Access Protocol (SOAP) Binding: Stellt die Möglichkeit von SOAP-Bindings für die Benutzung von Liberty Alliance vor.
- Extended Client Support: Bietet Möglichkeiten Clients bei Liberty Alliance zu benutzen, welche über keinen oder einen nur sehr eingeschränkten TCP/IP- oder HTTP-Support verfügen (Mobilfunk Netze).
- Identity Services Templates: Informationen, um Identitätsdienste, basierend auf ID-WSF, aufzubauen. Behandelt die Abfrage und Modifikation von gespeicherten Daten.

Modul 4 - Liberty Identity Services Interfaces Specifications (ID-SIS): Dieses Modul baut auf ID-WSF auf und erweitert es um konkrete spezielle Service-Dienste, wie z.B. Registrierungen-, Kalender-, Adressbuch- oder Benachrichtigungsdienste. Somit soll gewährleistet werden, dass unterschiedliche Produkthersteller interoperable Dienste anbieten. *Personal Profile Service Specification (IDSIS-PP)* und *Employee Profile Service Specification (IDSIS-EP)* sind die ersten realisierten Spezifikationen daraus und sind Templates, um persönliche oder dienstliche Profildaten zu verwalten. Weitere Schemata sollen mit der Zeit folgen.

3.4.2 Systemvoraussetzungen

Die Liberty Alliance Spezifikation besteht, wie eben aufgezeigt, aus unterschiedlichen Unterspezifikationen. Zudem benutzt Liberty Alliance, wenn möglich, schon existierende etablierte Spezifikationen für den Unterbau und erweitert diese gegebenenfalls. Teilnehmer eines Liberty Netzwerkes müssen nicht alle Spezifikationen implementieren, sondern können je nach Funktionalität und Einsatzgebiet auch nur Teile derselben erfüllen, wie folgende Tabelle aufzeigt [Lib03c, S.5].

Funktionalität	**ASP**	**SP Basic**	**SP**	**LECP**
Single Sign-On mit Artifact Profile	MUSS	MUSS	MUSS	
Single Sign-On mit Browser POST Profile	MUSS	MUSS	MUSS	
Single Sign-On mit LECP Profile	MUSS	MUSS	MUSS	MUSS
Register Name Identifier (**ASP** initiiert) HTTP-Redirect	OPTIONAL	MUSS	MUSS	
Register Name Identifier (**ASP** initiiert) SOAP/HTTP	OPTIONAL	OPTIONAL	MUSS	
Register Name Identifier (**SP** initiiert) HTTP-Redirect	MUSS	MUSS	MUSS	
Register Name Identifier (**SP** initiiert) SOAP/HTTP	MUSS	OPTIONAL	MUSS	
Federation Termination Notification (**ASP** initiiert) HTTP-Redirect	MUSS	MUSS	MUSS	
Federation Termination Notification (**ASP** initiiert) SOAP/HTTP	MUSS	OPTIONAL	MUSS	
Federation Termination Notification (**SP** initiiert) HTTP-Redirect	MUSS	MUSS	MUSS	
Federation Termination Notification (**SP** initiiert) SOAP/HTTP	MUSS	OPTIONAL	MUSS	
Single Logout (**ASP** initiiert) HTTP-Redirect	MUSS	MUSS	MUSS	
Single Logout (**ASP** initiiert) HTTP-GET	MUSS	MUSS	MUSS	
Single Logout (**ASP** initiiert) SOAP	MUSS	OPTIONAL	MUSS	
Single Logout (**SP** initiiert) HTTP-Redirect	MUSS	MUSS	MUSS	
Single Logout (**SP** initiiert) SOAP	MUSS	OPTIONAL	MUSS	
Identity Provider Introduction (mittels cookies)	MUSS	OPTIONAL	OPTIONAL	

Tabelle 3.2: Liberty Alliance Teilnehmer und zu erfüllende Funktionalität

Jeder Teilnehmer benötigt eine eindeutige URI [BLuLM98] für die Teilnahme bei Liberty Alliance. Zudem darf die daraus resultierende identifizierende URI des Teilnehmers nicht 1024 Zeichen übersteigen [ABCS03, S.9] Zudem wird teilweise ein 30-byte langer Identifikator gefordert. Dieser wird mittels eines SHA-1 Hashwertes aus seiner URI erzeugt.

Die Benutzerkennung muss rein zufällig gewählt werden und nicht mehr als 256 Zeichen lang sein [ABCS03, S.9].

Bei der Signaturprüfung fordert LA, dass wenn möglich auch die Einbeziehung einer Vertrauenskette genutzt wird. Dies ist z.B. bei einer X.509 Zertifizierungsstruktur gegeben [ABCS03, S.9].

Liberty Alliance verlangt, dass ein teilnehmender Client zumindest HTTP[26], SSL 3.0 oder TLS 1.0 (dieses kann auch über einen Proxy realisiert sein) beherrscht. Zudem muss er URL-Längen bis 256 Bytes verarbeiten können. Bei WAP-Clients wird zusätzlich noch WML[27] vorausgesetzt.

Liberty Alliance schreibt nicht vor, in welchem Vertragsverhätniss die **ASP**s und **SP**s zueinander stehen müssen und wie mit den erhobenen Daten umgegangen werden muss. Es empfiehlt aber eine möglichst große Transparenz dieser Informationen dem Benutzer gegenüber aufzubringen.

> „Identity federation must be predicated upon prior agreement between the identity and service providers. It should be additionally predicated upon providing notice to the user, obtaining the user's consent, and recording both the notice and consent in an auditable fashion. Providing an auditable record of notice and consent will enable both users and providers to confirm that notice and consent were provided and to document that the consent is bound to a particular interaction.“ [CHKT03]

[26] Version 1.0 oder 1.1
[27] Version 1.0 bis 1.3

3.4.3 Registrierungsprozess

Serverregistrierung

Die genauen Formate für die Kommunikation zwischen den **SP** und **ASP** wird unter [Lib04b] erläutert. Außerdem kann ein Server als eigenständiger Provider oder innerhalb einer Gruppe auftreten. Anhand einer `AffiliationID` können mehrere Provider zu einer Gruppe verbunden werden, welche dann als eine Instanz im Liberty Alliance Verbund angesprochen werden kann.

Benutzerregistrierung

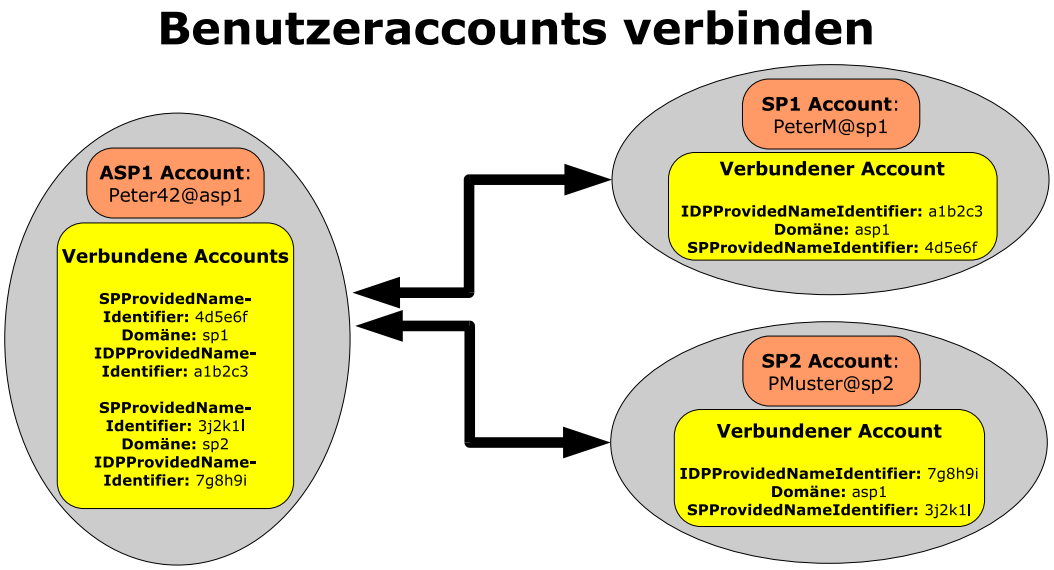

Abbildung 3.12: LA - Benutzeraccounts verbinden

Betrachten wir nun, wie eine Benutzeridentität bei Liberty Alliance verbunden/verknüpft wird.

1. Der Benutzer loggt sich bei einem **ASP** lokal ein, gibt dort z.B. Username und Passwort ein.

2. Der **ASP** fragt den Benutzer, ob er an seinem Circle-of-Trust teilnehmen will und ob der **ASP** ihn bekannt machen darf.

3. Bejaht der Benutzer dieses, so kann er eine Bekanntmachung des Benutzers erreichen via „Identity Provider Introduction Profile“ (LibertyBindProf-Section 2.1) oder auch durch andere, nicht näher spezifizierte Verfahren [CHKT03, S.9]. Es werden aber soweit noch keine Identitätsdaten des Benutzers ausgetauscht.

4. Surft der Benutzer irgendwann danach auf eine Webseite eines **SP** innerhalb des Circle-of-Trust, so wird er sich, wenn er dort einen Account hat, anmelden und anschließend gefragt, ob er seine Identity verknüpfen will. Dazu müssen dem Benutzer aber zuvor noch die Vereinbarungen zwischen den beiden Providern, bezüglich des Umgangs mit seinen Profildaten, angezeigt werden.[28]

5. Bejaht der Benutzer erneut, so können beide Seiten ein Handle für diesen Benutzer ausmachen. Dieses Handle wird `<IDPProvidedNameIdentifier>` genannt und vom **ASP** erzeugt. Zusätzlich kann der **SP** noch ein zweites Handle (`<SPProvidedNameIdentifier>`) beim **ASP** für diesen Benutzer angeben mit Hilfe des „Name Registration Protocols“. Wird ein zweites Handle vom **SP** angemeldet, so muss der **ASP** den `<SPProvidedNameIdentifier>` bei der Kommunikation verwenden und der **SP** den `<IDPProvidedNameIdentifier>`. Dies wird deshalb notwendig, da das vorgegebene Handle vom **ASP** eventuell schon beim **SP** vergeben ist. Sowohl `<IDPProvidedNameIdentifier>` als auch `<SPProvidedNameIdentifier>` sind beide vom Typ $NameIdentifier$.

3.4.4 Benutzer authentisieren

Ich beziehe mich bei der Darstellung des Prozesses auf die ID-FF Spezifikation Version 1.2. Single Sign-On kann bei Liberty Alliance erst dann erreicht werden, wenn zuvor der Benutzer die beiden Accounts verbunden hat. Ist dieses im Voraus geschehen, können vier unterschiedliche Verfahren zur Durchführung von

[28] „Business prerequisites must be met to offer identity federation. Two prerequisites are notifying the user of the capability to federate and soliciting consent to facilitate introductions. Another is creating agreements between the affinity group members to establish their policies for recognizing identities and honoring reciprocal authentication.“ [CHKT03]

Single Sign-On benutzt werden. Das „Browser/Artifact Profile“[29], das „Browser/Post Profile“[30], SOAP-over-HTTP[31] oder mittels LECP[32]. Es wird verlangt, das „Browser/Post Profile“ nur zusätzlich zum „Browser/Artifact Profile“ anzubieten, da es mehr Funktionalität[33] beim Benutzer-Client voraussetzt [CHKT03, S.32].

Ich beschränke mich bei der Darstellung des Single Sign-On Prozesses auf eine mögliche Verfahrensweise. Die anderen Verfahren sind aber weitgehend ähnlich aufgebaut und benutzen zum Teil nur andere Übertragungstechniken.

[29] Dieses Verfahren nutzt QueryStrings, um Nachrichten auszutauschen. Die Querystrings müssen vom MIME-Typ „application/x-www-form-urlencoded“ sein. [AKW03, S.13]

[30] Hier werden die Nachrichten mittels des POST-Verfahren übermittelt.

[31] Bietet Verfahren an, XML-Nachrichten über HTTP zu versenden.

[32] *Liberty enabled Client Proxy* ist ein Proxy-Verfahren und erlaubt somit die Anbindung von funktional eingeschränkten Geräten.

[33] Bei dieser Methode wird Java Script verwendet, um automatisch beim Benutzer ein Submit zum Senden der Formulardaten anzustoßen.

Single Sign-On nach dem „Basic Single Sign-On Profile“ [AKW03]:

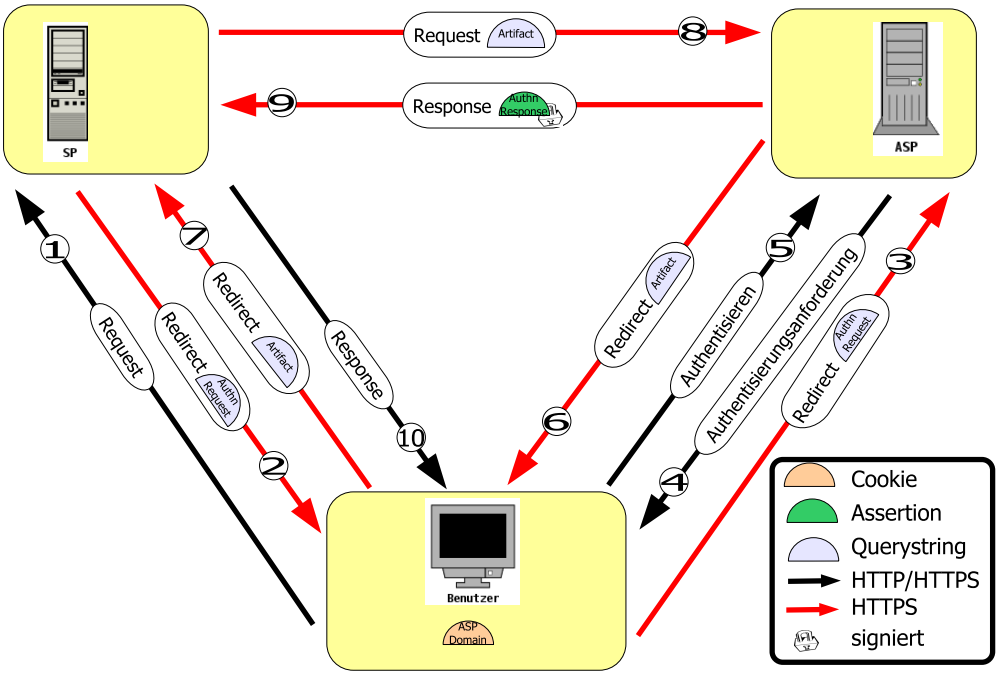

Abbildung 3.13: Liberty Alliance Single Sign-On

In den folgenden Nachrichtenaufbauten sind die Felder unterstrichen, welche enthalten sein müssen.

1. Der Benutzer besucht eine zugangsbeschränkte Webseite eines **SP**. Ist der Benutzer noch nicht beim **SP** authentisiert, so bietet dieser dem Benutzer z.B. eine Liste von **ASP**s[34] an, mit denen der **SP** eine Vertrauensbeziehung führt. Anhand dieser Liste kann der Benutzer sich nun einen **ASP** aussuchen. Liberty Alliance empfiehlt bei diesem Schritt schon das Benut-

[34]Im Falle eines Liberty-enabled Client oder dem Liberty-Proxy besteht für den **SP** die Möglichkeit, beim Client anhand des „Liberty identity provider introduction profile“ abzufragen, welchen **ASP** er wünscht. Eine andere Methode besteht in der Benutzung von Cookies. Entweder hat jeder Provider in dem Circle-of-Trust eine (zusätzliche) Subdomäne in der Domäne des **ASP** oder man nutzt einen zusätzlichen Dienstprovider, welcher für die anderen Provider stellvertretend die Cookies beim Benutzer schreibt und liest.

zen von SSL/TLS, um Vertraulichkeit und Authenzität zu gewährleisten [AKW03, S.19].

2. Nachdem der **ASP** für den Benutzer ermittelt wurde, wird er an diesen mit Hilfe eines Redirect[35] oder einer anderen Methode[36] verwiesen. Liberty Alliance fordert, dass die benutzte URL des **ASP** mittels HTTPS beschrieben wird [ABCS03, S.21]. Bei einem Redirect wird in dem Querystring der URL eine `<AuthnRequest>` Nachricht verpackt. Die `<AuthnRequest>` Nachricht basiert auf der SAML-Struktur `samlp:RequestAbstractType` [OAS03a, S.26] und ist folgendermaßen aufgebaut:

   SP → **ASP** `<AuthnRequest>`:

$$\underline{RequestID}||\underline{MajorVersion}||\underline{MinorVersion}||\underline{IssueInstant}||\underline{ProviderID}$$
$$\underline{\langle Subject\rangle}\,||AffiliationID||NameIDPolicy||IsPassive||ForceAuthn$$
$$ProtocolProfile||AssertionConsumerServiceID||\langle RequestAuthnContext\rangle\,||RelayState$$
$$\langle Scoping\rangle\,||Consent||Extension||SigAlg||Signature$$

wobei

$$\langle Subject\rangle = NameIdentifier||SubjectConfirmation$$

und

$$\langle RequestAuthnContext\rangle = AuthnContextClassRef||AuthnContextStatementRef||AuthnContextComparison$$

und

$$\langle Scoping\rangle = ProxyCount||IDPList$$

ist.

   - $RequestID$: Gibt eine Zeichenkette an, welche in der Antwort vom **ASP** im Feld $InResponseTo$ wiederholt werden muss. Der Wert dieses Feldes muss vom Typ `xsd:ID` sein.
   - $MajorVersion$: Dient zur Angabe der verwendeten Version von Liberty Alliance. Hier: 1
   - $MinorVersion$: Dient zur Angabe der verwendeten Version von Liberty Alliance. Hier: 2
   - $IssueInstant$: Gibt den Zeitpunkt der Erzeugung dieser Nachricht an, z.B. „2004-09-12T10:08:56Z“.
   - $ProviderID$: Gibt in diesem Fall einen Handle an, der den **SP** beim **ASP** eindeutig identifiziert.
   - $\langle Subject\rangle$: Beschreibung und zusätzliche Authentikationsinformationen bezüglich des Benutzers für den die Kommunikation zwischen **SP** und **ASP** stattfindet.
   - $AffiliationID$: Dies ist ein Handle, welches eine „affiliation group“ identifiziert. Also die Zugehörigkeit des **SP** zu einer Gruppe.
   - $NameIDPolicy$: Gibt dem **SP** die Möglichkeit, Einfluss zu nehmen, wie das Subjekt, für das der **ASP** eine Authentication Assertion ausgeben soll, behandelt wird. Hat dieses Feld den Wert „none“ oder fehlt dieses Feld, so wird nur dann eine Authentication Assertion ausgegeben, wenn der Benutzer den **SP** und den **ASP** schon verknüpft hat. Bei „one-time “ generiert der **ASP** ein anonymes Authentication Assertion. Ist „federated “ angegeben, so kann, falls noch keine Verknüpfung existiert, eine solche beim Benutzer beantragt werden. Bei „any“ wird vom **ASP** bei fehlender Verknüpfung einfach ein anonymes Authentication Assertion erzeugt.

[35] Entweder mit HTTP-302 oder auch mittels WML-Redirect.
[36] Möglich wären auch `HTTP-GET` oder `HTTP-POST` Verfahren.

- $IsPassive$: Wenn dieses Feld „True“ ist, darf der **ASP** nicht die Kontrolle des Benutzer User-Interface übernehmen, also z.B. keine Passwortabfrage unternehmen. Hat das Feld den Wert „false“, so kann der **ASP** mit dem Benutzer interagieren. Fehlt dieses Feld, so wird der Wert „true“ angenommen.
- $ForceAuthn$: Wenn dieses Feld „true“ enthält, muss der **ASP** den Benutzer erneut authentisieren. Dieser Fall darf nur eintreten, wenn $IsPassive$ den Wert „false “ besitzt. Wenn dieses Feld fehlt, wird „false “ angenommen.
- $ProtocolProfile$: Gibt das Profil an, welches der Absender für die Antwortnachricht des **ASP** vorgeben kann. Fehlt dieses Feld wird „Browser/Artifact Profile“benutzt.
- $AssertionConsumerServiceID$: Gibt in der Regel die URL an, welche der **ASP** für die Antwortnachricht benutzen soll.
- $\langle RequestAuthnContext \rangle$: Beschreibt das gewünschte Authentikationsverfahren des **SP**, also welche Voraussetzungen erfüllt sein müssen, damit der **SP** die Authentikation des Benutzers beim **ASP** akzeptiert.
- $RelayState$: Gibt eine zusätzliche Zeichenkette an, die in der Antwort des **ASP** wiederholt werden muss.
- $\langle Scoping \rangle$: Erlaubt die Angabe und Reihenfolge von weiteren **ASP**s, welche der kontaktierte **ASP** nutzen soll, um den Benutzer zu authentisieren. Dies wird für einen Proxy-**ASP** benutzt.
- $Consent$: Dieses Feld gibt an, ob die Erlaubnis beim Benutzer eingeholt wurde, um diese Nachricht zu schicken.
- $Extension$: Dieses Feld kann frei vom **SP** belegt werden, um zusätzliche Informationen zum **ASP** zu übertragen.
- $SigAlg$: Gibt den Signieralgorithmus an und eventuell noch zusätzliche Informationen zur Signatur. Es können `DSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#dsa-sha1`) oder `RSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#rsa-sha1`) benutzt werden.
- $Signature$: Enthält die Signatur der signierten Querydaten bis $SigAlg$, also nur alles ab „?“ in der URL.

$\langle Subject \rangle$ enthält

- $NameIdentifier$: Bezeichner des Subjektes und eventuell zugehörige Security Domain.
- $SubjectConfirmation$: Informationen für die Authentikation des Subjektes, z.B. bei Challenge-Response Verfahren.

$\langle RequestAuthnContext \rangle$ besteht aus

- $AuthnContextClassRef$: Enthält eine geordnete Liste der Authentikationskontextklassen, die der **SP** akzeptiert.
- $AuthnContextStatementRef$: Enthält eine geordnete Liste der Authentikationsverfahren, welche der **SP** akzeptiert. Es dürfen nicht dieses Feld und das darüber Beschriebene zugleich benutzt werden.
- $AuthnContextComparison$: Gibt an, wie der **ASP** die Anforderungen des **SP** zu erfüllen hat. Ein „exact“ gibt an, dass das Verfahren beim **ASP** exakt mit einem in der übergebenen Liste übereinstimmen muss. „minimum“ erlaubt dem **ASP**, ein Verfahren zu nutzen, welches zumindest genauso gut wie die übergebenen ist. „better “ gewährt dem **ASP** ein Authentikationsverfahren zu benutzen, das auf jeden Fall besser sein muss, als die angegebenen. „maximum“ erlaubt den **ASP** ein Verfahren zu verwenden, welches zwar nicht in der Liste ist, aber mindestens so sicher ist, wie alle übergebenen Verfahren.

und $\langle Scoping \rangle$ aus

- $ProxyCount$: Die maximale Anzahl von zu benutzenden Stationen (**ASP**s), die für die Authentisierung des Benutzers erlaubt sind.
- $IDPList$: Eine geordnete Liste von **ASP**s, welche für die Authentisierung des Benutzers benutzt werden können.

3. Der Webbrowser des Benutzers erkennt den Redirect und kontaktiert den **ASP** inklusive der <AuthnRequest> Nachricht im Querystring.

4. Ist der Benutzer noch nicht beim **ASP** angemeldet, so fordert nun der **ASP** den Benutzer auf, sich zu authentisieren. Liberty Alliance schreibt kein konkretes Verfahren vor. Es wird aber ein Nachrichtentyp angegeben, um die Art des Anmeldeverfahrens an die anderen Teilnehmer zu übertragen. Mit Hilfe dieser Nachricht kann z.B. ein **SP** feststellen, ob ihm das Authentikationsverfahren ausreicht. Es könnten für die Authentisierung des Benutzers, z.B. Benutzername/Passwort Abfragen, Zertifikate oder sonstige Verfahren genutzt werden. Hat der **ASP** den Benutzer verifiziert, so leitet der **ASP** den Benutzerclient erneut zum **SP** um. Liberty Alliance verlangt das dies mittels HTTPS geschieht [AKW03, S.24]. In der Redirect-URL werden entweder die Nachricht <AuthnResponse> mit den Authentication Assertions direkt übertragen oder ein Artefakt, welches auf diese Authentication Assertions beim **ASP** referenziert. Zudem besteht die Möglichkeit, die Authentication Assertions mittels des POST-Profile zu versenden.

   In diesem Beispiel gehe ich von der Übertragung eines Artefakts aus, das auf Authentication Assertions beim **ASP** verweist. Interessanterweise werden bei dieser Nachricht keine zusätzlichen Liberty Alliance Headerdaten eingefügt, sondern ein reines SAML-Artifact.

   ASP → SP:

$$RelayState || \langle SAML - Artifact \rangle$$
mit
$$\langle SAMLArtifact \rangle = \underline{TypeCode} || \underline{IdentityProviderSuccinctID} || \underline{AssertionHandle}$$

   - $RelayState$ = Dieses Feld muss vorhanden sein, wenn bei der <AuthnRequest> das gleiche Feld gesetzt wurde [AKW03, S.26]. Der Inhalt ist die Wiederholung der Zeichenkette aus der Anfragenachricht.
   - $\langle SAMLArtifact \rangle$ = Dies ist das SAMLArtefaktt, mit dessen Hilfe der **SP** an das Authentication Assertion beim **ASP** gelangen kann.
   - $TypeCode$ = Ist eine 2-Byte lange Kennung, die bei Liberty Alliance mit dem Wert 0x0003 gefüllt wird.
   - $IdentityProviderSuccinctID$ = Dies ist ein 20-Byte langes Feld, das einen Identifizierungsstring des **ASP** beinhaltet. Anhand dieser Kennung kann der **SP** in einer lokalen Tabelle die notwendigen Daten referenzieren, um den **ASP** für die <AuthnResponse> Nachricht zu kontaktieren. Liberty Alliance beschreibt nicht, wie diese Tabelle beim **SP** aufgebaut wird.

- $AssertionHandle$ = Dieses 20-Byte lange Feld enthält eine eindeutige Kennung für die referenzierte Authentication Assertion. Der Wert muss mit einem vertrauenswürdigen Zufallsgenerator erzeugt werden und soll mindestens 8 Byte lang sein [ABCS03, S.25].

5. Erneut führt der Webbrowser beim Client den Redirect aus und leitet den Querystring an den **SP** weiter.

6. Der **SP** extrahiert das Artefakt. Anhand der $IdentityProviderSuccinct-ID$ und $AssertionHandle$ besitzt der **SP** die notwendigen Daten, um beim **ASP** die Authentication Assertions anzufordern. Dazu nutzt er eine <samlp:Request>[37] Nachricht.

   In diesem Beispiel nutzt der **SP** nun SOAP-over-HTTP, um direkt mit dem **ASP** zu kommunizieren. Die SAML-Spezifikation verlangt in diesem Schritt die Benutzung von SSL3.0/TLS mit Serverzertifikatsüberprüfung [OAS03b, S.16].
   SP → ASP:

   $$\underline{RequestID}||\underline{MajorVersion}||\underline{MinorVersion}||\underline{InResponseTo}||\underline{IssueInstant}||SigAlg||Sigature||AssertionIDReference||AssertionArtifact$$

   - $RequestID$: Gibt eine Zeichenkette an, die in der Antwort vom **ASP** im Feld $InResponseTo$ wiederholt werden muss. Der Wert dieses Feldes muss vom Typ `xsd:ID` sein.
   - $MajorVersion$: Dient zur Angabe der verwendeten Version von *SAML*. Hier: 1
   - $MinorVersion$: Dient zur Angabe der verwendeten Version von *SAML*. Hier: 1
   - $InResponseTo$: Wiederholung der Zeichenkette, welche zuvor im Feld $ResponseID$ empfangen wurde. Die Liberty Alliance Spezifikation geht auf dieses Feld nicht genauer ein. Da bei der Übertragung des Artefakts vom **ASP** zum **SP** keine $ResponseID$ angegeben wird, macht dieses Feld in diesem Fall keinen Sinn.
   - $IssueInstant$: Gibt den Zeitpunkt der Erzeugung dieser Nachricht an, z.B. „2004-09-12T10:08:56Z“.
   - $SigAlg$: Gibt den Signieralgorithmus an und eventuell noch zusätzliche Informationen zur Signatur. Es können `DSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#dsa-sha1`) oder `RSA-withSHA1` (`http://www.w3.org/2000/09/xmldsig#rsa-sha1`) benutzt werden.
   - $Signature$: Enthält die Signatur der signierten Querydaten bis $SigAlg$.
   - $AssertionIDReference$: Referenz auf das gewünschte *AuthenticationAssertion*. Dieses Element kann auch mehrmals vorkommen, falls mehrere Assertions angefordert wurden.
   - $AssertionArtifact$: Das zuvor erhaltene Artefakt bei der Authentikationsanfrage. Dieses Element kann auch mehrmals vorkommen, falls mehrere Assertions angefordert wurden.

[37] samlp: Namespace aus der SAML-Spezifikation.

7. Der **ASP** überprüft die Rechtmäßigkeit der Anfrage[38] vom **SP** und übermittelt dem **SP** eine <AuthnResponse> Nachricht mit den gewünschten Assertions.

   Eine <AuthnResponse> Nachricht ist folgendermaßen aufgebaut:
   ASP → SP:

$$Extension||\underline{ProviderID}||RelayState||ResponseID||InResponseTo||MajorVersion||$$
$$MinorVersion||Consent||IssueInstant||\underline{\langle Status\rangle}||\langle Assertions\rangle||SigAlg||Signature$$

wobei

$$\langle Status\rangle = \underline{StatusCode}||StatusMessage||StatusDetail$$

und

$$\langle Assertions\rangle = \underline{MajorVersion}||\underline{MinorVersion}||\underline{AssertionID}||\underline{Issuer}||\underline{IssueInstant}||$$
$$\underline{InResponseTo}||\langle Conditions\rangle||\langle AuthenticationStatement\rangle||\underline{SigAlg}||\underline{Signature}$$

und

$$\langle Conditions\rangle = NotBefore||NotOnOrAfter||AudienceRestrictionCondition$$

und

$$\langle AuthenticationStatement\rangle = AuthenticationInstant||\underline{SessionIndex}||$$
$$ReauthenticateOnOrAfter||AuthenticationContext||\langle SubjectStatement\rangle$$

und

$$\langle SubjectStatement\rangle = \underline{NameIdentifier}||\underline{NameQualifier}||\underline{Format}||$$
$$\underline{SubjectConfirmation}||\underline{\langle IDPProvidedNameIdentifier\rangle}$$

und

$$\langle IDPProvidedNameIdentifier\rangle = \underline{NameIdentifier}||NameQualifier||Format$$

   - $Extension$: Dieses Feld kann frei vom **ASP** belegt werden, um zusätzliche Informationen zum **SP** zu übertragen.
   - $ProviderID$: Die eindeutige Kennung des **ASP**.
   - $RelayState$: Enthält eine vom **SP** vorgegebene Zeichenkette, die vom **ASP** wiederholt werden muss.
   - $ResponseID$: Enthält eine Zeichenkette, welche bei einer Antwort vom **SP** in dem Feld $InResponseTo$ wiederholt werden muss. Da aber diese Nachricht den Abschluss der Authentikationszeugnisanfrage bildet und Liberty Alliance keine weitere Antwortnachricht spezifiziert, ist der Inhalt nicht relevant.
   - $InResponseTo$: Dies muss den Wert aus dem $RequestID$ Feld in der <AuthnRequest> Nachricht wiederholen. Dieser zusätzliche Replayschutz ist notwendig, da nicht eine ganze <AuthnResponse> Nachricht signiert werden muss. Es muss aber zumindest jedes enthaltene Authentication Assertion signiert werden. Die einzige Ausnahme, bei der dieses Feld fehlt, ist, wenn der **ASP** ohne eine Anfrage eine <AuthnResponse> Nachricht an den **SP** verschickt.
   - $MajorVersion$: Dient zur Angabe der verwendeten Version von Liberty Alliance. Hier: 1
   - $MinorVersion$: Dient zur Angabe der verwendeten Version von Liberty Alliance. Hier: 2
   - $Consent$: Dieses Feld gibt an, ob die Erlaubnis beim Benutzer eingeholt wurde, um diese Nachricht zu verschicken.
   - $IssueInstant$: Gibt den Zeitpunkt der Erzeugung dieser Nachricht an, z.B. „2004-09-12T10:08:56Z".
   - $\langle Status\rangle$: Dieses Feld gibt den Status der korrespondierenden Anfrage zurück.
   - $\langle Assertions\rangle$: Hier sind die angeforderten Assertions untergebracht.

[38] Liberty Alliance verlangt keine genaue Art, wie dies geschehen soll. *SAML* verlangt, dass die Herausgabe nur an den **SP** erlaubt ist, der zuvor das Artefakt erhalten hat. Der **ASP** kann z.B. eine Liste von **SP**s und ausgelieferten Artefakten vorhalten.

- $SigAlg$: Gibt den Signaturalgorithmus an und eventuell noch zusätzliche Informationen zur Signatur. Es können `DSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#dsa-sha1`) oder `RSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#rsa-sha1`) benutzt werden.
- $Signature$: Dieses Feld enthält den Signaturwert.

$\langle Status \rangle$ besteht aus

- $StatusCode$: Enthält den Statuswert der Anfrage. Zusätzlich können noch weitere $StatusCodes$ verschachtelt angegeben werden, um präzisere Informationen anzugeben, z.B. wenn mehrere Bedingungen zutreffen müssen für eine positive Rückgabemeldung. Einige vordefinierte Werte finden sich unter [OAS03a, S.34].
- $StatusMessage$: Hier kann eine Meldung untergebracht werden, welche dem Anfragenden angezeigt werden kann.
- $StatusDetail$: Dieses Feld kann zusätzliche Informationen enthalten, falls der StatusCode nicht positiv ausfällt.

$\langle Assertions \rangle$ besteht aus

- $MajorVersion$: Kennzeichnet die verwendete SAML-Version, bei Liberty Alliance muss hier 1 stehen.
- $MinorVersion$: Kennzeichnet die verwendete SAML-Version, bei Liberty Alliance muss hier 1 stehen.
- $AssertionID$: Muss ein eindeutiger Bezeichner für dieses Authentikationszeugnisses sein.
- $Issuer$: Kennzeichnung des **ASP**, welcher dieses Assertion ausgestellt hat.
- $InResponseTo$: Enthält den Wert aus dem $RequestID$ von der `<AuthnRequest>` Nachricht vom **SP**. Es gibt eine Ausnahme, wann dieses Feld nicht benutzt wird. Dies ist dann der Fall, wenn der **ASP** ohne Anfrage eine `<AuthnResponse>` Nachricht verschickt.
- $\langle Conditions \rangle$: Bedingungen, die bei der Auswertung der Assertion beachtet werden müssen.
- $\langle AuthenticationStatement \rangle$: Hier wird eine Aussage über den Authentikationprozess vorgenommen.
- $\langle SubjectStatement \rangle$: In diesem Feld wird festgelegt, über welche Person oder welchen Prozess das Assertion eine Aussage trifft.
- $SigAlg$: Gibt den Signaturalgorithmus an und eventuell noch zusätzliche Informationen zur Signatur. Es können `DSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#dsa-sha1`) oder `RSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#rsa-sha1`) benutzt werden.
- $Signature$: Dieses Feld enthält den Signaturwert.

$\langle Conditions \rangle$ besteht aus

- $NotBefore$: Enthält den Zeitpunkt, ab wann das ausgestellte Assertion erst Gültigkeit besitzt.
- $NotOnOrAfter$: Enthält den Zeitpunkt, ab wann das ausgestellte Assertion ungültig wird.
- $AudienceRestrictionCondition$: Mit diesem Feld kann man den Verwendungszweck und Empfänger für dieses Assertion festlegen.

$\langle AuthenticationStatement \rangle$ besteht aus

- $AuthenticationInstant$: Der Zeitpunkt, an dem die Authentikation vom **ASP** durchgeführt wurde.
- $SessionIndex$: Enthält einen Wert, der beim **ASP** eine bestimmte Sitzung mit dem Benutzer kennzeichnet.
- $ReauthenticateOnOrAfter$: Dieses Feld gibt den Zeitpunkt an, an welchem der **SP** den Benutzer zum Zwecke einer neuen Authentisierung erneut an den **ASP** verweisen muss.

- $AuthenticationContext$: Gibt das Verfahren und die Umgebung an, die der **ASP** benutzt hat um den Benutzer zu authentisieren, und somit zu diesem Assertion geführt hat. Das zugehörige umfangreiche XML-Schema unter [Lib03a] beschreibt dieses Feld ausführlich.

$\langle SubjectStatement \rangle$ besteht aus

- $NameIdentifier$: Gibt an, für wen die Authentication Assertion ausgestellt wurde. Haben **SP** und **ASP** unterschiedliche Handles für einen Benutzer, so muss hier der Handle des **SP** eingetragen sein. Es sei noch erwähnt, dass es eine besondere Erweiterung bei Liberty Alliance von diesem Feld gibt, und zwar das $EncryptableNameIdentifier$. Der $EncryptableNameIdentifier$ dient für die Übertragung einer verschlüsselten Benutzerbezeichnung, falls der **SP** oder **ASP** als Proxystation benutzt werden. Somit kann die Proxystation keine weiteren Informationen aus dem Feld $NameIdentifier$ gewinnen.
- $NameQualifier$: Kennzeichnet einen eindeutigen Bezeichner, in dessen Kontext der $NameIdentifier$ gültig ist. Dies ist in diesem Fall der Bezeichner des **SP** oder seine $AffiliationID$.
- $Format$: Beschreibt das Format, die Semantik und Verarbeitungsregeln des $NameQualifier$s in der Authentication Assertion. Es stehen 4 unterschiedliche Möglichkeiten bereit. $federated$ gilt, wenn der Benutzer die kommunizierenden Provider bereits verknüpft hat. $one-time$ wird benutzt, wenn es sich um eine „anonyme" Erklärung handelt. Diese Methode bietet dem Benutzer die Möglichkeit, bestimmte Informationen über sich selbst einem **SP** zugänglich zu machen, ohne seine Identität zu offenbaren. $encrypted$ wird dann eingesetzt, wenn diese Authentication Assertion nicht direkt für den **SP** bestimmt sind, sondern der **SP** als Proxy für einen anderen Provider agiert. Es wird dann der $NameIdentifier$ und der $NameQualifier$ verschlüsselt. $entityID$ kennzeichnet, dass diese Erklärung für einen Liberty Provider oder einer „affiliation group" ausgegeben wurde. Dieses kann für Authentication Assertion zwischen Providern benutzt werden.
- $SubjectConfirmation$: Dieses Feld hat als einziges Element $ConfirmationMethod$ mit dem Wert `urn:oasis:names:tc:SAML:1.0:cm:artifact`.
- $\langle IDPProvidedNameIdentifier \rangle$: Beschreibt das Handle des Benutzers beim **ASP**.

$\langle IDPProvidedNameIdentifier \rangle$ besteht aus

- $NameIdentifier$: Hier steht ein String, der den Benutzer beim **ASP** eindeutig identifiziert.
- $NameQualifier$: Mit Hilfe dieses Feldes kann die zugehörige Kontextdomäne des Benutzers angegeben werden. Somit kann es nicht zu Kollisionen bei unterschiedlichen Providern mit schon vergebenen Namen kommen.
- $Format$: Dieses Feld hat den selben Zweck wie schon bei $\langle SubjectStatement \rangle$ beschrieben.

8. Der **SP** kann den Status und das benutzte Verfahren der Anmeldung des Benutzers beim **ASP** feststellen. Anhand dieser Information kann er eine strengere Authentisierungsmethode fordern. Wenn dies nicht notwendig ist, startet der **SP** seine eigene Session mit dem Benutzer und sieht ihn als authentisiert an.

   Beim Reauthentisieren lässt die Spezifikation offen, wer dieses durchführen soll. Der Vorgang kann sowohl vom **SP** oder vom **ASP** angestoßen werden. Möglicherweise ist es sogar notwendig, dass der **SP** anschließend seinen eigenen Authentisierungsprozess mit dem Benutzer beginnt, z.B. weil der **SP** bei bestimmten Dienstleistungen eine stärke Authentisierung fordert,

als die vom **ASP** durchgeführte. Dieses könnte bei Kontoabbuchungsaufträgen bei einer Bank der Fall sein.

3.4.5 Single Logout bei Liberty Alliance

Der Single Logout Prozess kann bei Liberty Alliance sowohl bei einem **ASP**, als auch bei einem **SP** angestoßen werden.

Eine Session sollte von einem Teilnehmer erst dann beendet werden, wenn eines der drei Ereignisse eintritt [ABCS03, S.10]

- <LOGOUT-REQUEST> wird empfangen.
- Die normalen Timeout Zeiten sind überschritten
- ReauthenticateOnOrAfter wird, falls angegeben, erreicht.

Liberty Alliance bietet drei Verfahren für das Single Logout an:

- HTTP-Redirect-Based: Benutzt HTTP 302 Nachrichten und kann beim **SP** oder **ASP** beginnen.
- HTTP-GET-Based: Basiert auf HTTP GET-Methoden und IMG-Tags. Kann nur beim **ASP** initiiert werden.
- SOAP/HTTP-Based: Funktioniert über SOAP-over-HTTP Nachrichten. Kann beim **SP** oder **ASP** beginnen.

Beispiel eines Single Sign-Out Prozesses

Beginnt der Single Sign-Out Prozess bei einem **SP**, so schickt dieser dem **ASP** eine <LogoutRequest> Nachricht. Eine <LogoutRequest> Nachricht besteht aus folgenden Feldern (unterstrichene Felder müssen enthalten sein):

SP $\rightarrow$ **ASP**:

$\underline{Extension}\|\|\underline{MajorVersion}\|\|\underline{MinorVersion}\|\|\underline{IssueInstant}\|\|\underline{RequestID}\|\|\underline{ProviderID}\|\|$ $\underline{NameIdentifier}\|\|\underline{SessionIndex}\|\|\underline{RelayState}\|\|\underline{Consent}\|\|\underline{RespondWith}\|\|\underline{SigAlg}\|\|\underline{Signature}$

- $\underline{Extension}$: Bietet Platz für zusätzliche Informationen für die kommunizierenden Provider. Der Inhalt wird aber nicht von der Liberty Alliance vorgegeben. Erfordert bei der Benutzung eine vorherige Absprache der Provider.
- $\underline{MajorVersion}$: Kennzeichnet die verwendete SAML-Version, bei Liberty Alliance muss hier 1 stehen.

- $\underline{MinorVersion}$: Kennzeichnet die verwendete SAML-Version, bei Liberty Alliance muss hier 1 stehen.
- $\underline{IssueInstant}$: Gibt den Zeitpunkt der Erzeugung dieser Nachricht an, z.B. „2004-09-12T10:08:56Z“.
- $\underline{RequestID}$: Gibt eine Zeichenkette an, welche in der Antwort vom **ASP** im Feld $InResponseTo$ wiederholt werden muss. Der Wert dieses Feldes muss vom Typ `xsd:ID` sein.
- $\underline{ProviderID}$: Die ID des Providers, der diese Nachricht versendet. In diesem Fall die ID des **SP**.
- $\underline{NameIdenifier}$: Das zwischen den Providern vereinbarte Benutzerhandle, eventuell inklusive Domainbezeichnung und Formatbeschreibung.
- $SessionIndex$: Dieses Feld muss ausgefüllt sein, wenn zuvor bei dem „authentication statement“ ein solcher Index vereinbart wurde.
- $RelayState$: Gibt einen Nounce für eine Antwortnachricht an.
- $Consent$: Gibt an, ob der Benutzer um Erlaubnis gefragt wurde, diese Nachricht zu schicken.
- $RespondWith$: Dieses Feld enthält die Adresse beim **SP**, an die der **ASP** seine Antwortnachricht schicken soll. Besteht in der Regel aus einer URL.
- $SigAlg$: Gibt den Signaturalgorithmus an und eventuell noch zusätzliche Informationen zur Signatur. Es können `DSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#dsa-sha1`) oder `RSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#rsa-sha1`) benutzt werden.
- $Signature$: Dieses Feld enthält den Signaturwert.

Der **ASP** prüft, falls vorhanden, die Signaturen der Nachricht und ob diese auch von einem **SP** kommen, dem er vorher eine Authentication Assertion übermittelt hat. Anschließend schickt der **ASP** an alle **SP**s auch eine <LogoutRequest> Nachricht, jedoch nicht an den **SP**, der ihm das Ausloggen signalisiert hat. Ausserdem wird auch jedem **ASP** eine <LogoutRequest> Nachricht übermittelt, für die der **ASP** eine Proxyfunktion übernommen hat. Zuletzt beendet er die Loginsession des Benutzers und löscht auch jeden zugehörigen <SessionIndex>.

Jeder Provider, der ein <LogoutRequest> erhalten hat, muss zunächst die Signaturen der Nachricht überprüfen und falls er eine Authentication Assertion zuvor vom **ASP** erhalten hat, diese löschen. Anschließend antwortet er mit einer <LogoutResponse> Nachricht:

ASP $\rightarrow$ **SP**:

$$Extension||\underline{MajorVersion}||\underline{MinorVersion}||\underline{IssueInstant}||\underline{ResponseID}||\underline{InResponseTo}||\underline{ProviderID}||\underline{Status}||RelayState||SigAlg||Signature$$

- $Extension$: Bietet Platz für zusätzliche Informationen für die kommunizierenden Provider. Der Inhalt wird aber nicht von der Liberty Alliance vorgegeben. Erfordert bei der Benutzung eine vorherige Absprache der Provider.
- $\underline{MajorVersion}$: Kennzeichnet die verwendete SAML-Version, bei Liberty Alliance muss hier 1 stehen.

- $\underline{MinorVersion}$: Kennzeichnet die verwendete SAML-Version, bei Liberty Alliance muss hier 1 stehen.
- $\underline{IssueInstant}$: Gibt den Zeitpunkt der Erzeugung dieser Nachricht an, z.B. „2004-09-12T10:08:56Z“.
- $\underline{ResponseID}$: Enthält eine Zeichenkette, welche bei einer Antwort in dem Feld $InResponseTo$ wiederholt werden muss. In der Regel erfolgt aber auf diese Nachricht keine Antwort.
- $\underline{InResponseTo}$: Enthält den Wert aus dem Feld $RequestID$ aus der `<LogoutRequest>` Nachricht.
- $\underline{ProviderID}$: Die ID des Provider, der diese Nachricht versendet.
- $\underline{Status}$: Gibt den Status der Abmeldung an. Es muss zumindest einen $StatusCode$ enthalten, kann aber darüber hinaus noch aus $StatusMessage$ und $StatusDetail$ bestehen. Den genauen Aufbau dieser Werte wird in der SAML Spezifikation behandelt.
- $RelayState$: Enhält den selben Wert wie das gleichnamige Feld in der `<LogoutRequest>` Nachricht.
- $SigAlg$: Gibt den Signaturalgorithmus an und eventuell noch zusätzliche Informationen zur Signatur. Es können `DSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#dsa-sha1`) oder `RSAwithSHA1` (`http://www.w3.org/2000/09/xmldsig#rsa-sha1`) benutzt werden.
- $Signature$: Dieses Feld enthält den Signaturwert.

Hat der **ASP** alle Rückmeldungen von den Stationen erhalten, denen er eine `<LogoutRequest>` zugesandt hatte, so schickt er dem **SP**, welcher das Ausloggen initiiert hatte, dann abschließend auch eine `<LogoutResponse>` Nachricht. Tritt in diesem Abmeldeprozess ein Fehler auf, so kann dieses innerhalb des $Status$ Feldes bekannt gemacht werden.

3.4.6 Sonstige Bemerkungen zu Liberty Alliance

- Mit einer `<RegisterNameIdentifierRequest>` Nachricht kann ein **ASP** einen neuen Handle für einen Benutzer beim **SP** aktualisieren.
- Wünscht ein Benutzer eine Verknüpfung zwischen zwei Providern aufzulösen, so wird dies mit einer `<FederationTerminationNotification>` Nachricht zwischen den beiden Providern signalisiert.[ABCS03, S.36]
- Ein **SP** kann als selbstständiger Provider innerhalb eines Circle-of-Trust auftreten oder aber innerhalb einer Gruppe („affiliation group“) in Erscheinung treten. Das Auftreten kann anhand des `NameQualifier` für die Kommunikationspartner unterschieden werden.

- Anhand des „Name Identifier Mapping Protocol“ bietet sich die Möglichkeit für einen **SP**, sich mit einem anderen **SP** über einen Benutzer zu unterhalten, ohne dass der Benutzer diese beiden **SP** verknüpft hat. Es werden aber keine Klartextinformationen zwischen diesen **SP**s ausgetauscht [ABCS03, S.41]. Der Trick besteht in dem Übertragen von verschlüsselten Handles an die jeweiligen **SP**s.

- Wird SOAP-over-HTTP benutzt, erlaubt die Liberty Alliance keine Authentisierung, HTTP-basic-client-authentication [RFC2617] und/oder SSL 3.0 oder TLS 1.0[39]. In dem Single Sign-On Beispiel dieser Diplomarbeit wurde ein bestimmtes Liberty Alliance Profil benutzt. Dieses konkrete Profil schreibt an den aufgezeigten Schritten zwingend die Benutzung von SSL/TLS vor.

- Wird bei dem Transport von Nachrichten Integrität und Vertraulichkeit verlangt, so fordert Liberty Alliance mindestens SSL 3.0 oder TLS 1.0 mit Serverzertifikat. Werden die Liberty Alliance Nachrichten selbst signiert, so sollten die benutzten Schlüssel keine Gemeinsamkeiten mit den Schlüsseln für den Transportkanal aufweisen. Es wird in der Spezifikation sehr ausdrücklich auf Probleme bei fehlender Benutzung von kryptographischen Methoden eingegangen.

  „Providers MUST use secure transport (https) to achieve confidentiality and integrity protection. The initiator of the secure connection MUST authenticate the server using server-side X.509 certificates.“ [AKW03, S.11]

  „For signing and verification of protocol messages, identity and service providers SHOULD use certificates and private keys that are distinct from the certificates and private keys applied for SSL or TLS channel protection.“ [AKW03, S.12]

[39] Bei SSL/TLS werden sowohl Server-, wie auch Clientseitige-Authentisierung unterstützt.

Kapitel 4

Risikoanalyse von Microsoft Passport und Liberty Alliance

4.1 Die Risikoanalyse

Das Untersuchen und Bewerten von IT-Systemen hinsichtlich ihrer Sicherheitsgüte, ist ein Hauptbestandteil der IT-Sicherheit. Zunächst kann man die Sicherheit von Systemen in zwei Kategorien aufteilen.

- **safety**: Bezieht sich auf die Sicherheit des Gesamtsystems vor „katastrophalen" Gefahren. Dies können Auswirkungen von Natureinflüssen sein oder auch nur der unbeabsichtigte Ausfall von Ressourcen und Diensten.
- **security**: Betrachtet den Schutz des Gesamtsystems vor Schäden, die durch menschliche Aktionen ausgelöst werden. Dabei muss es sich nicht um einen bewussten und gezielten Vorgang, seitens des Menschen handeln. Auch unbewusste und versehentlich ausgelöste Aktionen gehören dazu.

Diese Arbeit beschäftigt sich mit der „security" der vorgestellten Systeme.

Die IT-Sicherheit ist ein relativ junger Forschungszweig innerhalb der Computerwissenschaften. Folgende Thesen können aus den Erkenntnissen der IT-Sicherheit gewonnen werden.

- Es kann keine 100-prozentige Sicherheit von IT-Systemen erreicht werden.
- Sicherheit kann in der Regel bei komplexen Systemen nie bewiesen werden, sondern nur deren Unsicherheit.

- Sicherheit ist kein statisches Moment, das irgendwann erreicht wird, sondern ist ein dynamischer Prozess, welcher immer wieder überprüft werden muss.
- Schutzmaßnahmen zur Steigerung der Sicherheit sind eine Kosten-/Nutzenrechnung. Welche Daten und Verarbeitungsschritte will man mit bestimmten Schutzmaßnahmen vor konkreten Gefahren schützen?
- Die Kosten für die Einführung einer Schutzmaßnahme sollten nicht den Wert der zu schützenden Güter übersteigen.
- Praktizierte IT-Sicherheit ist keine abgeschottete Laborwissenschaft, sondern wird von Menschen, Prozessen, Umgebungen und Technologien beeinflusst.

Innerhalb der IT-Sicherheit gibt es zwei unterschiedliche Ansätze, die Sicherheit eines IT-Systems oder Teile davon zu bewerten und gegebenenfalls zu erhöhen, der Grundschutzansatz und der Risikoanalyseansatz.

- **Grundschutzmaßnahmen (engl. baseline security measures)**: Das Sicherheitskonzept wird durch die Auswahl für einen bestimmten Anwendungsbereich für notwendig erachtet und durch allgemein anerkannte Sicherungsmaßnahmen entwickelt. Die Sicherungsmaßnahmen werden anhand von fertigen Bausteinen erstellt, die schon zuvor in einem vergleichbaren Umfeld eingesetzt wurden. Meist geschieht dies mit Hilfe von Checklisten und einer Auswahl von vorgefertigten Komponenten. Es erfolgt keine Analyse des aktuellen Systems mitsamt dessen Risiken und auch keine anschließende Erarbeitung von Risikoreduzierungsmechanismen. Ein klassischer Vertreter dieses Ansatzes ist das „IT-Grundschutzhandbuch“ vom „Bundesamt für Sicherheit in der Informationstechnik “ (BSI) [Bun03].
- **Risikoanalyse (engl. Riskanalysis)**: Die Risikoanalyse ist ein Prozess, bei dem ein bestimmtes IT-System oder Teile davon untersucht werden. Ziel ist die systematische Erkennung von Werten, Ressourcen und Verarbeitungsschritten und den Gefahren, denen diese Objekte ausgesetzt sind. Dies erfordert zudem ein Bewertungssystem, um die Kosten beim Eintritt

einer Beeinträchtigung des Sicherheitsniveaus zu beschreiben. Ein häufiges Problem bei der Risikoanalyse ist die Berechnung der Häufigkeit oder Wahrscheinlichkeit des Eintritts eines Schadenfalles oder Angriffs. Der Vorteil dieser Methode liegt in der detaillierten Untersuchung des konkreten Systems, ohne auf vorgefertigte Standardkomponenten und Checklisten zurückzugreifen.

Der Grundschutzansatz bietet somit ein Verfahren an, IT-Systeme mit einem niedrigen bis mittleren Schutzbedarf abzusichern. Wird ein höheres Sicherheitsniveau verlangt, so sollte man unbedingt die detaillierte Risikoanalyse wählen. Kritiker des Grundschutzansatzes bemängeln die Allgemeingültigkeit von Verfahren und die Auswahl von vorgefertigten Sicherheitsservices zur Erreichung eines Sicherheitsniveaus. Sie vertreten den Standpunkt, dass

1. in vielen Organisationen sich die Struktur, Bedeutung und das Umfeld der eingesetzten IT-Systeme sowie die zu schützenden Werte unterscheiden.

2. keine Standardvorschläge für angemessene Sicherungsmaßnahmen existieren können. Schon innerhalb einer komplexen Organisation ergeben sich meist sehr unterschiedliche Sicherheitsanforderungen.

3. es eine detaillierte Analyse der Systeme erfordert, um Aussagen über den Schutzbedarf zu treffen und anschließend die notwendigen Sicherungsmaßnahmen auszuwählen.

[Ste02, S.5]

In dieser Diplomarbeit wird der Ansatz der Risikoanalyse verfolgt. Deshalb soll zunächst die Risikoanalyse genauer innerhalb der IT-Sicherheit betrachtet werden.

Die ISO/IEC[1] bezeichnet *Risiko* als eine

> „Kombination der Wahrscheinlichkeit des Auftretens eines Schadens und des Schweregrades dieses Schadens." [Sta99b]

[1] ISO steht für „International Standardisation Organisation" und IEC für „International Electrotechnical Commission".

Schaden wird als eine

> „Physische Verletzung oder Schädigung der Gesundheit von Menschen oder Schädigung von Gütern oder der Umwelt.“ [Sta99b]

definiert.

Dies bedeutet, ein Risiko in der IT-Sicherheit wird als Verknüpfung von der Wahrscheinlichkeit oder Häufigkeit des Eintreffens eines Vorfalls mit dem dadurch entstandenen Schaden betrachtet. Ein Risiko ist kein statischer Zustand, die Sicherheit eines IT-Systems ändert sich in der Regel ständig. Es werden Komponenten und Verfahren entfernt, hinzugefügt oder verändert.

Zur Erkennung und Bemessung eines Risikos wird die Risikoanalyse eingesetzt. Die ISO/IEC definiert die Risikoanalyse als

> „die systematische Auswertung verfügbarer Informationen, um Gefährdungen zu identifizieren und Risiken abzuschätzen.“ [Sta99b]

Die Risikoanalyse dient somit zur Aufdeckung von Gefährdungen für die Systemsicherheit und zur Abschätzung der Folgen bei einem Schadenseintritt. Es sollte erwähnt werden, dass die Gefährdung sich erst aus einer Schwachstelle und zugehöriger Bedrohung ergibt. Dies bedeutet, dass eine Schwachstelle allein noch keine Gefährdung ist. Erst wenn auch eine Bedrohung für diese Schwachstelle exisitiert, wird daraus eine Gefährdung.

Vier konzeptionelle Unterschiede bei Risikoanalysen werden in [Ste02] vorgestellt. Es sei darauf hingewiesen, dass die meisten praktischen Risikoanalysen eine Mischung dieser Konzepte sind.

- **Das Szenariokonzept** bildet ausgesuchte Zustände des IT-Systems in einem hypothetischen Modell nach. Dabei wird das erstellte Modell begutachtet und verändert, um anschließend die Auswirkungen zu betrachten. Meistens werden nur Teile des Gesamtsystems in dem Modell abgebildet. Der Schwerpunkt der Analyse liegt auf Ursache-Wirkung Beziehungen. Das Szenariokonzept gehört zu den qualitativen Risikoanalysen.

- **Das Simulationskonzept** überführt das zu untersuchende IT-System in ein mathematisches Modell. Anschließend lassen sich automatisiert und

rechnerunterstützt Simulationsläufe mit dem Modell durchführen. Auch das Simulationskonzept zählt zu den qualitativen Risikoanalysen.

- **Das kardinale Bewertungskonzept** legt den Schwerpunkt auf die Risikobewertung. Schäden werden Kosten zugewiesen und die Gefahren werden statistisch bewertet. Anschließend können Erwartungswerte anhand einer Multiplikation der Faktoren beziffert werden. Mit Hilfe der errechneten Kenngrößen lassen sich Kosten-Nutzen Analysen erstellen. Das primäre Zuweisen von Kosten und Wahrscheinlichkeiten kann sich aber als sehr schwer herausstellen. Das kardinale Bewertungskonzept gehört zu den quantitativen Risikoanalysen.

- **Das ordinale Bewertungskonzept** teilt das IT-System in gedankliche Teilsysteme. Gefahren werden diesen Objekten zugeordnet, anschließend wird das Risiko mit Hilfe von Listen und Matrizen bemessen. Das Ziel ist aber nicht die Berechnung der genauen Werte, sondern eine Klassifizierung. Auch das ordinale Bewertungskonzept ist eine quantitative Risikoanalyse.

Da eine 100-prozentige Sicherheit nicht zu erzielen ist, sollte das System ein den Werten entsprechendes Sicherheitsniveau erreichen. Bei der Betrachtung nicht zu vernachlässigen sind die Wandelbarkeit des IT-Systems bei Veränderung und die Betreuung und Verwaltung der Systeme. Dabei sollten sowohl eine umfassende Abwehr von möglichen Bedrohungen als auch die schnelle Wiederherstellung des gewünschten Systemzustands nach dem Eintritt eines Schadens mit einbezogen werden. Dies ist aber schon Bestandteil des Risikomanagements.

Die Risikoanalyse ist ein Instrument des Risikomanagements.

In dem Request for Comment 2828 [Shi00] wird das Risikomanagement wie folgt definiert.

> „The process of identifying, controlling and eliminating or minimizing uncertain events that may affect system resources."

Das Risikomanagement baut sich aus folgenden Komponenten auf:

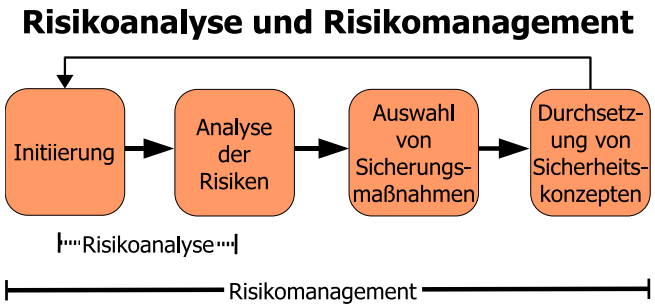

Abbildung 4.1: Risikomanagement

- **Initiierung**: In dieser Phase werden Informationen über das System gesammelt. Vermögenswerte (engl. Assets) werden festgestellt und Bedrohungen (engl. Threats) ermittelt. Dabei sollten die technische Ausrüstung (Hard-/Software), Netzwerke und Kommunikationsverbindungen, Datenbestände in jeder Form, Entwicklungsergebnisse oder strategische Richtlinien, sowie Personen und Hilfsmittel mit einbezogen werden. Schon in dieser Phase sollte eine Gewichtung der Werte nach deren Schutzgüte stattfinden. Sicherheitsziele werden optional erstellt.

- **Analyse der Risiken**: In dieser Phase findet ein Hauptteil der Risikoanalyse statt. Es werden Schwachstellen ermittelt und das Risiko beurteilt. Es können kleine Bereiche oder spezielle Eigenschaften eines Systems untersucht werden aber auch Zusammenhänge und Auswirkungen des Gesamtsystems.

- **Auswahl von Sicherungsmaßnahmen**: Die Ergebnisse der zweiten Phase werden ausgewertet. Anhand von Risikobewertungen und geeigneten Gegenmaßnahmen (engl. Countermeasures) wird ein Sicherheitskonzept erarbeitet. Es kann dabei auch vorkommen, dass bestimmte Risiken beibehalten werden, da die Einführung einer Sicherheitsmaßnahme die Kosten des zu

schützenden Wertes übersteigt. Schwachstellen können ebenso bestehen bleiben, wenn das Risiko als zu gering eingeschätzt wird.

- **Durchsetzung von Sicherheitskonzepten**: Das System wird nach dem Sicherungskatalog angepasst. Dabei kann es eventuell zu erneuten Problemen kommen, da geplante Maßnahmen sich aufgrund unvorhergesehener Umstände nicht umsetzen lassen. Das neue Sicherheitskonzept kann auch Auswirkungen auf die Organisationsstruktur haben, sowie Arbeitsabläufe verändern. Da die Vermögenswerte, die Bedrohungen und die Schwachstellen einem stetigen Wandel unterliegen, muss man das Sicherheitskonzept und die Schutzmaßnahmen regelmäßig überprüfen.

Ich komme nochmal zurück zur Risikoanalyse. In dieser Arbeit wird die Risikoanalyse hauptsächlich nach dem Szenariokonzept erstellt. Das Szenariokonzept lässt sich nochmals unterteilen in

- **Szenario-Schaden Projektion**: Notwendige Einheiten und Aspekte der eingesetzten Techniken werden zunächst erfasst und bewertet. Durch Bildung von kausalen Zusammenhängen und durch die Veränderung selektiver Elemente des Systems werden anschließend die Konsequenzen analysiert. Ausgehend von einem Szenario werden mögliche Schäden ermittelt. Es handelt sich um ein Top-Down Verfahren und kann auch als Ursache-Wirkung Prinzip beschrieben werden.

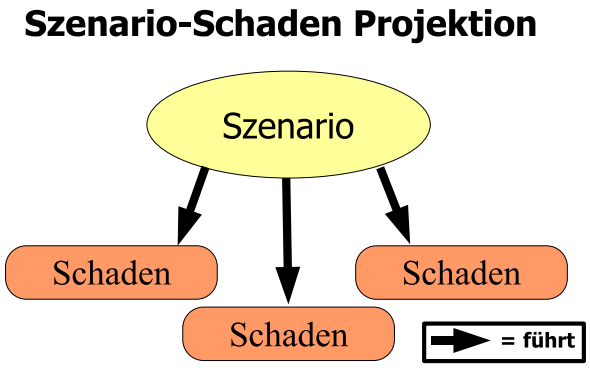

Abbildung 4.2: Szenario-Schaden Projektion

Beispiel: Man betrachtet die Auswirkungen, wenn SSL/TLS innerhalb einer Kommunikationssitzung nicht eingesetzt wird.

- **Schaden-Szenario Projektion**: Die möglichen Konsequenzen sind Ausgangspunkt der Untersuchung. Ausgehend von den Konsequenzen wird untersucht, was zu diesem Schaden führen kann. Nach der Bestimmung einer Konsequenz, wird untersucht, welche Zustände und Eigenschaften eintreffen müssen, damit es zu einem Vorfall kommt. Es wird also beschrieben, welche möglichen Szenarien zu einem festgelegten Schaden führen. Deshalb ist dies ein Bottom-Up Verfahren und kann auch als Wirkung-Ursache Prinzip bezeichnet werden.

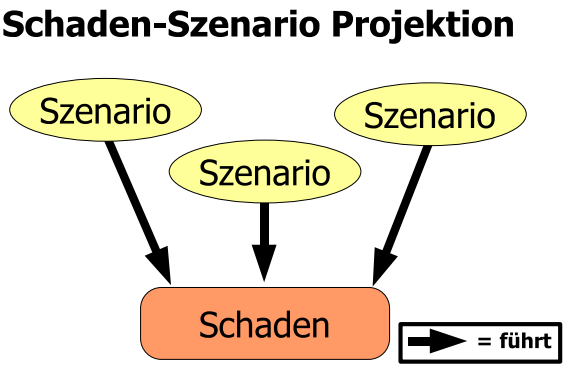

Abbildung 4.3: Schaden-Szenario Projektion

Beispiel: Welche Gegebenheiten müssen eintreten, damit es zum Verlust der Integrität von Nachrichten innerhalb einer Kommunikationssitzung kommt.

In dieser Arbeit wird vor allem die Kommunikationssicherheit als Betrachtungsobjekt für die Risikoanalyse untersucht.

Kommunikationssicherheit

Die Kommunikationssicherheit befasst sich mit der Sicherheitsgüte beim Transfer von Nachrichten über einen Kommunikationskanal zwischen Knoten in einem Netzwerk. Zusätzlich umfasst die Kommunikationssicherheit den Schutzgrad von Netzwerkknoten vor unerlaubter/ungültiger und unbefugter Benutzung. Ziel ist die Integrität, Authentizität und Vertraulichkeit der Daten zu gewährleisten und die Verfügbarkeit der IT-Systeme sowie die Sicherung der Daten.

Unter dem Begriff Risikoanalyse soll in dieser Arbeit die Untersuchung und Bewertung von Gefährdungen der Informationsverarbeitung sowie ihrer Ursachen und Konsequenzen auf die Kommunikationssicherheit verstanden werden. Dies bedeutet, es werden Voraussetzungen für die Kommunikationsteilnehmer bestimmt und anschließend bestimmte Aspekte betrachtet und analysiert. Diese Art der Risikoanalyse untersucht somit nicht das ganze System, sondern betrachtet nur Teilaspekte unter bestimmten Voraussetzungen.

4.1.1 Ausgewählte Aspekte der Kommunikationssicherheit

Der Angreifer

Ein Angreifer versucht einen Vorfall zu generieren, mit dem Ziel eine Sicherheitsverletzung zu erreichen. Dabei lässt sich unterscheiden, ob es sich bei dem Angreifer um einem Teilnehmer des Systems handelt oder um einen Aussenstehenden. In unserem Fall ergibt sich also eine Einteilung in

- **Interner Angreifer**: Ein Benutzer, **SP** oder **ASP** ist ein anerkannter und legitimer Teilnehmer innerhalb des Circle-of-Trust, versucht aber seine Rechte und Möglichkeiten auszudehnen mit Hilfe eines Angriffs auf das System. Vielleicht täuscht dieser Angreifer seine Rolle den anderen Teilnehmern vor und nimmt eine andere Rolle an, als er eigentlich innerhalb des Systems inne hat, z.B. ein **SP** gibt sich als ein **ASP** aus.

- **Externer Angreifer**: Ein Benutzer, **SP** oder **ASP** täuscht dem Circle-of-Trust vor, ein legitimer Teilnehmer zu sein, d.h., eine externe Instanz versucht an der Kommunikation teilzunehmen oder beobachtet nur die legitimen Teilnehmer. In der Regel kann davon ausgegangen werden, dass die verfügbare Informationsdichte variiert, den ein interner Angreifer wird meist über mehr Informationen zum System verfügen als ein externer. Dies ist jedoch nicht zwingend der Fall. Ein weiterer Unterschied ergibt sich durch den Zustand, in dem sich der Angreifer innerhalb des Systems befindet. Einige Angriffe erfordern, dass der Angreifer ein akzeptierter Teilnehmer des Systems ist. Diesen Zustand muss ein externer Angreifer somit zunächst erreichen, um anschließend einen solchen Angriff auszuführen.

Ein weiteres Unterscheidungsmerkmal bezieht sich auf die bewusste Ausführung des Angriffs durch den Angreifer. Zu fragen ist, ob das angreifende System den Angriff mit der Intention des Besitzers ausführt. Durch den Einsatz von Malware kann es z.B. vorkommen, dass der Benutzer eines Systems zwar eine bestimmte Vorstellung davon hat, was sein System ausführen sollte, das System aber in Wirklichkeit etwas anderes durchführt.

Die nachfolgende Auflistung nach [Kas98] soll einen Überblick von Motivationen für Angreifer auflisten.

- Hacker - persönliche Herausforderung, Sendungsbewusstsein, Ehrenkodex.
- Berufsverbrecher - eigenen finanziellen Vorteil erzielen.
- Vandalen - persönliche Unzufriedenheit, Rachsucht, Machtstreben.
- Spione - eigenes Land oder Organisation politisch oder wirtschaftlich voranbringen.
- Terroristen - politischen Einfluss ausüben, Schaden verursachen.
- Firmenbanditen - finanzieller oder strategischer Vorteil des eigenen Unternehmens.

Ein gemeinsamer Anreiz könnte der wahrscheinlich zu erwartende Informationsgewinn sein.

Zusammenschluss von Angreifern: In unserem Fall können sich auch Angreifergruppen formieren. Verschiedene Instanzen unseres Systems arbeiten zusammen, um einen Angriff auszuführen. Folgende Kommbinationen von Angreifern können gebildet werden:

- **Ein Benutzer und ein SP**. Ein Beispiel wäre, dass ein **SP** und ein Benutzer zusammenarbeiten, um anderen **SP**s zu schaden oder den **ASP** anzugreifen. Zu diesem Zweck könnte der Benutzer auch bewusst vom **SP** angelegt worden sein.

- **Ein Benutzer und der ASP**. Diese Konstellation könnte benutzt werden, um einen **SP** mit einem **ASP** anzugreifen, der überhaupt kein gültiger Teilnehmer innerhalb des Circle-of-Trust ist.

- **Der ASP und ein SP**. Zunächst scheint diese Kombination keinen Sinn zu machen, aber es können sich auch Gefahren für den Benutzer ergeben, wenn der **ASP** und ein **SP** mehr Informationen austauschen als dem Benutzer bekannt sind.

- **Mehrere Benutzer zusammen**. Eventuell lassen sich vertrauliche Informationen bei den **SP**s und den **ASP** durch das Zusammenarbeiten von mehreren Benutzern aufdecken.

- **Mehrere SPs zusammen**. Arbeiten verschiedene **SP**s zusammen, lässt sich vielleicht ein **ASP** kompromittieren, oder es könnten vertrauliche Benutzerdaten ausgelesen werden. Auch ungewünschte Benutzeraktionen könnten sich eventuell ergeben.

- **Mehrere ASPs zusammen**. Auch hier scheint das Zusammenarbeiten von **ASP**s die Gefahr nicht zu erhöhen, da ein **ASP** schon an oberste Position innerhalb der Vertrauenskette steht. Betrachtet man aber Verbunde von **ASP**s innerhalb einer Authentication Infrastruktur, so bieten sich wieder Angriffsziele an.

Es lassen sich natürlich noch weitere Kombinationen von Angreifergruppen finden. Man erkennt jedoch schon an diesen Möglichkeiten die Komplexität einer umfassenden Sicherheitsuntersuchung.

Der Prozess eines Angriffs. Zunächst betrachten wir, welchen Gefahren die Systeme ausgesetzt sein können und was überhaupt ein Angriff ist. Computersysteme und Netzwerke können Aktionen ausgesetzt sein, die das Ziel verfolgen, eine nicht autorisierte oder unerwünschte Reaktion auszuführen. Ein Angriff besteht laut Howard [HL98, S.12] aus mehreren Elementen.

Ein Angreifer durchläuft also bei einem Angriff folgende Schritte. Mit einem oder mehreren Werkzeugen führt der Angreifer eine Aktion aus. Damit es zu

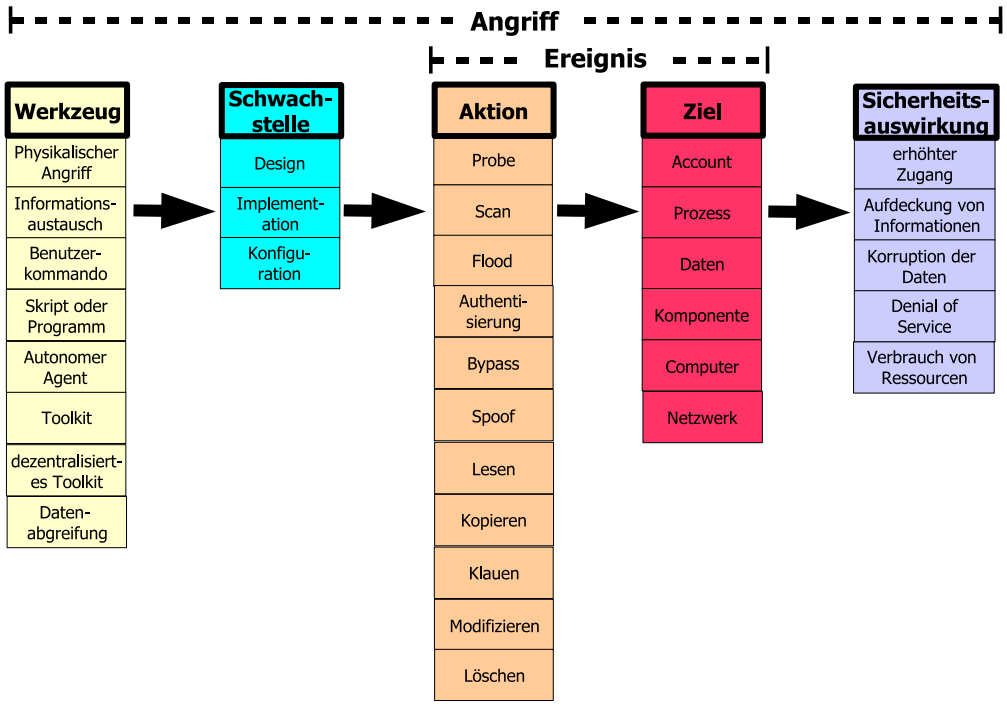

Abbildung 4.4: Der Angriffsprozess

einer Gefährdung für das System kommt, muss vor der Aktion des Angreifers eine Schwachstelle innerhalb des Systems existieren. Trifft die Aktion auf die Schwachstelle, so wird dies unter Umständen Auswirkungen auf die Sicherheit nach sich ziehen.

Um die Sicherheit eines Systems einzuschätzen und anschließende Verbesserungen auswählen zu können, bedienen wir uns folgender Einteilung:

- **Angriff auf die Sicherheit**: Jegliche Aktion, die die Sicherheit von Arbeitsabläufen oder Informationen einer Organisation gefährdet.
- **Sicherheitsmechanismus**: Ein Mechanismus, der entworfen wurde, um Angriffe zu erkennen, zu verhindern oder zu beheben.
- **Sicherheitsdienst**: Ein Sicherheitsdienst verbessert die Sicherheit von Datenverarbeitungssystem und schützt die Informationen von Organisationen.

Der Unterschied zwischen Sicherheitsmechanismus und Sicherheitsdienst besteht darin, dass ein Sicherheitsdienst eine konkrete Implementation oder Maßnahme

beschreibt, die sich aus einem oder mehreren Sicherheitsmechanismen zusammensetzt. Ein Beispiel: Einem Angriff auf die Integrität der Daten kann man mit der Kryptographie entgegentreten. Somit bildet ein Verfahren aus der Kryptographie dann einen Sicherheitsmechanismus. Durch die Auswahl geeigneter Sicherheitsmechanismen erstellt man dann den Sicherheitsdienst.

Zu betrachtende Sicherheitsziele: Es existieren Gefahren für die folgenden vier Sicherheitsziele: Vertraulichkeit, Verfügbarkeit, Integrität und Authentikation. Es wird analysiert, ob Szenarien möglich sind, die zu einer Beeinträchtigung der Verfügbarkeit, der Integrität, der Vertraulichkeit oder der Authentizität führen. Eine symbolische Zuordnung von Angriffen zu einer dieser vier Konsequenzen zeigt folgendes Bild.

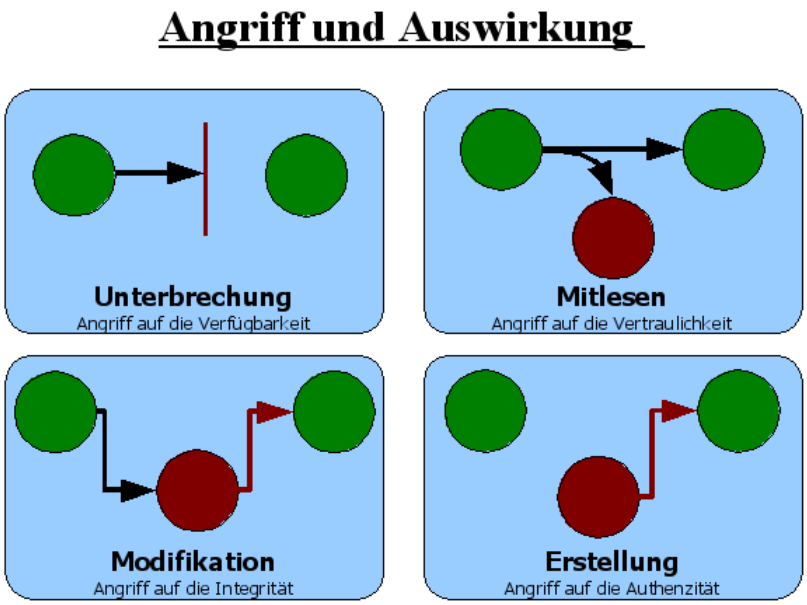

Abbildung 4.5: Angriffe auf die Verfügbarkeit, Integrität, Vertraulichkeit und Authentikation.

- Unterbrechen der Kommunikationsverbindung (Interruption)[aktiv]: Es findet ein Angriff auf die Verfügbarkeit statt. Bekanntestes Beispiel ist der DoS-Angriff[2]. Eventuell lässt sich eine Unterbrechung eines bestimmten

[2]DoS steht für Denial of Service, also für den Ausfall eines Dienstes.

Kommunikationskanals ausnutzen, um leichter einen Angriff auf dem Ausweichkanal auszuführen. Dies könnte bei SSLv2 der Fall sein, wenn SSLv3 nicht erlaubt wird.

- Mitlesen der Kommunikation (Interception)[passiv]: Dies ist ein klassischer Angriff auf die Vertraulichkeit. Dabei kann man unterscheiden, ob auch der Nachrichteninhalt mitgelesen und ausgewertet wird oder ob nur die Verkehrswege (Traffic Analysis) betrachtet werden.

- Erstellung von Nachrichten (Fabrication)[aktiv]: Die Authentikation gilt als Angriffsziel. Ein Angreifer erstellt Nachrichtenpakete und schleust diese erfolgreich in den Kommunikationskanal. Dazu gehört auch die Replay Attacke. Der Angreifer speist schon übertragene Nachrichtenpakete erneut in den Kommunikationskanal. In der digitalen Welt kann nicht mehr zwischen einer Kopie und dem Orginal eines Dokuments unterschieden werden.

- Modifikationen der Nachrichten (Modification)[aktiv]: Dieser Angriff stellt einen Verlust der Integrität dar. Ein Angreifer fängt Nachrichtenpakete aus dem Kommunikationskanal ab und verändert deren Inhalt. Anschließend wird das veränderte Nachrichtenpaket an den Empfänger geschickt. Bekanntes Beispiel ist der Man in the Middle Angriff.

In der Praxis lassen sich die eben genannten vier Punkte nicht gut trennen, da sie nicht orthogonal zueinander sind. Häufig hat eine Einschränkung oder der Verlust eines dieser Sicherheitsziele Auswirkungen auf die anderen Ziele. Die folgenden Szenarien schließen häufig mehrere Sicherheitsziele ein. Ich habe versucht, sie ihrem Risikoschwerpunkt nach diesen vier Kategorien zuzuordnen. Einige der folgenden Szenarien sind schon aus [KR00], [Lib03b] und [PW03] bekannt, andere sind in dieser Arbeit neu hinzugekommen.

4.2 Angriff auf die Verfügbarkeit

Die Verfügbarkeit spielt eine nicht zu unterschätzende Rolle bei den vorgestellten Systemen. Damit solche Systeme nutzbar sind und eine weite Verbreitung finden, muss von einer hohen Verfügbarkeit ausgegangen werden. Die beiden Infrastrukturen sollten auf ein Minimum an Ausfallzeit hin optimiert sein. Eine Einschränkung der Verfügbarkeit oder sogar der totale Verlust einer Komponente kann gravierende Auswirkung auf das Gesamtsystem haben.

Zunächst kann man hinsichtlich der Infrastruktur eine Unterscheidung der Systeme vornehmen. Microsoft Passport stellt einen Vertreter der zentralisierten Single Sign-On Lösungen dar. Es hat einen zentralen **ASP** und bietet somit ein interessantes Angriffsziel innerhalb des Gesamtsystems. Der Ausfall des **ASP** würde das ganze System lahmlegen. Somit ist der **ASP** ein bevorzugtes Ziel für Denial of Service Attacken.

Zwar kann der **ASP** aus Redundanzgründen mehrfach im Netz vorkommen, er bleibt aber weiterhin unter der Verwaltung von Microsoft. Microsoft Passport wird dabei wohl ohnehin nicht mit einem einzigen Rechner als **ASP** arbeiten, auch wenn der Cluster unter einer festen URL adressiert wird. Jeglicher Dienst im Internet, der eine bestimmte Anfrageschwelle zu verzeichnen hat, muss geclustert oder mit Proxies betrieben werden. Proxies dürften nicht sinnvoll einsetzbar sein, da jegliche Kommunikation mit dem Proxy aus Sicherheitsgründen auch eine Kommunikation zwischen Proxy und **ASP** erfordert. Der Status des Benutzers ist eine nicht zu cachende Information.

Nutzen die **SP** statische Anfrage-URLs, um den **ASP** zu kontaktieren oder den Benutzer dorthin weiterzuleiten, schützt eine Redundanz nicht zwangsläufig. Fällt jetzt nämlich für diese URL der DNS-Dienst aus oder existiert kein Kommunikationskanal, z.B. wegen Routingfehlern, dann wird ein direkt im gleichen Netz erreichbarer redundanter Ausweich-**ASP** wohl nicht zu nutzen sein. Es ist daher davon auszugehen, dass die redundanten **ASP**s in unterschiedlichen Netzen und geographisch verteilt betrieben werden. Gerade aber dieser notwendige Ausfallschutz birgt neue Gefahren. Die Datenbank mit den geheimen Schlüsseln und den Benutzerprofildaten muss repliziert und ständig synchron gehalten werden. Die geheimen Schlüssel beinhalten sowohl die Kommunikationsschlüssel für den

Austausch mit den **SP**s, als auch die nur vom **ASP** verwendeten. Für den Informationsaustausch dieser Daten müssen erneut Kommunikationskanäle zwischen diesen **ASP**s eingerichtet werden. Zusätzlich steigt der Aufwand der Betreuung dieser **ASP**s und die Komplexität der Verwaltung. Microsoft macht zu der Redundanz und der damit anfallenden Notwendigkeit von Sicherheitskonzepten keine Angaben. Hinzu kommt, dass Microsoft auch keine Aussagen über die Art und Weise macht, wie die Daten abgespeichert werden und wie das Backupkonzept aussieht.

Liberty Alliance sieht im Aufbau eines Circle-of-Trust schon die Möglichkeit unterschiedlicher **ASP**s vor. Jeder einzelne **ASP** muss die gleichen Probleme betrachten wie schon bei Microsoft Passport erläutert. Zusätzlich kommt noch eine erhöhte Komplexität durch die Nutzung von unterschiedlichen **ASP**s sowohl beim **SP** als auch beim Benutzer hinzu. Wie die technischen Einrichtungen aufgebaut sein müssen, wird nicht innerhalb der Liberty Alliance Spezifikation vorgeschrieben, geschweige denn erwähnt. Wie bei Microsoft Passport geht die Liberty Alliance Spezifikation auch nicht auf die Datenspeicherung und auf Datenlagerung, sowie Datensicherung ein. Es werden keine Mindeststandards verlangt. Es bleibt somit jedem Circle-of-Trust überlassen, ob überhaupt und wenn, welche Anforderungen an die Datenhaltung und Datensicherung auf technischer Ebene zu stellen und zu realisieren sind.

Was passiert aber bei den beiden Systemen, sollte es zu einem Ausfall der **ASP**s kommen? Beide Systeme verfolgen in so einem Fall ein differierendes Konzept. Microsoft Passport sieht die Möglichkeit vor, den *Passport Manager* in einem *Standalone* Modus zu betreiben, so dass dieser nicht mehr den **ASP** kontaktiert, sondern jegliche Anfrage als authentisiert ansieht [Mic04j]. Dies wird für den weiteren Betrieb notwendig, da der **SP** keine eigene Authentikation und Benutzerverwaltung vorhalten muss. Der Verlust des **ASP** würde somit den **SP** dienstunfähig machen. Bei kommerziellen Diensten ist dies aber nicht hinnehmbar. Welche Auswirkung beim Betreiben des *Passport Managers* in diesem Modus beim **SP** hat, sollte jedem klar sein: Es findet keine vernünftige Authentisierung statt.

Liberty Alliance baut von vornherein auf eine eigene Benutzerverwaltung beim **SP**. Da ein Benutzer ohnehin ein separates Konto beim **SP** einrichten muss und

anschließend dann dieses Konto federiert/verknüpft, hat der Benutzer bei einem Ausfall des **ASP** weiterhin die Möglichkeit, die Dienste vom **SP** wahrzunehmen. Er braucht sich bloß mit seinem bei diesem **SP** angelegten Benutzeraccount direkt beim **SP** anzumelden. Nur die komfortable Anwendung eines Single Sign-On würde verloren gehen. Kein **SP** müsste bei einer weiteren Diensteanbietung einen Verlust des Sicherheitsniveaus in Kauf nehmen. Man sollte aber erwähnen, dass durch die Teilnahme des **SP** an einem Liberty Alliance Netzwerk, die Zugangsmöglichkeiten gestiegen sind und somit auch die Gefahr eines Missbrauchs. Neben seinem eigenen Benutzerzugangsverfahren und seiner Benutzerverwaltung beim **SP** wird zusätzlich noch das Trustmodel von Liberty Alliance genutzt. Da die Benutzerdaten für eine direkte Anmeldung beim **SP** wohl selten benutzt werden und somit diese Daten irgendwo abgelegt sein müssen, um bei Bedarf darauf zurückgreifen zu können, sollte dieses nicht vom **SP** vernachlässigt werden. Diese Daten sollten vom Benutzer sicher abgespeichert sein und besonders geschützt werden. Ansonsten hätte man von Single Sign-On bei Liberty Alliance nicht viel gewonnen. Eine Motivation der Spezifikation war ja gerade, die Unsicherheiten bei der Verwaltung von vielen Passwortsystemen zu beheben.

Eine Einschränkung der Verfügbarkeit kann eventuell durch das eingesetzte Kommunikationsprotokoll ausgelöst werden. Je nachdem welche Felder innerhalb einer Nachricht gesetzt sind, oder das Fehlen eines Sicherheitsfaktors kann einen erhöhten Ressourcenverbrauch auf dem Rechner auslösen. Ein Beispiel sei der Missbrauch mit einer gefälschten `AuthnRequest` Nachricht.

4.2.1 Erhöhter Ressourcenverbrauch beim ASP mit gefälschten `AuthnRequest` Nachrichten

Sendet ein Betrüger eine gefälschte `AuthnRequest` Nachricht an einen **ASP**, so belegt dieser Auftrag beim **ASP** Ressourcen. Dabei braucht der Betrüger nicht erst irgendwelche Vorbereitungen zu treffen. Er schickt von seinem mit dem Internet verbundenen Rechner eine `AuthnRequest` Nachricht an den **ASP**. Bei bestimmten Feldern innerhalb der Nachricht setzt er einfach bestimmte Werte von einem teilnehmenden **SP** ein. An diese Information gelangt er ganz leicht,

indem er einfach bei einem legitimen Gebrauch des Systems den Netzwerkverkehr seines Webbrowsers mitliest. Wünscht der Anfragende dann noch das Artefaktverfahren, um an das Authentication Assertion zu gelagen, so werden weitere Ressourcen beim **ASP** belegt. Der **ASP** muss die Anfrage bearbeiten und somit Daten aus dem System anfordern und Rechenzeit verbrauchen. Außerdem muss er die Authentication Assertion erstellen und zumindest signieren. Eventuell fallen noch weitere rechenintensive oder I/O-lastige Aufgaben an. Der **ASP** kann erst nach einem Timeout, wenn nämlich das Assertion nicht angefordert wird, erkennen, dass die Anfrage nicht gültig ist. Setzt der Angreifer jetzt innerhalb kurzer Zeit ganz viele Anfragen ab, kommt es beim **ASP** zu einer Beeinträchtigung der Verfügbarkeit und eventuell sogar zu deren Verlust.

Ein **ASP** könnte eine solche gefälschte Anfrage leicht bemerken, wenn er die Anfragen nur über HTTPS erlaubt mit einem gültigen Clientzertifikat oder auch nur eine gültige Signatur für das `AuthnRequest` fordert. Diese beiden Maßnahmen würden zwar auch bei jeder Anfrage Ressourcen verbrauchen, es würden aber weniger Ressourcen verbraucht, und der **ASP** kann wesentlich schneller die Fälschung erkennen und somit Ressourcen schneller wieder freigeben.

4.2.2 Erhöhter Ressourcenverbrauch beim SP mit gefälschten `AuthnResponse` Nachrichten

Findet eine Anfrage bei einem **ASP** ohne Verschlüsselung statt und ein Angreifer kann somit den Nachrichteninhalt mitlesen, dann kann er das für einen Angriff beim anfragenden **SP** missbrauchen. In diesem Fall würde eine Signierung der `AuthnRequest` Nachricht keine Besserung bringen. Der Angreifer kann in der `AuthnRequest` Nachricht das $RequestID$ auslesen und diesen Wert in einer gefälschten Antwort einsetzen. Die vom Angreifer an den **SP** verschickte `AuthnResponse` Nachricht würde in dem $InResponseTo$ Feld den ausgelesen Wert enthalten. Der **SP** würde von einer legitimen Antwort des **ASP** ausgehen und das Assertion validieren. Dieser Schritt würde zwar sehr wahrscheinlich einen Fehler erzeugen, aber bis dahin eine erhöhte Rechenzeit benötigen.

Bei beiden Beispielen besteht der Fehler in der Annahme von unautorisierten

Nachrichten und somit die Gefahr, dass die Verfügbarkeit eingeschränkt wird. Bei Microsoft Passport kann in dieser Arbeit keine Aussage über ähnliche Probleme getroffen werden, da Microsoft nicht den genauen Inhalt der Nachrichten angibt. Es bliebe zwar die Möglichkeit, durch Sniffing oder Reverse Engeniering Methoden diese herauszubekommen, aber selbst das wäre keine Gewähr für einen Erfolg, z.B. bei Verschlüsselung von Nachrichtenfeldern.

Gerade weil die Verfügbarkeit bei kommerziellen Interessen ein wichtiges Ziel darstellt, soll noch kurz erwähnt werden, dass der Einsatz von HTTPS die Sicherheit einerseits erhöhen kann. Anderseits zeigt die Realität, dass gerade die somit hinzugekommene Komplexität von HTTPS eine neue Gefahr bilden kann. Bei allen großen Webserversystemen gab es im letzten Jahr mehrere Schwachstellen[3] in der HTTPS-Verarbeitung die zu einem Denial of Service führten.

4.3 Angriffe auf die Vertraulichkeit

In diesem Abschnitt betrachten wir die Vertraulichkeit der ausgetauschten Informationen über den Kommunikationskanal. Vertraulichkeit bezieht sich dabei auf den Umstand, dass nur legitime Teilnehmer bestimmte Informationen einsehen können. Alle anderen sollten so gut wie keine Informationen oder zumindest nur ein Minimum davon erhalten können. Eine Vertraulichkeit bei einer Kommunikation wird in der Regel über eine Verschlüsselung realisiert. Das Thema des Vertrauens zwischen den Teilnehmern wird später in dieser Arbeit unter 4.7.2 betrachtet und ist somit nicht Teil dieses Abschnitts.

4.3.1 Senden der Nachrichten im Klartext

Sowohl Microsoft Passport als auch die Liberty Alliance Spezifikation verlangen nicht, dass sämtliche Kommunikation über einen verschlüsselten Kanal erfolgen muss. Somit hat ein Angreifer die Möglichkeit anhand eines passiven Angriffs zum Zwecke des Mitlesens der Nachrichtenpakete, sensible Informationen zu erlangen.

[3] Bei der Mailingliste „Bugtraq“ finden sich mehrere Schwachstellen aus dem Jahre 2004.

Wie zuvor beschrieben, kann z.B. das Mitlesen der $RequestID$ zu einem Angriff auf die Verfügbarkeit genutzt werden. Aber auch andere Informationsfelder können für einen Angreifer interessant sein und als Grundlage für weitere Angriffe dienen. Die Möglichkeit, Ausschnitte der Kommunikation oder sogar den kompletten Ablauf mitzulesen, ist nicht nur für Angriffe gefährlich. Gerade die Gewährleistung, bestimmte Informationen nur dem gewünschten Empfänger zugänglich zu machen, sollten Single Sign-On Lösungen umsetzen. Da ansonsten sensible Informationen mitgelesen werden können. Zudem können die Daten für ein Usertracking missbraucht werden. Der Unterschied zum herkömmlichen Ablauf ohne Single Sign-On ist, dass wenn ein Angreifer die Kommunikation nahe bei dem **ASP** mitlesen kann, er Daten über wesentlich mehr Benutzer erhält. Einen Teil dieser Informationen könnte ein Angreifer auch bei einem Benutzer ohne Single Sign-On mitlesen, wenn er den Netzverkehr nah an dem Benutzerrechner abgreifen würde. Beim Abgreifen der Daten in der Nähe vom **ASP** erhält der Angreifer aber gleich die Kommunikationsinformationen vieler Benutzer, egal wo diese Benutzer sich geographisch aufhalten und welche **SP** sie nutzen.

Interessante Felder wären bei Passport zunächst die $PUID$, anhand derer der Angreifer eindeutig den Benutzer bei jeder Nachricht identifizieren kann. Bei Liberty Alliance hätte er das $NameIdentifier$ und das $ProviderID$ Feld, um den Benutzer zu identifizieren. Betrachtet er zusätzlich die IP des Anfragenden beim **ASP**, also wegen des Redirectverfahrens die IP des Benutzers, so kann er eine Liste aufstellen, in der die unterschiedlichen $NameIdentifier$ des Benutzers mit zugehörigem **SP** stehen. Beobachtet der Angreifer das Netz lang genug, so kann er mit dieser Liste, einen Benutzer leicht mitsamt den besuchten **SP**s identifizieren und tracken. Das $AuthnContextClassRef$ kann zusätzlich benutzt werden, um eventuell die Art der angeforderten Dienstleistung beim **SP** zu identifizieren. Nutzt der **SP** nämlich unterschiedliche Authentikationsklassen, je nach angefordertem Dienst, so kann diese Information auch eingesehen werden.

Bei Single Sign-On Systemen mit zusätzlicher Benutzerprofilverwaltung besteht bei fehlender Verschlüsselung der Kommunikation zudem die Gefahr, die Profildaten Unberechtigten zugänglich zu machen. Gerade diese Informationen

stellen sehr sensible Daten dar und ein Mitlesen dürfte nicht im Interesse des Benutzers sein. Microsoft Passport verschlüsselt zwar die Profildaten bei der Übersendung an den **SP**, es steht dem **SP** aber frei, diese Informationen selbstständig nochmals in einem Cookie beim Benutzer abzuspeichern. Zu diesem Vorgang existieren seitens Microsoft Passport keine Vorgaben. Die gleiche Problematik existiert auch bei Liberty Alliance, wenn dort Profildaten übertragen werden. Doch zu diesem Vorgang existieren noch keine Dokumente und deswegen können dazu keine genaue Aussage getroffen werden.

Bestimmte Felder innerhalb der ausgetauschten Nachrichten ermöglichen das Analysieren von Benutzerverhalten und somit auch die Zuordnung der mitgelesenen Nachrichten zu bestimmten Benutzersessions, da an mehreren Stellen bestimmte Zeichenketten aus einer Anfrage in der Anwortnachricht wiederholt werden müssen. Zu nennen wäre bei Liberty Alliance z.B. das $RequestID$ mit anschließendem $InResponseTo$ Feld. Ein anderes Nachrichtenfeld, welches eine Zuordnung von Nachrichten zu Benutzersitzungen zulässt, ist das $RelayState$ bei Liberty Alliance. Kann ein Angreifer die Kommunikation mitlesen, so hat er anhand dieses Feldes die Möglichkeit, Anfrage und Antwort einzelnen Benutzersessions zuzuordnen.

Ein anderes Problem ergibt sich, wenn ein Angreifer bei Liberty Alliance ein Artefakt bei der Übertragung zwischen **ASP** und **SP** mitlesen kann. Genügt dem **ASP** nämlich allein das Vorweisen des Artefakts für eine Auslieferung der Authentication Assertion, so könnte ein Angreifer dieses mitgelesene Artefakt nutzen, um an das Authentication Assertion zu gelangen. Er müsste nur schneller als der **SP** das Artefakt dem **ASP** präsentieren. Aus diesem Grund sollte jeder **ASP** buchführen, an wen er eine Artefakt ausgeliefert hat und ob die anschließende Anfrage für das Authentication Assertion auch wirklich von diesem **SP** kommt.

Die oben genannten Gefahren sollten aufzeigen, dass eine Signierung der Daten nicht ausreicht und gerade im Interesse des Benutzers zusätzlich verschlüsselt übertragen werden sollten. Kann die gesamte Kommunikation nicht mittels HTTPS erfolgen, so sollten zumindest die wichtigen Kommunikationsschritte ver-

schlüsselt erfolgen. Das erfordert zwar einen höhren Ressourcenverbrauch, dient aber dem Datenschutz des Benutzers sowie der Sicherheit des Systems. Bei Liberty Alliance sollten die `AuthnRequest` und die `AuthnResponse` Nachrichten über einen verschlüsselten Kanal übertragen werden. Bei Microsoft Passport sind die zu übertragenden Daten zwar schon verschlüsselt, dennoch sollte zusätzlich noch HTTPS für die Verschlüsselung eingesetzt werden. Die Cookiedaten sind nämlich symmetrisch mit immer dem gleichen Schlüssel verschlüsselt und damit bei wiederkehrenden Inhalten leicht für einen Beobachter zu identifizieren. Zudem sollte ein **SP** die enthaltenen Benutzerinformationen nicht wiederholt unverschlüsselt als Cookie beim Benutzer ablegen. Sowohl bei Liberty Alliance als auch bei Microsoft Passport sollte unbedingt die Authentikationskommunikation zwischen Benutzer und **ASP** über einen verschlüsselten Kanal erfolgen, insbesondere der Loginprozess beim **ASP**. Wie später bei in Kapitel 4.4.1 gezeigt wird, sollte aber jegliche Kommunikation bei beiden Systemen über HTTPS erfolgen.

4.3.2 Der Cache beim Benutzerbrowser

Obwohl bisher nur die Netzwerkkommunikation betrachtet wurde, soll noch auf ein Problem mit dem Webbrowser Cache hingewiesen werden. Durch die verwendete Technik nimmt der Benutzerwebbrowser einen wichtigen Stellenwert in den Sicherheitsbetrachtungen ein. Webbrowser speichern die getätigte Kommunikation inklusive übertragener Daten in ihrem Cache. Dies beinhaltet nicht nur die Seiteninformationen, sondern auch die verwendeten URLs. Diese Informationen werden unabhängig, ob diese über einer HTTP- oder HTTPS-Verbindung übertragen wurden, im Klartext auf dem Benutzerrechner abgespeichert. Erhält ein Angreifer Zugriff auf den Cache beim Benutzer, so kann er Informationen erhalten, die trotz verschlüsselter Übertragung nun für jeden lesbar sind. Dieser Zugriff auf den Cache kann lokal passieren, wenn jemand Zugang zu dem Benutzerrechner hat oder auch über das Netz mittels Schwachstellenausnutzung beim Benutzerbrowser.

Aber nicht nur über den Cache des Webbrowsers könnten Informationen veröffentlicht werden. Moderne Webbrowser bieten die Möglichkeit, bei fehlerhaften Anfragen, die angegebene URL automatisch an einen Suchdienst im Internet zu

übermitteln oder an einen sogenannten „What's related“ Dienst zu übersenden. Zusätzlich könnten benutzte URLs auf nicht legitimen Servern ersichtlich werden, wenn die zuvor besuchte URL in dem Referrer Feld bei HTTP an dem Server mit übermittelt wird. Bei beiden Systemen wird in der URL nicht nur ein Kommunikationsendpunkt beschrieben, sondern auch die Single Sign-On sowie Profildaten.

4.4 Angriff auf die Integrität

In diesem Kapitel betrachten wir, ob Nachrichten während der Übertragung verändert werden können, ohne, dass der Empfänger dieses erkennen kann. Dabei wird in diesem Abschnitt hauptsächlich Liberty Alliance betrachtet, da die Struktur der ausgetauschten Nachrichten bei Microsoft Passport nicht in der Dokumentation von Microsoft aufgeschlüsselt wird.

4.4.1 Austauschen der Kommunikationsendpunkte

Sowohl bei Microsoft Passport als auch bei Liberty Alliance wird für initiale Kommunikation mit dem **SP** kein HTTPS vorgeschrieben. Dies bedeutet, dass der erste Verbindungsaufbau vom Benutzer zu einem **SP** in der Regel über HTTP erfolgt. Bei der Benutzung von HTTP ist keine Integrität gegeben. Das Verändern des Nachrichteninhalts bei der Übertragung kann nicht vom Empfänger erkannt werden.

Gehen wir davon aus, dass der Angreifer die Kommunikation an irgendeiner Stelle zwischen Benutzer und **SP** mitlesen und verändern kann. Ein Angreifer könnte somit die Anfrage-URL des Benutzers ändern oder die Antwort vom **SP** modifizieren. Bei der Antwortnachricht könnte er den Redirect Endpunkt austauschen, um den Benutzer zu einem anderen **SP** umzuleiten. Er würde den Webseitenaufbau und die Struktur des ursprünglichen **SP** auf dem umgeleiteten Rechner nachbilden, damit der Benutzer keinen Verdacht schöpft. Durch die Nutzung von HTTP hat ein aufmerksamer Benutzer nur die „Locationbar“ oder das „Statusfeld“ im Webbrowser zur Verfügung, um Informationen über den Kommunikationsendpunkt zu erhalten. Diese Information ist aber nicht beson-

ders vertrauenswürdig. Immer wieder werden Methoden gefunden, um fehlerhafte Informationen an diesen beiden Stellen anzuzeigen. Eine Auswahl von Modifikationsverfahren, um diese Felder mit anderen Werten zu befüllen, sind Java Script Verfahren, überlange URLs, Überdecken mit Bildern oder das sogenannte Nullbyte Verfahren[4].

Dabei sollte nicht nur davon ausgegangen werden, dass der vom Benutzer durch Modifikation des Redirect besuchte **SP** ein außenstehender Teilnehmer des Single Sign-On Systemes ist und lediglich dem Angreifer dazu dienen soll, um an Informationen des Benutzers zu gelangen. Es wäre auch möglich, den Benutzer zu einem **SP** eines teilnehmenden Konkurrenten innerhalb des Single Sign-On Systems zu leiten. Da dieser Konkurrent ein legitimer **SP** innerhalb des Systems ist und bei sehr ähnlicher Seitengestaltung die Gefahr bestünde, dass der Benutzer die Dienste eines anderen **SP** benutzen würde, als eigentlich von ihm gewünscht. Dies wäre nicht nur eine Gefahr für den Benutzer, sondern auch für teilnehmende **SP**s, die somit nicht wie gewünscht vom Benutzer angesprochen würden und eventuell einen finanziellen Schaden erleiden würden.

Für einen Angreifer könnte es aber auch interessant sein, den zu kontaktierenden **ASP** innerhalb der Redirect-URL auszutauschen. Kann der Angreifer den Benutzer zu einem gefälschten **ASP** leiten, den der Angreifer unter seiner Kontrolle hat, so kann er die Anmeldedaten des Benutzers für das Single Sign-On System erfahren. Je nach Vorsicht des Benutzers müsste er eventuell einigen Aufwand treiben um dem Benutzer erfolgreich den **ASP** vorzutäuschen. Wiederum ist dies bei beiden Systemen möglich, da beide Systeme mit Redirects arbeiten und die URL des zu kontaktierenden **ASP** dem Webbrowser des Benutzers innerhalb der Nachrichten übergeben.

Bei beiden Systeme besteht die Möglichkeit, bei der Umleitung des Benutzers vom **SP** zum **ASP** die URL des Redirects auszutauschen. Bei Microsoft Passport antwortet der **SP** auf eine Anfrage eines noch nicht angemeldeten Benutzers mit einem URL-Redirect auf den **ASP**. Angehängt an die URL des **ASP** wird als Querystring eine ID und Antwort-URL übergeben. Alle drei Felder sind bei fehlendem Integritätsschutz beliebig austauschbar.

[4]Eine Auflistung von Verfahren findet sich im Anhang.

Bei Liberty Alliance wird im Redirect neben der URL des **ASP**s auch die `AuthnRequest` Nachricht im Querystring angegeben. Innerhalb dieser Nachricht können bei fehlender Signierung auch der anfragende **SP**, die Anwort-URL des **SP** oder die Benutzerkennung ausgetauscht werden. Zudem könnte der angeforderte Sicherheitskontext oder andere Felder in der Nachricht modifiziert werden. Die optionale Signierung der `AuthnRequest` Nachricht könnte diese ganzen Felder vor Veränderung schützen. Dies wird aber nicht von Liberty Alliance verlangt. Hinzuweisen ist allerdings darauf, dass nur die `AuthnRequest` Nachricht bei einer Signierung geschützt würde, eine Modifikation des zu kontaktierenden **ASP** würde weiterhin unerkannt bleiben. Der Umstand, dass die Liberty Alliance Spezifikation nur verlangt, dass der Redirect als Kommunikationsprotokoll HTTPS benutzen soll, bewirkt, dass ein gesicherter Kanal erst zwischen Benutzer und **ASP** aufgebaut wird. Der Kommunikationskanal zwischen Benutzer und **SP** ist weiterhin nicht über SSL/TLS gesichert.

Das folgende Bild soll den Sachverhalt verdeutlichen.

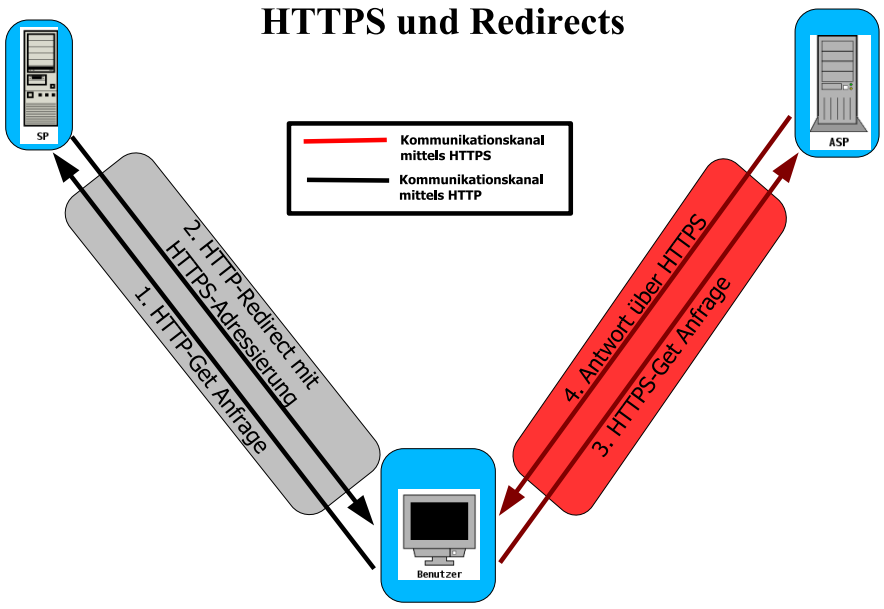

Abbildung 4.6: SSL und Redirects

Bei **1.** fragt der Benutzer eine Ressource über HTTP beim **SP** an. Dieser antwortet bei **2.** dem Benutzer mittels einem Redirect. Der Kommunikations-

endpunkt des Redirects wird über HTTPS adressiert. Die Antwort des **SP** wird aber bis zum Benutzer weiterhin über den HTTP-Kanal übertragen. Erst bei **3.** baut der Benutzerwebbrowser einen Kommunikationskanal mit dem **ASP** mittels HTTPS auf. Der **ASP** antwortet dem Benutzer bei **4.** über den HTTPS-Kanal. Bei der Benutzung von Redirects und HTTPS-Adressierung bleibt also ein Teilstück der Kommunikation ungeschützt.

Aus den eben genannten Gründen sollten bei beiden Systemen der Verbindungsaufbau zum **SP** schon mit SSL/TLS erfolgen.

Über die Anmeldeprozedur beim **ASP** gibt die Liberty Alliance Spezifikation keine Auskunft. Sie verlässt sich darauf, dass im anschließenden Assertion der Sicherheitskontext genug Auskunft über das Anmeldeverfahren gibt. Bei Microsoft Passport wird nur verlangt, dass die Anmeldedaten mit einem POST über HTTPS erfolgen. Dies beinhaltet auch den Fall, dass die Anmeldeseite, die der Benutzer beim **ASP** anfordert, auch mittels HTTP übertragen wird und nur der eingebettete Formularübertragungsbefehl die HTTPS-Adressierung benutzt. Dies hat zur Folge, dass ein Angreifer den Seiteninhalt bei der Übertragung vom **ASP** zum Benutzer verändern könnte. Dies könnte der Angreifer dazu nutzen, die vom Benutzer eingegebenen Anmeldeinformation zu dem Single Sign-On System auszulesen. Er müsste nur die Adresse innerhalb des Formulars ändern und die Antwort auf einem vom Angreifer kontrollierten Server im Internet leiten.

Gehen wir von einem korrekten Verlauf bis zur Anmeldung des Benutzers beim **ASP** aus, so antwortet der **ASP** mittels einem Redirect beim Benutzer dem **SP**. Wieder kann bei beiden Lösungen eine Integrität der übermittelten Daten nicht garantiert werden. Bei Microsoft Passport könnte der **SP**, also der Empfänger, ausgetauscht werden. Der Benutzer würde anschließend nicht den gewünschten **SP** kontaktieren, sondern irgendeinen Rechner im Internet. Somit könnte ein Angreifer an die verschlüsselten Cookies gelangen und diese eventuell weiterverwenden. Einen Empfängeraustausch kann der Angreifer auch für Trafficanalysen missbrauchen, indem er den Benutzer anschließend mit einem neuen Redirect zu dem ursprünglichen **SP** leitet. Dies würde dem Benutzer transparent erscheinen und er würde diese Umleitung wohl nicht bemerken. Ein Angreifer könnte auch einfach die verschlüsselten Cookiedaten verändern. Dies würde wohl sehr wahrscheinlich vom System erkannt werden, aber erst zu einem späteren

Zeitpunkt. Auf jeden Fall verlören die Cookies ihre Gültigkeit und wären nicht mehr benutzbar. Dies wäre auch bei den von Passport beim Benutzer gesetzten Cookies möglich, falls diese über HTTP übertragen werden.

Bei Liberty Alliance haben wir die gleiche Mindestanforderung wie beim ersten Redirect. Es wird verlangt, dass der **SP** mittels HTTPS adressiert wird. Ist der Kommunikationskanal zwischen Benutzer und **ASP** durch HTTP realisiert, würde der gesicherte Kanal erst auf dem Weg zwischen Benutzer und **SP** aufgebaut. Die Strecke zwischen Benutzer und **ASP** wäre ungesichert. Dies wäre eine der Spezifikation konforme Betriebsart, und ein Angreifer könnte diese Strecke missbrauchen, um den Benutzerbrowser zu einem anderen Server umzuleiten. Er könnte dem Benutzer jetzt u.a. einen Fehler vortäuschen und diesen bitten, sich erneut anzumelden. Wird der Benutzer gut genug getäuscht, könnte er leicht auf die erneute Aufforderung hereinfallen und durch eine Eingabe seine Anmeldedaten einem Fremden übermitteln. Dass der alleinige Einsatz von SSL/TLS nicht zwingend eine hohe Sicherheit bringt wird noch in Kapitel 4.6.4 genauer erläutert.

Wenn der **ASP** bei Liberty Alliance in der Antwort für den **SP** nicht ein Artefakt überträgt, sondern gleich die `AuthnResponse` Nachricht, kann ein Angreifer Teile der Nachricht verändern, ohne dass es auffällt. Die Liberty Alliance Spezifikation verlangt nur, dass das Assertion signiert sein muss. Wenn nicht die ganze `AuthnRespone` Nachricht signiert wird, dann kann ein Angreifer z.B. das $IssueInstant$ Feld und somit den Ausgabezeitpunkt verändern. Bei einem Artefakt könnte er u.a. das $IdentityProviderSuccinctID$ Feld ändern und dadurch den **SP** veranlassen, anschließend einen anderen **ASP** zu kontaktieren als ursprünglich vorgesehen.

Es sollte deswegen bei beiden Lösungen die indirekte Kommunikation zwischen **ASP** und **SP** auf beiden Kanälen vom Benutzerwebbrowser über HTTPS aufgebaut sein.

4.4.2 Manipulieren des Single Logout Prozesses

Microsoft Passport und Liberty Alliance verwenden Logout Protokolle, um das Abmelden eines Benutzers innerhalb einer Session den Teilnehmern zu signali-

sieren. Beide Lösungen bieten einem Angreifer die Möglichkeit, die Nachrichten zu verändern. Als Beispiel sei bei beiden Lösungen die Logout Methode mittels Image-URLs eingebettet in der Webseite vom **ASP** aufgeführt. Der **ASP** wird von dem Benutzer informiert, dass dieser sich aus seiner Session ausloggen möchte. Dazu generiert er für den Benutzer eine Webseite, in der der **ASP** für alle zur Zeit vom Benutzer besuchten **SP**s eine Image URL einbettet. Hinter der Image URL befindet sich aber beim **SP** nicht direkt ein Bild sondern ein Ausloggskript. Nachdem der **SP** den Logout Prozess vollzogen hat, übermittelt er anschließend das Bild. Anhand des Bildes kann somit der Status der Abmeldung des Benutzers signalisiert werden. Werden die vom **ASP** gesetzten Image URLs nicht mit einem Integritätsschutz bei der Übertragung versehen, so kann ein Angreifer die Kommunikationsendpunkte austauschen und somit ein erfolgreiches Ausloggen des Benutzers vortäuschen. Obwohl es scheint, als ob der Benutzer aus dem System ausgeloggt ist, bleibt er aber noch bei vielen **SP**s angemeldet. Dies könnte ein Angreifer für Session Hijacking Angriffe ausnutzen.

4.4.3 Sonstige Bemerkungen zur Integrität

Außer bei der Artefakt Methode bei Liberty Alliance existiert nie ein direkter Kommunikationskanal zwischen **ASP** und **SP**. Es baut immer der Benutzerwebbrowser die Verbindungen zu den anderen Teilnehmern auf. Somit ist der Benutzerrechner auch ein interessantes Ziel für einen Angreifer, um Modifikationen an den Nachrichten auszuführen. Es ist bei beiden Lösungen wichtig, die ausgetauschten Nachrichten zwischen **ASP** und **SP** mit einem separaten Integritätsschutz zu versehen, so dass **ASP** und **SP** Veränderungen an den Nachrichten erkennen können. Es reicht nicht aus, sich nur auf eine HTTPS-Übertragungen zu verlassen, da diese immer über den Benutzerbrowser als Endpunkte laufen.

In diesem Abschnitt wurde somit aufgezeigt, warum jegliche Kommunikation über HTTPS erfolgen sollte. Zusätzlich sollte ein zusätzlicher Integritätsschutz benutzt werden, da HTTPS nur die einzelnen Kommunikationskanäle zum Benutzerrechner hin schützen kann. Es sei noch darauf hingewiesen, dass neben dem Austausch von Kommunikationsendpunkten auch Modifikationen innerhalb einer XML-Struktur interessant sein könnten. XML benutzt häufig externe Ver-

weise, um Gültigkeitsbereiche oder Namensräume innerhalb des Dokuments zu beschreiben. Eventuell kann ein Austauschen dieser Verweise von einem Angreifer benutzt werden, um die Semantik der Nachricht zu ändern.

4.5 Angriffe auf die Authentikation

In diesem Abschnitt wird untersucht, ob die Teilnehmer sich gegenseitig authentisieren, bevor sie Nachrichten austauschen. Fehlt eine Authentikation an bestimmten Stellen im Protokoll, so besteht die Gefahr, dass sich ein Angreifer entweder als Empfänger oder Sender einer Nachricht ausgibt. Können die Teilnehmer dieses nicht erkennen, so kann dies unterschiedliche Auswirkungen auf die Sicherheit des Systems haben. Es wird bei einer Authentikation auch die Anzahl der Überprüfungen bei einer Kommunikation zwischen zwei Teilnehmern unterschieden.

- **One-Way Authentication:** Hier wird innerhalb einer Kommunikation nur einer der beiden Teilnehmer authentisiert. Also wird entweder der Sender oder nur der Empfänger überprüft.
- **Two-Way Authentication:** Bei dieser Variante müssen sich die beiden Kommunikationsteilnehmer gegenseitig authentisieren. Dabei wird sich in der Regel zuerst der Empfänger gegenüber dem Sender ausweisen und anschließend der Sender gegenüber dem Empfänger.

Es können zwei Klassen von Angriffen auf die Authentizität unterschieden werden. Da ist einmal die Fälschung und Generierung von Nachrichten und das Einspeisen dieser in das System. Der Angreifer verschickt blind Nachrichten an Teilnehmer des Systems und versucht zu erreichen, dass der Empfänger die zugesandten Nachrichten als legitime Nachrichten weiterverarbeitet. Bei der anderen Klasse von Angriffen geht man davon aus, dass der Angreifer irgendwo zwischen den legitimen Kommunikationspartnern auf den Kommunikationskanal Einfluss nehmen kann. Dabei speist er nicht nur selbst generierte Nachrichten in den Kanal ein, sondern kann auch Inhalte der originalen Nachrichten verändern oder

erweitern. Bietet das System eine Möglichkeit, dass der Empfänger sich gegenüber dem Sender authentisieren kann oder umgekehrt, so sollte ein dritter nicht die Möglichkeit haben, sich als jemand anderes auszugeben.

4.5.1 Man in the Middle Angriffe

Der Man in the Middle Angriff (abgekürzt auch M-i-M) ist ein typischer Vertreter der eben erwähnten zweiten Klasse. Der Trick besteht darin, beiden Teilnehmern den jeweils anderen Kommunikationspartner vorzutäuschen. Dieses kann sehr leicht gelingen, wenn die Kommunikationspartner sich nicht gegenseitig authentisieren. Der Angreifer nutzt diesen Umstand aus, um sich als der legitime Kommunikationspartner zu präsentieren.

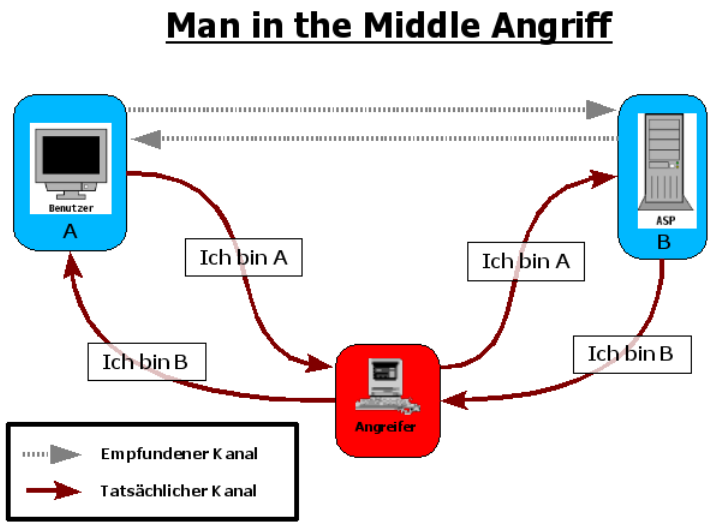

Abbildung 4.7: Man in the Middle Angriff

Dabei können sehr unterschiedliche Ansätze verfolgt werden, um sich als Angreifer in die Mitte der Kommunikation zu bringen. Ein paar Verfahren sollen nun vorgestellt werden.

- Modifikation der DNS-Informationen: Das „Domain Name System“ (DNS) sorgt für die Umwandlung von Domänennamen zu IP-Adressen. Dabei benutzt das DNS für die Auflösung eine Baumstruktur. Der anfragende Rechner schaut zunächst in seinem lokalen DNS-Cache nach und setzt bei einem

fehlenden Eintrag eine Anfrage an den eingestellten DNS-Server ab. Dieser DNS-Server fragt, falls er den Eintrag nicht lokal vorfindet, anschließend rekursiv andere DNS-Server ab und durchwandert so den Baum bis zu dem für diese Domäne zuständigen DNS-Server. Das DNS nutzt bei diesem Auflösungsprotokoll das in Kapitel 2.2.2 beschriebene verbindungslose ungeschützte UDP.

Bei DNS-Poisoning nutzt man aus, dass der Anfragende eine legitime Antwort nur anhand der Empfänger-IP und Portnummer sowie der Transaktions-ID (Werte können zwischen 1 und 65535 liegen) erkennen kann. Ein Angreifer setzt irgendwo in der Abfragekette bei den DNS-Servern an. Die Idee dabei ist, eine Anfrage an diesen DNS-Server zu stellen und anschließend den DNS-Server eine gespoofte Antwort annehmen zu lassen, noch bevor er die richtige Antwort erhält. Dabei muss der Angreifer nur die drei richtigen Werte für die IP-, Port- und Transaktionsnummer finden, und der DNS-Server darf die gewünschte Domain natürlich nicht schon im Cache haben. Die IP und den Port kann man herausbekommen, indem man zuvor eine Anfrage stellt nach einer Adresse, für die man selber den DNS-Server betreut. Multithreaded Serverdienste verwenden häufig der Geschwindigkeit wegen einen geöffneten Port für nachfolgende Anfragen weiter. Ein Angreifer hätte also die IP und einen Port für die gespoofte Nachricht. Viele DNS-Server im Internet nutzen noch ein sequenzielles Verfahren für die Transaktions-ID, dabei wird der Zähler bei jeder Anfrage um 1 erhöht. Erst BIND[5] 9 führte Zufallswerte für die Transaktions-ID ein. Erzeugt ein Angreifer anhand dieser Informationen viele parallele Antwortnachrichten mit unterschiedlichen Werten, hat er eine gute Chance, eine gültige zu treffen. Somit wäre der Cache des DNS-Servers in der Anfragekette vergiftet und alle Anfragen aus seinem unteren Teilbaum würde er mit den Angaben des Angreifer beantworten.

Außer dem DNS-Poisoning Angriff gibt es noch viele weitere Angriffe, um einem Client eine falsche IP für eine DNS-Anfrage zu übermitteln. Da wären

[5]BIND steht für „Berkeley Internet Name Domain“ und ist der mit Abstand am häufigsten eingesetzte DNS-Server im Internet.

das DNS-Hijacking oder das DNS-Spoofing. Ein Angreifer müsste nur einen DNS-Server in der Anfragekette unter seine Kontrolle bringen oder Einfluss auf den Kommunikationskanal haben. Wenn er Daten verändern würde, könnte dieses nicht vom Empfänger erkannt werden.

- Routinginformationen modifizieren: Ein Angreifer könnte aber auch eine Ebene tiefer ansetzen und die Routinginformationen manipulieren. Somit könnte er den Client trotz korrekter benutzter IP auf einen anderen Rechner lenken. Es gibt hier wieder viele unterschiedliche Ansätze. Er könnte Schwachstellen auf dem Router ausnutzen, um diese unter seine Kontrolle zu bekommen, oder er könnte einem schlecht konfigurierten Router falsche Daten unterschieben. Da Routinginformationen zwischen den Maschinen über das Netz ausgetauscht werden, kann er auch die eingesetzten Protokolle missbrauchen.

  Da wäre einmal das „Internet Control Message Protocol“ (ICMP), bei dem die Nachrichten ICMP Router Discovery Protocol (IRDP) und ICMP-Redirect Meldungen von Interesse sind. IRDP dient zum automatischen Auffinden eines Standardgateways für den Client. Es kann aber auch dazu benutzt werden, um nachträglich einen neuen Standardgateway anzugeben. Mit ICMP-Redirect Meldungen können Router einem Sender mitteilen, dass eine Route überlastet ist und eine alternative Route vorschlagen. ICMP-Redirects betreffen zwar immer nur das nächste direkte Gateway, aber mit einer Kette von gefälschten Nachrichten könnte eine Umleitung vollzogen werden.

  Da wegen der Größe des Internets die Verwaltung der Routinginformationen mit ICMP nicht praktikabel durchführbar ist, kamen im Laufe der Zeit weitere Routingprotokolle dazu. Router tauschen dabei ihre Routinginformationen untereinander aus, um eine optimale Übertragung der Nachrichten zu gewährleisten.

  Dabei unterscheidet man im Internet zwischen „Exterior Gateway Protocols“ (EGP) und „Interior Gateway Protocols“ (IGP). EGP kümmern sich um den Austausch von Routerinformationen über die „Autonomes System“ (AS) Grenzen hinweg. Ein AS bildet ein Teilnetz innerhalb des Internets

und wird in der Regel von einer Organisation verwaltet. Dies kann ein Internetdienstleister oder eine Firma sein. Im Gegensatz dazu werden IGPs dazu benutzt, Routinginformationen innerhalb eines AS auszutauschen. Es bestehen auch Möglichkeiten, Informationen aus EGPs mit IGPs weiterzuverarbeiten.

Ein prominenter Vertreter der EGP ist das gleichnamige „Exterior Gateway Protocol“. Dieses Protokoll hat aber keine großen Sicherheitsvorkehrungen und läßt sich leicht missbrauchen, um Netzwerkverkehr zu einem anderen AS umzuleiten. Deswegen wird heute häufiger das „Border Gateway Protocol“ (BGP) eingesetzt. BGP setzt auf TCP auf und arbeitet in der Anwendungsschicht. BGP erhöht zwar die Sicherheit der Router vor eingespeisten Falscheinträgen, bietet aber auch weiterhin Möglichkeiten für einen Angreifer. Allerdings ist dies nicht mehr so einfach wie bei seinem Vorgänger.

Innerhalb einer AS werden andere Protokolle für den Austausch der Routinginformation benutzt. „Routing Information Protocol “ (RIP) ist ein solches Protokoll. Version 1 arbeitet dabei mit Broadcasts und Version 2 mit Multicasts Nachrichten. Innerhalb einer Routergruppe tauscht darüber jeder Router mit allen anderen seine Routinginformationen aus. Ein Angreifer muss nur eine selbsterzeugte Nachricht mit niedriger Metrik aussenden und schon fließt der Verkehr über seinen Rechner.

RIP wird deswegen nicht mehr häufig im Internet eingesetzt und ist durch das „Enhanced Interior Gateway Routing Protocol“ (EIGRP) ersetzt worden. Das EIGRP baut auf das „Interior Gateway Routing Protocol“ von Cisco auf. EIGRP arbeitet wie RIPv2 mit Multicasts Nachrichten, führt aber die Option einer Authentikation mittels Passwort und MD5 Hash ein. Diese Option ist aber nicht immer aktiviert. Das liegt u.a. daran, dass es keinen Schlüsselverteilungsmechanismus gibt. Soll ein Passwort oder MD5 Hash geändert werden, so muss dies manuell bei allen teilnehmenden Routern geschehen.

Natürlich existieren noch weitere Protokolle für den Austausch von Routinginformationen innerhalb einer AS, wie das „Open Shortest Path First“

Protokoll (OSPF). Dieses eignet sich besonders gut für große AS.

Je nachdem, wo ein Angreifer innerhalb der Routingtopologie ansetzt und welche Protokolle verwendet werden, kann er eventuell schon mit kleinen Eingriffen eine Umleitung erreichen. Bei den neueren Protokollen kann nicht davon ausgegangen werden, dass dies nicht mehr möglich ist, es erfordert nur mehr Aufwand für den Angreifer. Er muss kompliziertere Einträge fälschen und sich möglicherweise zuvor statische Authentikationsinformationen aus umlaufenden Nachrichten beschaffen. Beides ist aber anhand der Struktur des Internets für einen Angreifer mit genügend Energie weiterhin möglich.

- Manipulation der ARP Einträge: Auch eine Ebene tiefer innerhalb der verwendeten Protokollhierarchie hat ein Angreifer die Chance, Nachrichten umzuleiten. Dazu nutzt er aus, dass das verwendete „Address Resolution Protocol“ (ARP) überhaupt keine Sicherheitsmechanismen besitzt. ARP wird benutzt, um in lokalen auf Ethernet basierenden Netzwerken, IP Adressen auf Ethernetadressen aufzulösen. Ein Sender muss zunächst einmal die MAC-Zieladresse[6] des Empfänger herausfinden, bevor er die Nachricht verschicken kann. Dazu fragt er mittels einer Broadcastnachricht[7] innerhalb seines Segmentes nach, wer die MAC-Adresse für die gewünschte IP kennt. Der Rechner mit der gewünschten IP antwortet auf den Broadcast mit einer ARP-Antwortnachricht. In dieser ARP-Nachricht steht, welche IP-Adresse mit welcher MAC-Adresse erreicht wird.

  Da der Absender nicht verifiziert werden kann, braucht ein Angreifer nur schneller als der richtige Rechner zu antworten. Dazu schickt er eine gefälschte ARP-Antwort mit seiner MAC-Adresse für die gewünschte IP. Der anfragende Rechner kann die gefälschte Nachricht nicht erkennen und trägt die Daten in seinem Cache ein. Diesen Angriff nennt man ARP-Spoofing. Meistens gilt für einen Eintrag im ARP-Cache eine Gültigkeitsdauer von 20 Minuten. Somit braucht ein Angreifer häufig nicht lange zu warten, bis der Client erneut eine ARP-Anfrage stellen wird, sollte der richtige Empfänger

[6] MAC (Media Access Control) ist das Adressformat für Ethernetframes.
[7] Als MAC Empfängeradresse wird dabei ff-ff-ff-ff-ff-ff benutzt.

noch im ARP-Cache des Clients stehen. Das ARP kann aber auch benutzt werden, um ohne Anfragenachrichten Antworten zu versenden. Viele Clients aktualisieren dann ihren Cache mit dieser neuen Information. Nur wenn der Client eine Liste mit seinen ARP-Anfragen führt, kann er eine unbeauftragte Antwort erkennen und verwerfen.

Gingen solche Angriffe mit Hubs früher sehr einfach, so funktioniert dieses nicht mehr so leicht mit Switches. Switche merken sich in der Regel, welche MAC-Adresse an welchem Port angeschlossen ist und erlauben nur ARP-Antworten mit MAC-Adressen, die auch an dem Port angemeldet sind. MAC-Adressen sollten eigentlich weltweit einzigartig pro Teilnehmer sein, sehr viele Ethernetkartentreiber erlauben aber das Verändern der MAC-Adresse. Diesen Umstand und eine DoS-Attacke gegen den wahren Empfänger könnte der Angreifer ausnutzen, um dennoch eine Umleitung zu erreichen.

Wesentlich leichter geht es aber bei vielen Switchen mit einem ARP-Flooding-Angriff. Da ein Switch eine Tabelle im Speicher führen muss, an welchem Port welche MAC-Adressen verbunden sind, ist die Idee bei diesem Angriff, die Tabelle überlaufen zu lassen. Dazu sendet ein Angreifer einfach eine große Menge an gefälschten ARP-Nachrichten an den Switch mit unterschiedlichen MAC-Adressen. Irgendwann hat der Switch keinen Platz mehr für zusätzliche Einträge und schaltet anschließend in einen Hub Modus. In diesem Hub Modus sendet der Switch jetzt jede Nachricht an jeden Port und der Angreifer kann mit seinem ARP-Spoofing Angriff beginnen.

ARP-Probleme wurden lange Zeit vernachlässigt. Dies ist umso gefährlicher, wenn man bedenkt, dass fast alle Kabelmodemanschlüsse eigentlich über ein Ethernetnetzwerk realisiert sind und somit ein gutes Ziel bieten für ARP-Angriffe. Abhilfe gegen ARP-Angriffe könnte nur mit Hilfe einer festen statischen Zuweisung von erlaubten MAC-Adressen zu Ports auf dem Switch gelingen. Dies ist aber wegen der häufig vorhandenen dynamischen Netzwerkstruktur nicht möglich, geschweige denn handhabbar.

- Proxy Server oder Application Level Gateways: Bisher haben wir uns An-

griffe für einen Man in the Middle angeschaut, bei der die Verkehrswege geändert werden. Dabei wurden Informationen zu den Verkehrswegen gefälscht oder verändert. Aber auch ohne den Verkehrsweg der Nachrichten zu ändern, kann sich ein Angreifer zwischen die Kommunikation setzen. Neben den Routern und Switches und DNS-Servern gibt es noch andere Stationen die für einen Angreifer interessant sein können. Da wären eventuell Proxy Server oder Application Level Gateways auf der Übertragungsstrecke, die auch missbraucht werden könnten.

Proxy Server ist ein Stellvertreterdienst. Dabei kontaktiert der Client nicht den gewünschten Server direkt. Er stellt stattdessen die Anfrage an den Proxy Server, und dieser baut anschließend anstelle des Clients die Verbindung mit dem Server auf und fordert die Ressourcen an. Hat er die Ressourcen erhalten, so reicht der Proxy Server diese Daten weiter an den anfragenden Client. Ein Einsatz eines Proxy Servers kann aus unterschiedlichen Gründen innerhalb eines Netzes erfolgen. Durch einen Proxy unterbindet man eine direkte Kommunikation zwischen den Teilnehmern innerhalb des Netzwerkes mit außerhalb liegenden Stationen. Nur der Proxy Server darf eine Kommunikationsverbindung nach außen aufbauen. Ein Proxy Server kann auch eingesetzt werden, um weniger Verbindungen in das externe Netz aufzubauen. Da ein Proxy Server meist die angefragten Ressourcen zwischenspeichert, können nachfolgende Anfragen von Clients aus dem internen Netz nach denselben Ressourcen sofort von dem Proxy Server beantwortet werden. Proxy Server werden meist bei Firmen eingesetzt, aber auch mehrere Internet Service Provider bieten ihren Kunden eine Benutzung von Proxy Servern an. Einem Client muss ein Proxy Server nicht immer bekannt sein. Transparente Proxy Server arbeiten für den Client transparent im Netz.

Application Level Gateways sind Firewalls, die den Inhalt der verwendeten Protokolle nicht nur bis zur TCP/UDP-Ebene betrachten, sondern auch die Nachrichten aus der Applikationsebene untersuchen. Anhand von Filterregeln erlauben sie bestimmten Verkehr oder unterbinden diesen. Dadurch, dass solche Firewalls auch die Applikationsebene mitbetrachten,

können Application Level Gateways aber auch den Inhalt dieser Nachrichten austauschen. Meistens wird dabei dann ActiveX oder JavaScript aus den HTML-Seiten entfernt. Sie eignen sich aber auch, um Malware oder Spam Emails zu blockieren.

Hat ein Angreifer einen Proxy Server oder ein Application Level Gateway unter seiner Kontrolle, so kann er genauso gut in die Kommunikation eingreifen wie bei einer Umleitung. Die notwendigen Programme für eine Modifikation der ausgetauschten Nachrichten sind dabei schon gleich auf diesen Servern installiert.

- Kontrolle des Benutzerrechners: Bei Microsoft Passport läuft jegliche Kommunikation zwischen **ASP** und **SP** über den Clientrechner. Auch bei Liberty Alliance ist dies der Fall; bis auf einen optionalen direkten Authentication Assertion Austausch zwischen **SP** und **ASP** wird die restliche Kommunikation über den Benutzerclient abgewickelt. Da aber die notwendigen Daten für den Authentication Assertion Austausch in vorherigen Nachrichten ausgehandelt und diese Nachrichten über den Client ausgetauscht werden, ergibt sich kein Unterschied zu Microsoft Passport.

  Hat also ein Angreifer Kontrolle über den Benutzerrechner, kann er auch einen Man in the Middle Angriff ausführen. Bei der heutigen Bedrohung von Clientsystemen von Malware sollte dieser Aspekt nicht unterschätzt werden.

- Hotspots und öffentliche Netzzugänge: An immer mehr öffentlichen Plätzen oder in öffentlich zugänglichen Räumen werden heute mobile Internetzugänge angeboten, diese bezeichnet man meist als Hotspots. Sei es in Gaststätten, in Bahnhöfen, in Bibliotheken oder an anderen Orten, häufig wird dabei Wireless LAN verwendet, um den Client anzubinden. Dabei wird einer der IEEE 802.11 Standards der „Institute of Electrical and Electronics Engineers“ (IEEE) eingesetzt. Bei Wireless LAN verbinden sich dabei die Clientrechner an einen „Access Point“ (AP). Der AP wird dabei im „Infrastrukturmodus“ betrieben. Dies erlaubt die parallele Anbindung von mehreren Clientrechnern. Da die meisten öffentlich zugänglichen Hotspots mit

unzureichenden Sicherheitslösungen versehen sind, kann ein Angreifer sich leicht als legitimer AP ausgeben. Dazu benötigt er nur ein Laptop/PDA mit einer Netzwerkkarte und einer Software, die diese Netzwerkkarte im AP-Modus betreiben kann. Jeglicher Verkehr über den AP des Angreifers leitet dieser anschließend an den richtigen AP weiter. Je nach Konfiguration des richtigen APs nutzt der Angreifer für die Weiterleitung zusätzlich „Network Adress Translation“ (NAT), damit er als einziger Client bei dem richtigen AP erscheint, obwohl er die Verbindungen von mehreren Clients weiterleitet. Meldet sich nun ein Benutzer an dem AP vom Angreifer an, so kann der Angreifer jegliche Kommunikation bei der Übertragung mitlesen und modifizieren.

Dieser Angriff funktioniert auch mit geschützten Wireless LANs. Wird das unzureichende Sicherheitsprotokoll „Wired Equivalent Privacy“ (WEP) verwendet, muss der Angreifer zwar zunächst den Zugang zum Funknetzwerk hacken, anschließend kann er aber den gleichen Angriff wie eben beschrieben ausführen. Das Überwinden von WEP stellt für einen Angreifer kein großes Hindernis dar, da dieses Protokoll eine Vielzahl von Designschwächen aufweist. Setzen die Betreiber des richtigen APs „Wi-Fi Protected Access“ (WPA) als Sicherheitsprotokoll ein, so kann bei einer Wahl von schlechten Passwörtern ein Angreifer auch dieses Verfahren mittels eines Wörterbuchangriffes überwinden[8]. Ein Angreifer muss für die Vortäuschung eines AP nicht unbedingt in der Nähe des anzugreifenden Clientrechners sein. Mit Richtfunkantennen kann er auch aus weiterer Entfernung als der richtige AP zum Clientrechner ein stärkeres Signal beim Client liefern und so vom Client ausgewählt werden.

Natürlich kann der Angreifer diesen Angriff auch sehr leicht durchführen, wenn er ein legitimer Benutzer eines gesicherten Funknetzwerkes ist. In diesem Falle ist es nicht notwendig die Sicherheitsmechanismen zu überwinden, der Angreifer kann den anderen Teilnehmern einen gefälschten AP vortäuschen.

[8]Siehe dazu auch
`http://www.tinypeap.com/docs/WPA_Passive_Dictionary_Attack_Overview.pdf`

Auch bei anderen Funkverfahren kann ein Angreifer versuchen, sich in die Mitte der Kommunikation zu bringen. Sei es mit Bluetooth oder Mobiltelefonfunk. Bluetooth könnte dabei zwischen Laptop und Mobiltelefon beim Benutzer zum Einsatz kommen. Aber auch bei dem Gebrauch des Mobiltelefons als Benutzerclient kann der Benutzerverkehr mittels eines IMSI-Catcher über eine Station eines Angreifers relayed werden.

Selbst bei einem kabelgebundenen Netzwerkzugang hat ein Angreifer weiterhin die Möglichkeit, die Kommunikation über seinen Rechner laufen zu lassen. Häufig findet die notwendige Informationsverteilung an den Client für eine erfolgreiche Netznutzung über das „Dynamic Host Configuration Protocol“ (DHCP) statt. DHCP übermittelt dabei dem Client die zu benutzende IP, sowie weitere Netzwerkinformationen. U.a. auch das Standardgateway für die Kommunikation. Ein Client sendet für die Anfrage ein UDP Broadcast Netzwerkpaket mit der Absender-IP 0.0.0.0. Anschließend wartet der Client auf eine Broadcastantwort des DHCP-Servers. Antwortet ein an das Netzwerk angeschlossener Angreifer dem Client schneller als der richtige DHCP-Server, so kann er dem Client falsche Netzwerkeinstellungen übermitteln. Der Angreifer wird seinen Rechner als Standardgateway angeben und somit wird jeglicher Verkehr des Benutzerclients über den Angreifer umgeleitet.

4.5.2 Den Benutzer zu einem anderen ASP umleiten

Wie oben aufgezeigt, existieren somit eine Vielzahl von Möglichkeiten für einen Angreifer eine Man in the Middle Attacke durchzuführen. Hat sich ein Angreifer in die Mitte der Kommunikation gebracht, kann er unterschiedlich auf das Single Sign-On Protokoll Einfluss nehmen.

In [KR00] beschreiben Kormann und Rubin, wie man mittels eines Man in the Middle Angriffs den Benutzer an einen gefälschten **ASP** leiten kann. Eine generelle Möglichkeit wäre folgendes Szenario:

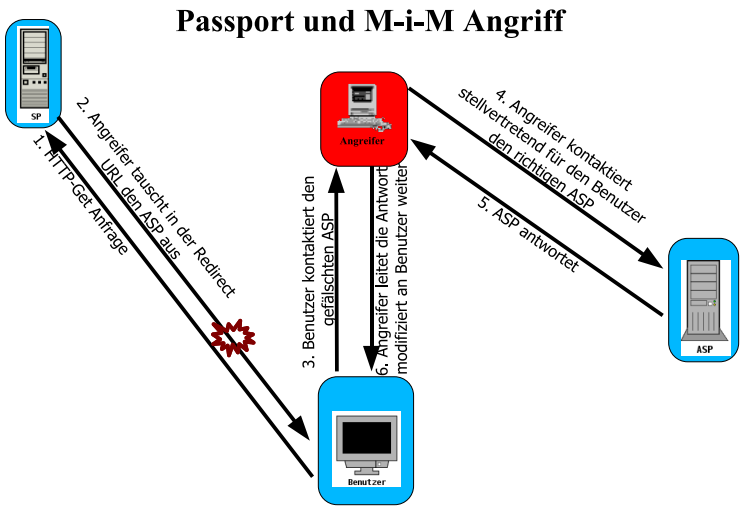

Abbildung 4.8: Man in the Middle Angriff (M-i-M) bei Microsoft Passport

Dadurch, dass sich der Angreifer am falschen **ASP** anmeldet, kommt der Angreifer nicht nur an die Masterzugangsdaten für das Single Sign-On System, er kann auch anderen Missbrauch vollziehen. Er kann sich mit den empfangenen Daten beim **SP** anmelden. Zusätzlich kann er sich für beliebige **SP** vom **ASP** die jeweilige Zugangserlaubnis besorgen. Dazu überträgt er das *MSPAuth* Cookie für den **ASP**. Erschwerend kommt hinzu, dass der Angreifer die Option für eine permanente Speicherung des **ASP**-Zugangscookie auswählen kann. Somit erhält der Angreifer immer wieder erneut Zugang zu dem Passport System unter der Identität des Benutzers.

Eine Authentisierung des **ASP** beim Benutzer muss auf jeden Fall erfolgen, um solche Risiken zu minimieren. Liberty Alliance verlangt bei diesem Schritt die Verwendung von HTTPS mit Serverzertifikatsverifizierung. Dass es aber trotz der Benutzung von HTTPS unter Umständen weiterhin möglich ist, für einen Angreifer eine solche Attacke auszuführen wird in Kapitel 4.6.4 aufgezeigt.

Zugangszeugnisse entwenden

Bei Liberty Alliance kann sich ein Angreifer erfolgreich für einen Benutzer bei einem **SP** einloggen, selbst wenn der Benutzer über einen sicheren Kanal mit dem **ASP** kommuniziert.

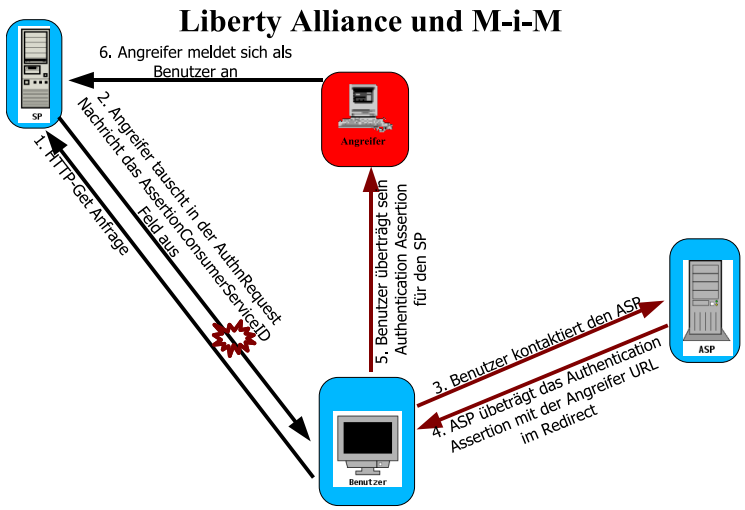

Abbildung 4.9: Man in the Middle Angriff (M-i-M) bei Liberty Alliance

Wie zuvor schon erläutert wurde, kann es bei Liberty Alliance vorkommen, dass eine `AuthnRequest` Nachricht zwischen **SP** und Benutzer über HTTP übertragen wird. Ist diese Nachricht dabei nicht signiert, weil der **SP** fälschlicherweise davon ausgeht, dass eine HTTPS-Adressierung im Redirect ausreicht, dann kann der Angreifer Felder in der Nachricht austauschen. Dieses nutzt der Angreifer aus, um an das Authentication Assertion zu gelangen. Anschließend kann der Angreifer sich beim **SP** als Benutzer anmelden. Dieses Beispiel funktioniert genauso bei Microsoft Passport. Der Angreifer müsste nur die Rücksprung-URL innerhalb des Querystrings beim Redirect ändern.

Das Problem bei beiden Systemen ist der fehlende dynamische Authentikator. Eine Lösung wäre ein Feld in dem Authentikator, welches die Clientadresse beschreibt. Weder Liberty Alliance noch Microsoft Passport führen in ihren Zugangszeugnissen ein Feld, in dem die Clientadresse vermerkt wird. Doch selbst wenn so ein Feld vorhanden wäre, so könnte immer noch Betrug stattfinden. Sendet der Angreifer nämlich von der gleichen Adresse, könnte die Anfrage nicht unterschieden werden. Zur Zeit wäre es bei beiden Systemen möglich, dass Benutzer A ein gültiges Authentication Assertion oder Passport Cookie erhält und dieses Assertion oder Cookie einem Benutzer B überreicht. Dieser Benutzer B kann sich von irgendeinem Rechner aus als Benutzer A anmelden.

Während bei Kerberos der Client immer den Timestamp mit dem geheimen

Schlüssel verschlüsselt und somit nachweist, dass er Kenntnis von dem in der Nachricht eingepackten Schlüssel hat, fehlt eine solche dynamische Funktion bei beiden Lösungen. Da bei beiden Systemen Standardbrowser als Clientsoftware vorgesehen sind, könnte ein signiertes Javaapplet dazu benutzt werden, um einen solchen dynamischen Authentikator auf dem Clientrechner zu generieren.

Natürlich kann ein Angreifer einem Benutzer nicht nur auf diese Weise einen **ASP** vortäuschen oder versuchen, an das Zugangszeugnis zu gelangen. Anhand der Protokolle können noch viele weitere Szenarien überprüft werden, bei denen ein Angreifer irgendwo in der Kommunikation Einfluss nehmen kann. Er könnte u.a. dem Benutzer einen gefälschten **SP** präsentieren oder andere Felder in den Nachrichten ändern.

4.5.3 Missbrauch von Zugangszeugnissen beim SP

Abschließend soll in diesem Kapitel noch ein weiteres Beispiel erläutert werden. Dabei wird betrachtet, ob ein **SP** ein erhaltenes Zugangszeugnis bei einem anderen **SP** missbrauchen kann.

Bei Microsoft Passport kann ein **SP** die empfangenen Daten nicht bei einem anderen **SP** verwenden. Das notwendige Cookie, welches den Benutzer beim **SP** ausweist, ist bei jedem **SP** mit einem anderen Schlüssel verschlüsselt. Sendet der betrügende **SP** das erhaltene Cookie an einen anderen **SP**, so wird dieser die Daten nicht sinnvoll entschlüsseln können.

Bei Liberty Alliance wird ein **SP** das erhaltene Authentication Assertion nicht für eine Anmeldung bei einem anderen **SP** benutzen können. Innerhalb eines Authentication Assertion existiert einmal das Feld $AudienceRestrictionCondition$. Dieses Feld kann dazu benutzt werden, um den **SP** und die zu benutzende Ressource beim **SP** anzugeben. Dieses Feld muss aber nicht innerhalb eines Authentication Assertions vorhanden sein. Sollte das Feld nicht vorgesehen sein, dürfte es trotzdem nicht möglich sein für den betrügenden **SP**, dieses Assertion bei einem anderen **SP** zu benutzen. Anhand der unterschiedlichen Benutzerhandles für jeden einzelnen **SP** sollte es nicht funktionieren, dass der **SP** sich als der Benutzer bei einem anderen **SP** ausgibt. Es bleibt aber bei fehlendem $AudienceRestrictionCondition$ Feld die Gefahr, dass die eingetragene

Benutzer-ID im $SubjectStatement$ einen gültigen anderen Benutzer bei einem anderen **SP** identifiziert und somit ein Authentication Assertion doch erfolgreich für eine Authentikation bei einem anderen **SP** benutzt werden könnte. Deswegen sollte jede Authentication Assertion das Feld $AudienceRestrictionCondition$ benutzen, um diese Gefahr auszuschließen.

Was ist aber, wenn mehrere **SP** unter einer Organisationsstruktur agieren? Wird in der Return-URL, die der **ASP** für die Antwort zum **SP** benutzt, nur eine generelle Adresse beim **SP** verwendet, so kann der **ASP** nicht erkennen, welche Ressource der Benutzer beim **SP** benutzen möchte. Dies erhöht den Datenschutz des Benutzers. Es lässt sich aber u.U. von einem betrügenden **SP** missbrauchen.

Der **SP** könnte z.B. als Diensteprovider für Dritte auftreten. Dies bedeutet, dass unter der $SP - URL$ unterschiedliche Dienstleistungen angeboten werden, die nicht von einer Instanz verwaltet werden. Somit könnte dann ein Subteilnehmer das empfangene Zugangszeugnis nutzen, um einen anderen Dienst beim **SP** zu gebrauchen. Solche Szenarien wären bei **SP**s denkbar, die als Shopreseller agieren. Dies ist heute schon vielerorts im Internet anzutreffen, sei es bei Ebay oder bei großen Providern wie Strato oder 1&1. Solche **SP** bieten die Möglichkeit, unter der Domäne des **SP** eine eigene Dienstleistung anzubieten.

Aber auch wenn der **SP** nur von einer Organisation benutzt wird, könnten dem Benutzer unterschiedliche Dienste angeboten werden. In einem solchen Fall könnte das übersendete Zugangszeugnis bei dem **SP** für eine andere Ressource benutzt werden, als der Benutzer angefragt hatte. Ein Beispiel wäre eine Bank, die Aktienkurse anbietet. Obwohl der Benutzer nur nach den personalisierten Aktienkursen angefragt hat, nutzt der **SP** das Zugangszeugnis, um eine Kontotransaktion zu legitimieren.

Diese Problematik trifft sowohl bei Microsoft Passport als auch auf Liberty Alliance zu. Eine Lösung wäre, die angeforderte Ressource innerhalb des Zugangszeugnisses anzugeben. Somit könnte aber ein **ASP** registrieren, welche Seiten der Benutzer beim **SP** nutzt. Deshalb wäre es eine bessere Lösung, wenn der **SP** für jede gewünschte Benutzeranfrage, bei der er den **ASP** kontaktieren muss, eine Zufalls-URL für die Antwort des **ASP** generiert. Diese Zufalls-URL verweist in einer geführten Liste beim **SP** auf die wirkliche Ressource, die der Benutzer angefragt hatte. Somit kann der **SP** die unterschiedlichen Ressourcen

und Zugangszeugnisse unterscheiden und der **ASP** kann nicht erkennen, welche Ressourcen der Benutzer beim **SP** nutzen möchte.

4.6 Weitere Infrastrukturbetrachtungen

4.6.1 Probleme bei der Verwendung von Cookies

Cookies sollten nicht zur Speicherung von sensiblen Information benutzt werden. Ein herkömmlicher Benutzer ohne großes technisches Hintergrundwissen wird die Auswirkungen seines Handelns nicht einschätzen können. Aus der Sicherheitsperspektive ist jedoch das Speichern der Zugangsinformationen für die Nutzung der Single Sign-On Dienste ein massive Herabsetzung des Sicherheitsniveaus.

Microsoft Passport nutzt Cookies sowohl für das Zugangszeugnis als auch für die Übermittlung von Profildaten. Dabei besteht die Gefahr, dass diese Cookies leicht von einem Dritten entwendet werden können. Hat ein Angreifer das *MSPAuth* Cookie für einen **SP** beim Benutzer ausgelesen, so kann er sich damit anschließend beim **SP** anmelden. Hat er das *MSPAuth* oder *MSPSecAuth* Cookie für den **ASP**, so kann er sich sogar neue Zugangszeugnisse für weitere **SP** besorgen.

Um an die Cookies zu gelangen, existieren unterschiedliche Verfahren. Zunächst einmal könnte jeder, der Zugang zu Subdomänen des **SP** hat, die **SP**-Cookies auslesen. Der Benutzerbrowser übersendet Cookies, wenn der anfragende Server einen identischen Teil der Domäne enthält, unter der das Cookie beim Benutzer gesetzt wurde. Wurde das Cookie z.B. unter der Domäne www.sp.de gesetzt, kann auch der Rechner kunde.sp.de das Cookie erhalten. Nur die beiden obersten Domänenkennungen (sp.de) innerhalb der Anfrage URL werden vom Webbrowser überprüft, und bei einer Übereinstimmung sendet er die Cookies. Erschwerend kommt hier der Umstand zum Tragen, dass mehrere Länder die TLD[9] aus zwei Oberdomänen erstellen. Großbritanien nutzt z.B. com.uk als TLD. Dies bedeutet, dass wenn der Webrowser nicht eine Ausnahmeliste von solchen Domänen führt, jede unter com.uk registrierte Domäne Cookies abfragen könnte.

[9] TLD steht für „Top Level Domain“ und stellt die höchste Ebene der Namensauflösung dar.

Es existieren weiterhin Angriffspunkte, selbst wenn nur vertrauenswürdige Instanzen unter der Domäne agieren, die das Cookie gesetzt hat. Besonders die schon genannten Man in the Middle Attacken oder auch Cross Site Scripting (XSS) seien hier erwähnt. XSS-Angriffe werden genauer im Anschluss bei Kapitel 4.6.2 erläutert.

Eine besondere Gefahr birgt die Benutzung von permanenten Cookies, wie es Microsoft Passport anbietet. Permanente Cookies bleiben auch nach einer Benutzersession aktiv. Dies bedeutet, dass solche Cookies vom Webbrowser auf die Festplatte geschrieben werden, um bei einem erneuten Starten des Webbrowsers wieder eingelesen zu werden. Neben dem Diebstahl der Datei, in der die Cookies auf dem Benutzerrechner gespeichert sind, ist der Benutzer vor allem bei öffentlich zugänglichen Rechnern gefährdet. Nutzt der Benutzer permanent gespeicherte Cookies auf einem solchen Rechner, kann jede nachfolgende Person sich als der Benutzer innerhalb des Systems anmelden. Dabei hilft es dem Benutzer nicht, wenn er zum Abschluss seiner Single Sign-On Sitzung den Browser beendet. Erwähnenswert sei noch, dass diese Problematik, auch bei der reinen Verwendung von nicht permanenten Cookies bestehen kann. Fast alle bekannten Webbrowser ermöglichen nämlich die Erstellung von versteckten Fenstern. Somit hält der Webbrowser ein Fenster offen, welches der Benutzer nicht ohne weiteres sieht. Schließt der Benutzer gegen Ende seiner Sitzung das Webbrowserfenster im Glauben, somit auch den Webbrowser zu beenden und damit auch alle Sessioncookies zu löschen, so bleibt in Wirklichkeit der Webbrowser inklusive der Cookies im Arbeitsspeicher des Rechners.

Leider funktioniert das Microsoft Passport System nur mit Cookies, und Microsoft erwähnt ausdrücklich, dass ein Abschalten von Cookies das System unbrauchbar macht [Mic04l]. Einerseits rät Liberty Alliance zwar von der Benutzung von Cookies ab,

> „Use of cookies by implementors and deployers should be carefully considered, especially if a cookie contains either or both personally identifying information and authentication information.“ [CHKT03][S.20]
>
> ,

Andererseits gestattet Liberty Alliance aber später dann doch die Benutzung.

> „The fact that identity providers cannot arbitrarily send data to service providers via cookies does not preclude identity providers and service providers from writing cookies to store local session state and other, perhaps persistent, information.“ [CHKT03][S.21]

Der Grund für die Erlaubnis, doch Cookies zu verwenden, dürfte sein, dass Liberty Alliance nicht beschreibt, wie ein **ASP** einen angemeldeten Benutzer verfolgen kann. Es werden keine Angaben gemacht, wie ein **ASP** verifizieren kann, ob diese Benutzeranfrage auch wirklich von dem angemeldeten Benutzer stammt. Der **ASP** weiß nur, dass der Benutzer im System angemeldet ist. Microsoft Passport nutzt zu diesem Zweck das *MSPAuth* Cookie vom **ASP**, welches der **ASP** bei jeder Anfrage des Benutzers auslesen kann, um festzustellen, ob die Anfrage auch wirklich vom Benutzer stammt.

Bei Microsoft Passport kommt noch hinzu, dass alle Cookies, die vom **ASP** gesetzt werden, immer mit dem gleichen Schlüssel verschlüsselt sind. Dieser Schlüssel ist bei allen Benutzern des Systems gleich. Kann ein Angreifer den Schlüssel herausfinden, so kann er anschließend für jeden Benutzer des Systems ein gültiges Zugangszeugnis ausstellen. Deswegen sollten bei Passport für jeden Benutzer die Cookiedaten mit einem vom Hauptschlüssel generierten benutzergebundenen Schlüssel verschlüsselt werden.

Beide Verfahren sehen auch keinen Mechanismus vor, der gewährleistet, dass Cookies, die vom **SP** gesetzt wurden, auch wirklich bei einer Beendigung einer Benutzersitzung gelöscht werden. Diese Verantwortung wird dem **SP** übertragen.

> „The participating site is responsible for deleting all cookies, including those in subdomains.“ [Mic04i]

Dadurch ist die Gefahr gegeben, dass sich Cookies beim Benutzer ansammeln, was Begehrlichkeiten seitens Dritter hervorrufen könnte.

4.6.2 Eingebettete HTML-Elemente und XSS-Angriffe

Kann ein **SP** Elemente in Webseiten beim **ASP** einbinden, so könnte dieses für einen Angriff benutzt werden.

Gehen wir zunächst davon aus, dass ein **SP** bestimmte Elemente in der Loginseite beim **ASP** platzieren darf. Bei Microsoft Passport ist dies möglich, um die Loginseite beim **ASP** mit Elementen des **SP** zu gestalten. Liberty Alliance schließt eine solche Möglichkeit bei den **ASP**s ebenso nicht aus.

Somit kann der **SP** Elemente in der Loginseite platzieren, die nicht unter der Kontrolle des **ASP** liegen. Ein **SP** kann damit die Semantik der Webseite vom **ASP** verändern, indem er Skripte in der Loginseite einsetzt, die beim Benutzerbrowser unter dem Kontext des **ASP** ausgeführt werden. Außer der Einflussnahme auf die Kommunikation zwischen dem Seitenabrufer und dem **ASP**, könnte er aber auch noch wesentlich mehr Informationen erhalten. Nutzt der **ASP** z.B. den Apache Webserver mit aktivierten PHP-Interpreter, so könnte ein **SP**, wenn er ein PHP-Skript integrieren kann, jeglichen gesicherten Nachrichtenverkehr beim **ASP** mitlesen. Dies war möglich, da der Apacheserver Anfang 2004 eine Schwachstelle[10] hatte.

Selbst wenn der **ASP** nur bestimmte HTML-Elemente wie Bilder und Textelemente erlaubt, so muss dadurch kein Schutz vor Missbrauch gegeben sein. Immer wieder werden Schwachstellen in Webbrowsern gefunden, die mit solchen Elementen ausgenutzt werden können. Zum Beispiel sind im Jahre 2004 mehrere Schwachstellen in Microsofts Internet Explorer veröffentlicht worden, die die Möglichkeit bieten, über Image Tags Malware beim Benutzer einzuschleusen[11].

Die aufgezählten Beispiele zeigen nur einen kleinen Ausschnitt der Gefahren, die durch die Erlaubnis fremder Elemente bei Webseiten entstehen können. Im Allgemeinen können Webdienste, die eine Dateneingabe erlauben, anfällig für Cross Site Scripting Angriffe (XSS) sein.

Man kann XSS-Attacken unterscheiden in clientseitige und serverseitige XSS. Bei clientseitigem XSS versucht der Angreifer, einen Code einer clientseitigen Skriptsprache bei einem fremden Server unterzubringen. Kann der Angreifer diesen Code irgendwo auf einer Webseite beim Server platzieren, braucht er nur noch zu warten, bis ein Benutzer diese Seite aufruft. Der Kontext, in dem dieser Code beim Benutzer ausgeführt wird, ist der Sicherheitskontext des **ASP**. Der

[10] siehe `http://www.securityfocus.com/archive/1/348368`

[11] z.B. `http://www.securityfocus.com/archive/1/378431`
`http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0841`
`http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0566`

platzierte Code stammt aber nicht vom **ASP** und kann dazu führen, Informationen vom Benutzer zum Angreifer zu übertragen. Eine clientseitige XSS-Attacke bedarf aber nicht zwingend der Möglichkeit, Inhalt in eine Seite beim Server zu platzieren. Es kann auch genügen den Benutzer mit seinem Webbrowser eine URL klicken zu lassen, in der die Adresse des Servers benutzt wird, der Querystring aber einen Codeteil vom Angreifer enthält. Wiederholt der Server anschließend den Codeteil des Angreifers innerhalb seiner Antwort an den Benutzer, so kann auch dieser Code im Kontext des **ASP** auf dem Benutzerbrowser ausgeführt werden.

Serverseitiges XSS versucht hingegen den Code auf dem Server ausführen zu lassen. Dabei sind dann z.B. „Include Statements“ interessant. Mit deren Hilfe wird versucht, dass das Skript, das von der URL auf dem Server angestoßen wurde, Codezeilen von einem anderen Rechner im Internet in seinen Ablauf einbindet.

XSS-Angriffe sind ein relativ neues Thema in der Sicherheitsbetrachtung von Internetsystemen. Sie lassen sich auf vielfältige Weise anwenden. Hier sollte nur das grundsätzliche Problem fremder Dateneingaben angesprochen werden, da diese Angriffe immer abhängig von der konkreten Implementierung sind. Dennoch kann festgehalten werden, dass ein **ASP** nicht erlauben sollte, dass fremder Inhalt in seinen Webseiten dargestellt wird.

Bei Liberty Alliance existiert für den Benutzer noch eine andere Schwachstelle auf der Anmeldeseite. Liberty Alliance bietet die Möglichkeit an, das Login mit einem „embedded-form“ auszuführen. Dabei wird die Anmeldeseite vom **SP** generiert, und nur die Anmeldedaten des Benutzers werden mittels eines HTTPS-Post Verfahren direkt an den **ASP** übermittelt. Das Risiko dabei ist, dass die Kontrolle über diese Anmeldeseite beim **SP** liegt, und er natürlich auch seine eigene Adresse für die Übertragung der Daten eintragen kann. Somit würde der **SP** an die Zugangsdaten des Benutzers für den **ASP** gelangen. Liberty Alliance ist sich dieser Problematik zwar bewusst, erlaubt aber dennoch die Verwendung solch einer Anmeldeseite.

> „Although users may like the seamlessness of this embedded form mechanism and deployers will like that the user does not leave their Website, it has serious policy and security considerations. In this me-

chanism, the user may be revealing his identity provider credentials to the service provider in cleartext. This is because the service provider controls the actual code implementing both the page and the embedded form and thus can conceivably capture users' credentials. In this way, privacy surrounding the user's identity provider account may be compromised by such a rogue service provider, who could then wield those credentials and impersonate the user. Because of this, when using authentication via embedded form, deployers may want to consider appropriate contract terms between identity providers and service providers to address this risk." [CHKT03][S.42]

4.6.3 Key for the Kingdom Problem

Der **ASP** bildet ein interessantes Angriffsziel. Kann sich ein Angreifer erfolgreich als ein Benutzer beim **ASP** anmelden, so kann er sich auch bei allen **SP**s anmelden. Je nachdem welche Profildaten zusätzlich beim **ASP** gespeichert sind, können die Konsequenzen für den Benutzer noch gravierender und weitreichender sein. Deswegen sollte der Zugang zum **ASP** besonders gut geschützt werden. Für Liberty Alliance kann keine Aussage dazu getroffen werden, da der Prozess des Anmeldens nicht Teil der Spezifikation ist. Deswegen wird hier nur Microsoft Passport untersucht.

Microsoft Passport nutzt als Anmeldedaten für den **ASP** eine Emailadresse und ein Passwort. Das Passwort muss mindestens sechs Zeichen lang sein. Dabei werden keine Sonderzeichen, Nummern oder Großbuchstaben verlangt. Jedes ordentliche Multiuserbetriebssystem verlangt höhere Anforderungen an das Benutzerpasswort und gerade ein **ASP** sollte sicherstellen, dass ein Benutzer Passwörter wählt, die nur schwer herauszubekommen sind. Schlecht gewählte Passwörter können leicht mit einer Wörterbuchattacke ermittelt werden, auch zu kurze Passwörter ohne Sonderzeichen können zu leicht aufgedeckt werden.

Mit den Standardeinstellungen der Anmeldemaske speichert der **ASP** die eingegebene Emailadresse in einem Cookie beim Benutzer. Dieser Cookie wird auch nicht bei einem Logout gelöscht. Microsoft Passport versucht, bei einem Besuch der Loginwebseite das Cookie auszulesen, um mit dem Wert automatisch

die Emailadresse im Anmeldeformular einzutragen. Ein Benutzer muss explizit das Speichern dieses Cookies unterbinden. Durch diese Standardeinstellung kann jeder andere Benutzer des Webbrowsers ohne Aufwand, die angemeldete Emailadresse erfahren. Dann benötigt er nur noch das richtige Passwort.

Hat ein Benutzer sein Passwort vergessen, bietet Microsoft Passport dem Benutzer die erneute Zusendung des Passworts an. Dafür muss der Benutzer drei aus fünf Fragen beantworten. Die Antworten hat der Benutzer schon einmal zuvor in seinem Profil beim **ASP** angegeben. Es handelt sich um allgemeine Fragen, wie „Augenfarbe", „Name der Mutter" oder „Lieblingstier". Bei diesen Frage-Antwort Systemen ist problematisch, dass bei einer wahrheitsgetreuen Beantwortung bei der Initialisierung diese Daten eventuell sehr leicht für einen Angreifer in Erfahrung zu bringen sind. Suchmaschinen und die private Homepage des Benutzers können dabei gute Anhaltspunkte für den Angreifer sein.

Generell sollte der Zugang zum **ASP** nicht mit Benutzer-ID und Passwort erfolgen. Gerade weil der **ASP** eine so sensible Stelle innerhalb eines Single Sign-On Systems ist, muss der Zugang zu diesem besonders gut geschützt werden. Idealerweise würden für die Authentikation des Benutzers beim **ASP** Verfahren verwendet, die nicht nur auf Wissen aufbauen. Es böten sich Smartcards oder Tokens wie SecureID oder auch biometrische Verfahren an.

Zusätzlich sollte ein Single Sign-On System für die Anmeldung eine definierte und möglichst einheitliche Schnittstelle bieten und nicht, wie bei Microsoft Passport, über viele unterschiedliche Wege funktionieren: Zu nennen wären die passport.com Webseite, die Hotmail Webseite, der MSN Messenger und sogar die Windows XP Anmeldung. Durch diese Vielfalt erhöhen sich zum einen die Angriffsgefahrenstellen und außerdem ist für den Benutzer nicht immer ersichtlich, wann er sich im Single Sign-On System anmeldet. Ein Benutzer könnte z.B. aufgrund eines fehlenden Internetszugangs seine Hotmail Accountdaten einem Bekannten geben, damit dieser stellvertretend Kontakt zu seinem Webmaildienst aufnimmt. Der Benutzer geht dann davon aus, dass er dem Bekannten nur die Zugangsdaten für seinen Hotmail Account aushändigt. Allerdings hat der Bekannte aber nun auch Zugriff auf den Passport Account des Benutzers.

4.6.4 Probleme bei SSL/TLS

Wie im Abschnitt über die Man in the Middle Angriffe aufgezeigt wurde, existieren für einen Angreifer eine Vielzahl von unterschiedlichen Verfahren, um Einfluss auf die Kommunikation zu nehmen. Mit der Protokollschicht TLS/SSL soll dieser Umstand verbessert werden. SSL/TLS setzt dabei auf TCP auf und versucht einen gesicherten Host-zu-Host Kommunikationskanal zu realisieren. SSL/TLS findet schon lange im Internet Verwendung und wurde schon vielfach Sicherheitsanalysen unterzogen. Dennoch existieren einige Risikien bei der Benutzung von SSL/TLS. Diese sollen nachfolgend erläutert werden.

Sicherheitsmängel bei SSLv2:

Grundsätzlich sollte die Version 2 von SSL nicht mehr benutzt werden, da diese Version gravierende Sicherheitsmängel aufweist.

- **Heruntersetzen der Sicherheitsparameter für einen SSL-Kanal:** Ein Angreifer kann während der Handshakephase die Ciphersuite mittels einer Man in the Middle Attacke modifizieren. Dieser Angriff hat massive Auswirkung auf die Güte des gesicherten SSL/TLS-Kanals. Ein Angreifer kann erreichen, dass sich Client und Server bei der Aushandlung der zu verwendenden Ciphersuite auf eine mit schwacher Verschlüsselung und schwachem Integritätsschutz einigen, obwohl beide, Client und Server, sicherere Verfahren unterstützen würden. Anschließend kann der Angreifer den geheimen Schlüssel innerhalb kürzester Zeit mittels einer Bruteforce Attacke in Erfahrung bringen. Dieser Angriff funktioniert auch, wenn sich beide Parteien mit Zertifikaten authentisieren.

- **Beeinflussung der Paddinglänge:** Ein Angreifer kann Einfluss auf die Paddinglänge der übertragenen Nachrichten nehmen. Dazu braucht er nur die Paddinglängenangabe zu verändern. Somit kann der Angreifer einzelne Bytes am Ende von Nachrichtenblöcken für den Empfänger entfernen. Kennt der Angreifer die Struktur der übertragenen Nachrichten, so könnte er diese Attacke ausnutzen, um eventuell die Semantik der Nachrichten zu verändern.

- **Benutzung gleicher Schlüssel bei der Verschlüsselung und MAC-Berechnung:** SSLv2 nutzt die gleichen Schlüssel für die Verschlüsselung der Daten und für die Berechnung des „Message Authentication Code“ (MAC). Dadurch können keine längeren Schlüssel für den MAC benutzt werden, sollten nur kurze Schlüssellängen für die Verschlüsselung erlaubt sein. Dieser Umstand betrifft vor allem Exportbeschränkungen und Länder, in denen nur eine bestimmte Schlüssellänge für eine Verschlüsselung erlaubt ist.

- **Berechnung des MAC nur mit MD5:** Dieser Algorithmus ist für die Berechnung von MAC nicht die beste Wahl. Zwar wurden schon 1994 von Bert den Boer und Antoon Bosselaers sogenannte Pseudokollisionen endeckt und 1996 veröffentlichte Hans Dobbertin weitere Ansatzpunkte für einen Kollisionsangriff [Dob96a] [Dob96b], dennoch ist bis heute nicht bekannt, ob zu einer vorgegeben Nachricht eine andere Nachricht mit gleichem MAC-Wert in angemessener Zeit gefunden werden kann. Die Diskussion über die Sicherheit von MD5 ist aber erneut entflammt, da ein chinesisches Wissenschaftlerteam im August 2004 eine Kollision in der vollständigen MD5-Funktion gefunden hatte und darauf aufbauend innerhalb kürzester Zeit weitere Kollisionen berechnen konnte [WFLY04]. Wie genau die Chinesen dabei vorgegangen sind, ist noch nicht veröffentlicht worden, ihre Ergebnisse wurden jedoch bereits von unabhängiger Stelle bekräftigt.

- **Initiierung einer einseitigen Beendigung der Verbindung:** Da SSLv2 keine eigenen Protokollschritte vorsieht, um eine aktive SSL-Sitzung zu beenden, kann ein Angreifer durch das Schicken einer Nachricht, die Sitzung bei einer der beiden Parteien terminieren. Die andere Seite erfährt von dieser Terminierung nichts und hält den Tunnel weiterhin aufrecht. Dieser Angriff kann für eine DoS-Attacke missbraucht werden. Ein Angreifer könnte aber auch versuchen, die Sitzung zu übernehmen, wenn er der aktiven Seite weiterhin glaubhaft machen kann, dass er der ursprüngliche Kommunikationspartner sei.

Aus den eben genannten Gründen sollte SSLv2 nicht mehr benutzt werden. Liberty Alliance schreibt deshalb in ihrer Spezifikation vor, dass bei einer Verwen-

dung von SSL die Version 3 eingesetzt werden sollte. Microsoft Passport macht keine genaueren Angaben, welche Version von SSL genutzt werden soll. Eine von mir durchgeführte Abfrage des **ASP** (passport.com) bei Microsoft Passport ergab, dass folgende SSLv2-Verfahren unterstützt werden:

DES-CBC3-MD5 - 168 bits
RC2-CBC-MD5 - 128 bits
RC4-MD5 - 128 bits
DES-CBC-MD5 - 56 bits (!)
EXP-RC2-CBC-MD5 - 40 bits (!)
EXP-RC4-MD5 - 40 bits (!)

Verfahren die mit einem Ausrufezeichen gekennzeichnet sind, gelten als nicht sicher gegen Bruteforce Attacken.

Allgemeine Sicherheitsprobleme bei SSL/TLS

Schwachstelle Benutzer: Der Benutzer ist bei den Sicherheitsbetrachtungen zu SSL/TLS ein wichtiger Faktor. Es obliegt nämlich dem Benutzer, eine falsche Gegenstelle zu erkennen. Zwar hilft der Webbrowser dem Benutzer durch angezeigte Informationen, eine Entscheidung zu treffen, letzendlich entscheidet aber der Benutzer, ob er mit der Gegenstelle kommunizieren will. Kann ein Angreifer den Benutzer Glauben machen, dass er (angeblich) mit der richtigen Gegenstelle kommuniziert, so kann er an die sensiblen Logininformationen gelangen.

Es gibt mehrere Verfahren der Ausführung eines solchen Angriffs. Gehen wir davon aus, dass der Angreifer den Benutzer zu seinem Server, anstatt zum **ASP**, lenken kann. Im einfachsten Fall baut der Angreifer einfach die Webseite des **ASP** nach und stellt die Verbindung nur über HTTP her. Der Benutzer würde dies aber sehr wahrscheinlich an mehreren Faktoren erkennen. Er könnte die URL in der Locationbar des Webbrowsers überprüfen und würde das fehlende Symbol für eine SSL/TLS-Verbindung bemerken. Deswegen wird ein Angreifer versuchen, diese Felder zu manipulieren.

Nutzt der Angreifer einen ähnlichen Namen für seine Domäne wie die des **ASP**, so kann ein unaufmerksamer Benutzer leicht den Unterschied übersehen. Dabei bieten sich viele Möglichkeiten für diese gefälschte URL an. Die folgenden Beispiele sollen aufzeigen, wie ähnliche Namen zu „www.adlerbank.com“ aussehen können.

Täuschungstechnik	Beispiel
Bindestriche	www.adler-bank.com
Andere TLD verwenden	www.adler-bank.com.au
Legitime Erweiterungen	www.secure-adlerbank.com
Wörter vertauschen	www.bankadler.com
Subdomänen	www.adler.bank.com
Ähnliche Zeichen	www.ad1erbank.com
Orthografische Fehler	www.alderbank.com

Der Kreativität des Angreifers sind hierbei keine Grenzen gesetzt. Erhält der Angreifer für seine Domäne noch ein gültiges Zertifikat bei einer Zertifizierungsstelle und ist diese Zertifizierungsstelle schon im Webbrowser eingetragen, dann hat er gute Chancen, dass die Täuschung dem Benutzer nicht auffällt.

Der Angreifer kann außerdem ein eigenes Zertifikat für seinen Server erzeugen. Aktuelle Webbrowser melden dann aber, dass sie das selbsterzeugte Serverzertifikat nicht verifizieren können, da die übergeordnete Zertifizierungsstelle dem Webbrowser nicht bekannt ist. Die Wahrscheinlichkeit, dass der Benutzer diese Warnung ignoriert, ist groß, da sehr viele Internetbenutzer nicht genau wissen, worauf sie bei einem Zertifikat achten müssen. Oft genügt es ihnen, wenn der Webbrowser anzeigt, dass die Daten über einen gesicherten Kanal übertragen werden. Um diese Warnmeldung beim Benutzerwebbrowser zu unterbinden, muss der Angreifer sein Zertifizierungsstellen Zertifikat im Webbrowser unterbringen. Dazu böten sich wieder verschiedene Verfahren an. Er könnte z.B. ein signiertes Java Spiel anbieten und dabei gleichzeitig dem Benutzer dieses Zertifikat der Zertifizierungsstelle übertragen.

Hat es der Angreifer mit einem sehr aufmerksamen Benutzer zu tun, so kann er versuchen, die Angaben im Webbrowser zu manipulieren. Allein im Jahre 2004 wurden monatlich neue Schwachstellen bei populären Webbrowsern bekannt, die zu solchen Zwecken ausgenutzt werden konnten[12]. Anhand dieser Schwachstellen konnten andere URLs im Webbrowser angezeigt werden, als die, die der Webbrowser in Wirklichkeit kontaktiert (Adresszeilen Spoofing oder URL-Spoofing), oder der Webbrowser zeigte eine gesicherte Verbindungen an, obwohl diese nicht

[12]Eine Auswahl solcher Schwachstellen findet sich im Anhang dieser Arbeit.

gesichert war.

Auch wenn diese Schwachstellen nicht vorhanden wären, brauchen „Publickey Infrastrukturen“ (PKI) einen aufgeklärten und informierten Benutzer. Dies ist aber häufig nicht gegeben, vor allem wenn sich mit der Zeit ein Automatismus beim Benutzer einstellt und er nicht mehr sehr sorgfältig den Inhalt von URLs oder Zertifkaten betrachtet, bietet sich die Möglichkeit der Manipulation. Manche Webtechniken gewöhnen den Benutzer sogar an falsche Zertifikate. Hier seien gerade Proxy-/Cachesysteme genannt, die Webinhalte zwischenspeichern. Erhält so ein Dienst eine HTTPS-Anfrage, so muss er sein eigenes Zertifikat präsentieren, da die Anfrage ja von ihm beantwortet wird und nicht vom ursprünglichen Server.

Ein anderes Problem mit Zertifikaten und aktuellen Webbrowsern ist die Verwaltung von „Certificate Revocation Lists“ (CRLs). Diese Listen geben Auskunft, welche Zertifikate nicht mehr gültig sind. Kein aktueller Browser aktualisiert automatisch solche Listen, und fast keinem Internetbenutzer sind solche Listen überhaupt bekannt. Diese Listen sind aber zwingend notwendig, um ungültige oder gestohlene Zertifikate zu erkennen.

Schwache Schlüssel: Sowohl SSLv3, als auch TLSv1 enthalten zu schwache Kryptoverfahren. Obwohl diese Verfahren auf dem Benutzerbrowser wie sichere Kommunikationsverbindungen erscheinen, können sie nicht als sicher angesehen werden. Entweder bieten Sie keine Teilnehmerauthentikation an oder aber die Verschlüsselung lässt sich zu leicht mit einer Bruteforce Attacke „knacken“.

Dazu gehören bei SSLv3 die Verfahren:

SSL_RSA_WITH_NULL_MD5 SSL_RSA_WITH_NULL_SHA
SSL_RSA_EXPORT_WITH_RC4_MD5
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_anon_EXPORT_WITH_RD4_40_MD5
SSL_DH_anon_WITH_RC4_128_MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_anon_WITH_DES_CBC_SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA

und bei TLSv1:

TLS_RSA_WITH_NULL_MD5 TLS_RSA_WITH_NULL_SHA
TLS_RSA_EXPORT_WITH_RC4_MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_anon_EXPORT_WITH_RD4_40_MD5
TLS_DH_anon_WITH_RC4_128_MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_anon_WITH_DES_CBC_SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA

Ein Angreifer kann bei SSLv3/TLSv1 nicht mehr das zu benutzende Kryptoverfahren bei einem Tunnelaufbau beeinflussen, und in der Regel sollten Client und Server das höchste gemeinsame Verfahren auswählen. Es besteht aber die Gefahr, dass mobile Geräte wie Mobiltelefone oder PDAs nur schwache SSL-Verfahren implementieren, da sie nicht genug Rechenleistung besitzen. Aus diesem Grund schreibt Liberty Alliance auch zwingend die Implementation von

TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

vor. Microsoft Passport macht wie schon zuvor erwähnt, keine weiteren Angaben zu der Benutzung von SSL/TLS.

4.6.5 Phishing Angriffe

Wie oben aufgezeigt wurde, existieren eine Fülle von Methoden, um einem Benutzer einen Server vorzutäuschen. Außer durch Eingriffe in die Infrastruktur, um einen Benutzer auf den gefälschten Server zu lenken, werden immer häufiger auch Phishing Angriffe benutzt. Der Vorteil von Phishing Angiffen ist, dass der Angreifer überhaupt keine Veränderung an der Infrastruktur und den Kommunikationswegen durchführen muss. Beim Phishing wird versucht, einen Benutzer durch gefälschte Emails oder andere Tricks dazu zu bringen, eine gefälschte Website zu besuchen und dort an die sensiblen Informationen zu gelangen.

Häufig werden dabei Massenmails vom Angreifer versendet. In diesen Mails wird der Eindruck erweckt, dass sie von einer vertrauenswürdigen Stelle stammen. Der Inhalt der Mail versucht den Empfänger zu überzeugen, die angegebene Webseite zu besuchen und seine sensiblen Daten in ein Webfomular einzugeben. Absender und die Zielwebseite haben dabei meistens gefälschte Namen oder Bezeichnungen, die sehr ähnlich Aussehen wie die offizielle Seite der Firma, auf die Bezug genommen wird. Die Zielseite beim Angreifer mit dem Webformular für die Dateneingabe hat das gleiche Aussehen wie die Originalseiten. Dadurch sind sie nur sehr schwer als Fälschungen identifizierbar.

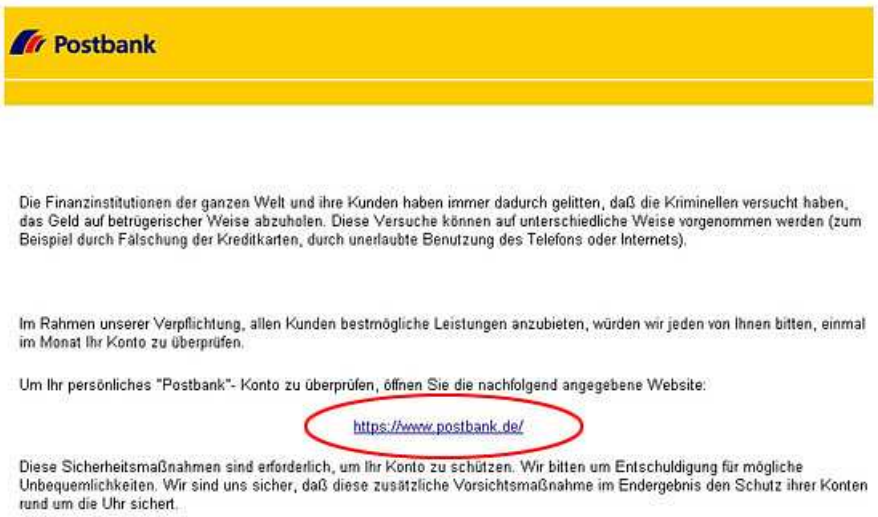

Abbildung 4.10: Ausschnitt einer realen Phishing Email

Die markierte URL in der Email beschreibt zwar die richtige Adresse der Postbank, der Benutzer wurde aber auf www.postbank.info geleitet, wenn er den Link klickte.

Die Anzahl von solchen Phishing Angriffen nimmt rasant zu. Laut MessageLabs[13] lag die Anzahl von identifizierten Phishing Mails im September 2003 noch bei 279 Stück. Im September 2004 wuchs die Anzahl dann rasant auf über 2 Millionen. Ingesamt will MessageLabs im Jahre 2004 über 18 Millionen solcher Phishing Emails identifiziert haben [Mes04]. Dabei sollen die Angreifer immer

[13] `http://www.messagelabs.com/home/default.asp`

raffiniertere Tricks benutzen, um den Benutzer zu täuschen. Bestimmte Phishing Angriffe fügen einfach in die Hosts Datei[14] gefälschte Daten ein und wenn der Benutzer beim nächsten Besuch der Webseite den Browser aktiviert, wird er in Wirklichkeit zum Server des Angreifers geleitet.

Auch die „Anti-Phishing Working Group“[15] bestätigt in ihren Reports einen massiven Anstieg von Phishing Emails und gefälschten Webseiten im Internet.

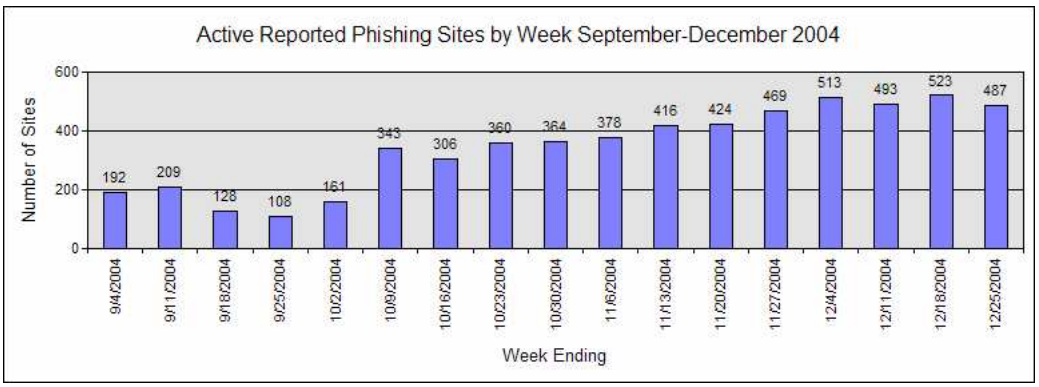

Abbildung 4.11: Anzahl gefälschter Websites für Phishing Angriffe Quelle:[Ant04]

Es bleibt zu befürchten, dass solche Angriffe auch in Zukunft vermehrt auftreten werden. Anhand der Versendung von Massenemails lassen sich sehr leicht eine Vielzahl von Benutzern erreichen und die Wahrscheinlichkeit, dass einzelne Benutzer auf diesen Angriff hereinfallen, ist sehr groß.

Kurz vor Abschluss dieser Diplomarbeit tauchte im Zusammenhang mit Phishing ein neuer Angriff mit dem Namen Pharming auf. Auch Pharming versucht, an die sensiblen Informationen eines Benutzers zu gelangen. Jedoch wird nicht jeder einzelne Benutzer seperat „geködert“, sondern es werden Eingriffe an dem DNS-System unternommen, wie beim DNS-Poisoning in Kapitel 4.5.1 beschrieben. Somit lassen sich wesentlich mehr Benutzer zu dem gefälschten Server umleiten als bei Phishing, da auch die Benutzer umgeleitet werden, die die zum richtigen Dienstleister gehörende URL mit ihrem Webbrowser verwenden.

[14] In der Hosts Datei auf einem Rechner werden DNS Namen auf IP Adressen gemappt.
[15] `http://www.antiphishing.org/index.html`

4.7 Kontrolle der persönlichen Daten

Damit Single Sign-On Systeme und insbesondere Identity Managemant Systeme bei den Benutzern eine große Akzeptanz finden, sollten solche Systeme nicht nur den technischen Aspekt der Übertragung von Daten umfassen, sondern auch einen gewissenhaften Umgang mit diesen Daten beinhalten. Der Benutzer vertraut seine digitale Identität einer Instanz im Internet an und hinterlegt dort eventuell eine große Anzahl von Profildaten. Meist werden die Kommunikationssysteme gegen externe Angreifer geschützt, der Schutz gegen potentielle interne Angeifer, wie beispielsweise Systemadministratoren, ist jedoch häufig gering. Unberücksichtigt bleiben dabei die Schutzinteressen der Anwender.

4.7.1 Datenschutzgesetze

Zum einen existieren für diese Zwecke die Datenschutzgesetze. Diese sollen die Rechte des Benutzers stärken, welche Daten erhoben, bearbeitet und gespeichert werden dürfen. Auf der Webseite des „Bundesbeauftragten für den Datenschutz[16]“ wird das Bundesdatenschutzgesetz folgendermaßen umschrieben:

> Das Bundesdatenschutzgesetz (BDSG) gibt dem einzelnen Bürger verschiedene Möglichkeiten, den Umgang mit seinen personenbezogenen Daten durch Auskunft und Benachrichtigung selbst zu überprüfen und durch Berichtigung, Löschung und Sperrung von Daten zu beeinflussen.

Neben dem Bundesdatenschutzgesetz gibt es noch weitere Gesetze, die auch den Datenschutz umfassen:

- Die Landesdatenschutzgesetze (LDSG), die bei der Verarbeitung personenbezogener Daten durch öffentliche Stellen das Recht auf informationelle Selbstbestimmung wahren sollen.
- Das Teledienstedatenschutzgesetz (TDDSG), das als Schutz personenbezogener Daten bei Nutzung von Telediensten dienen soll.

[16] `http://www.bfd.bund.de/index.html`

- Der Staatsvertrag über Mediendienste, der eine einheitliche Rahmenbedingungen für die verschiedenen Nutzungsmöglichkeiten von elektronischen Informations- und Kommunikationsdiensten schafft.

- Die Telekommunikations Datenschutzverordnung (TDSV), die den Schutz personenbezogener Daten der an der Telekommunikation beteiligten Unternehmen, die geschäftsmäßig Telekommunikationsdienste erbringen oder an deren Erbringung mitwirken, sicherstellt.

Daneben existieren noch weitere Gesetze für spezielle Bereiche, die dem Datenschutz dienen. U.a. wären dies der Schutz von Sozialdaten, die ärztliche Schweigepflicht und das Steuergeheimnis. Zu erwähnen sei auch, dass der Datenschutz durch das Volkszählungsurteil des Bundesverfassungsgerichts (15. Dezember 1983) mit dem Recht auf informationelle Selbstbestimmung Verfassungsrang erhielt.

Diese Gesetze gelten aber nur für Deutschland. Mit der „Richtlinie 95/46/EG zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr“ und der „Richtlinie 2002/58/EG über die Verarbeitung personenbezogener Daten und den Schutz der Privatsphäre in der elektronischen Kommunikation“ gibt es Ansätze, die unterschiedlichen Datenschutzbestimmungen innerhalb der Europäischen Union zu harmonisieren. Es exisitieren aber keine weltweit gültigen Gesetze oder Vereinbarungen die den Datenschutz festschreiben.

Dies ist ein großes Problem bei der Benutzung des Internets. Es ist nämlich für einen Benutzer nicht ersichtlich, wo auf der Welt der Server steht, zu dem er seine Daten übermittelt. Domänennamen geben keine Auskunft darüber, in welchem Land der Server betrieben wird. Zudem gibt es auch keine weltweiten Vereinbarungen oder Standards, wie Profildaten verwaltet, abgespeichert und gesichert werden müssen.

Microsoft Passport und Liberty Alliance weisen zwar auf eine Notwendigkeit von Datenschutz hin, um die Akzeptanz des Benutzers für diese Systeme zu gewinnen, es werden aber keine genauen Vorgaben gemacht und beide verweisen darauf, dass der Inhalt dieser Vereinbarungen Teil der Aushandlung zwischen den Teilnehmern ist. Es wird zusätzlich erwähnt, dass diese Vereinbarungen zwar

für den Benutzer ersichtlich sein sollen, aber keine Hilfsmittel gefordert oder angeboten, wie ein Benutzer diese zahlreichen, zum Teil sehr wahrscheinlich unterschiedlichen Angaben überschauen und vergleichen kann.

4.7.2 Die Notwendigkeit von Vertrauen

Bei den vorgestellten Single Sign-On Systemen spielt das Vertrauen der Teilnehmer untereinander eine sehr große Rolle. Innerhalb dieser Vertrauensbeziehungen nimmt der **ASP** eine herausragende Rolle ein. Alle Teilnehmer müssen dem **ASP** vertrauen. Der **ASP** verwaltet die Benutzerkennungen und optional die Profildaten der Benutzer. Einem betrügenden **ASP** steht das ganze System zur Verfügung. Er kann sich bei jedem **SP** als Benutzer einloggen und Missbrauch betreiben. Die **SP** vertrauen nämlich auf die Statusinformation des Benutzers beim **ASP** ohne eigene Verifikationen auszuführen.

Auswahl des ASP

Vor allem aber muss der Benutzer Vertrauen in das System und dessen Teilnehmer haben. Dies betrifft zunächst den **ASP**. Hat der Benutzer wie bei Microsoft Passport keine Auswahl zwischen unterschiedlichen **ASP**s, so werden seine gesamten Profildaten an einer Stelle gesammelt und verwaltet. Dies führt zu einer sehr großen Datensammlung und der Benutzer muss dem **ASP** sehr viel Vertrauen entgegenbringen, dass dieser verantwortungbewusst und im Sinne des Benutzers damit umgeht.

Das Liberty Alliance Modell beinhaltet die Nutzung von unterschiedlichen **ASP**s für den Benutzer. Dadurch hat der Benutzer die Möglichkeit, seine Profildaten über mehrere **ASP**s zu streuen und somit einer zentralen Datenhaltung entgegenzuwirken. Dies setzt natürlich voraus, das innerhalb des Circle-of-Trust auch mehrere organisatorisch unabhängige **ASP**s agieren. Der Datenschutz wird offensichtlich nicht verbessert, wenn innerhalb des Circle-of-Trust mehrere **ASP**s vorhanden sind, diese aber im Endeffekt zu einer Organisationsstruktur oder einem Firmenverbund gehören. Diese Gefahr ist vorallem deswegen gegeben, da davon auszugehen ist, dass sich zunächst viele kleine unterschiedliche Circle-of-Trust formieren werden.

Kontrolle der Profildaten

Ein Benutzer muss die Möglichkeit haben, seine Profildaten sehr granular freizuschalten. Bei Microsoft Passport hat der Benutzer nur die Auswahl zwischen keiner Freigabe seiner Profildaten für den anfragenden **SP** oder alle eingetragenen Profildaten freizugeben. Somit hat der Benutzer bei Microsoft Passport keinen Einfluss darauf, welche der angegebenen Profildaten beim **ASP** zu einem **SP** übertragen werden dürfen. Liberty Alliance hat bisher noch keine Spezifizierung für die Profilübertragung zwischen **ASP** und **SP** vorgelegt. Daher kann bei Liberty Alliance keine Aussage über die Profilverwaltung für den Benutzer getroffen werden. Hinweise in den bisher veröffentlichten Dokumenten scheinen aber eine höhere Kontrolle des Benutzers bezüglich der Profilübertragung zu ermöglichen. Dennoch ist zu diesem Zeitpunkt nicht erkennbar, ob diese Funktionalität für einen **ASP** bindend sein muss.

Ein Benutzer sollte zumindest für jeden einzelnen **SP** angeben können, welche Profildaten übertragen werden dürfen. Noch besser wäre es, wenn der Benutzer sogar eine Unterscheidung je Ressource beim **SP** treffen könnte. Dazu sind dem Benutzer Auswahlmasken vom **ASP** anzubieten. Diese Auswahlmasken sollten für jeden unterschiedlichen **SP** einstellbar sein. Es muss einem Benutzer auch jederzeit die Möglichkeit geboten werden, diese Einstellungen je **SP** zu verändern. Zusätzlich wäre es wünschenswert, einstellen zu können, ob die angegebenen Einstellungen zeitlich unbefristet gelten sollen oder nur für die aktuelle Single Sign-On Sitzung gültig sind.

Kontrolle der Kommunikationswege

Damit die Persönlichkeitsrechte des Benutzers gewahrt bleiben, sollte der Benutzer jederzeit die Kontrolle über die Übertragung seiner Daten haben. Dies ist aber bei den vorgestellten Protokollen nicht gegeben. Das liegt vorallem daran, dass beide Systeme, nach einer einmaligen Erlaubnis des Benutzers zur Datenübertragung, eine sogenannte „stille Authentikation“ inklusive anschließender Profildatenübertragung erlauben. Dies bedeutet, dass nachdem der Benutzer dem **ASP** die Erlaubnis erteilt hat, einem **SP** seine Daten übermitteln zu dürfen, dieser **SP** anschließend unbemerkt für den Benutzer die Profildaten beim **ASP** anfordern

kann. Ist der Benutzer während dieses Anfragezeitpunkts beim **ASP** angemeldet, so findet der Datenaustausch zwischen **ASP** und **SP** nicht notwendigerweise sichtbar für den Benutzer statt.

Um das Vertrauen des Benutzers in solche Systeme zu erhöhen, sollten folgende Voraussetzungen zwingend in diesen Systemen umgesetzt werden.

- Kein Informationsgewinn bei initialen Anfragen: Wenn ein **SP** eine intitiale Anfrage beim **ASP** stellt, um eine Erlaubnis für die Übertragung des Anmeldestatus oder der Profildaten zu erhalten, dürfen beide Teilnehmer keine Informationen über den Benutzer erhalten. Dies darf erst nach der Einwilligung des Benutzers erfolgen. Für einen **ASP** sollte nicht erkennbar sein, welche Ressourcen der Benutzer beim **SP** nutzen möchte und der **SP** darf keine Informationen über den Benutzer erhalten. Der **SP** sollte nicht erkennen können, ob der Benutzer überhaupt ein Konto bei dem **ASP** führt und ob der Benutzer z.Zt. beim **ASP** angemeldet ist. Solche Informationen dürfen, wenn überhaupt, nur dann ersichtlich sein, wenn der Benutzer seine Einwilligung für eine Kommunikation zwischen dem **ASP** und dem **SP** gegeben hat.

- Authentikationsanfragen nur bei Bedarf: Ein **SP** sollte den Status einer Authentikation nur dann beim **ASP** abfragen, wenn es für die Bearbeitung der angeforderten Ressourcen wirklich notwendig ist. Erfordern die vom Benutzer beim **SP** angeforderten Ressourcen keine Identifizierung oder Authentikation, so sollte ein **SP** nicht den **ASP** kontaktieren.

- Profildatenübertragung sparsam einsetzen: Wie bei Authentikationsanfragen sollte eine Profildatenübermittlung nur zu dem Zeitpunkt erfolgen, wenn es für den weiteren Verlauf notwendig ist. Zusätzlich sollte ein **SP** nur solche Profildaten beim **ASP** abfragen, die für Bearbeitung der aktuellen Aufgabe gebraucht werden. Es sollten nicht automatisch vom **ASP** immer alle freigegebenen Benutzerinformationen an einen anfragenden **SP** übermittelt werden, selbst wenn der Benutzer zuvor einer Übertragung zu diesem **SP** zugestimmt hat.

- Datenaustausch muss für den Benutzer erkennbar sein: Findet ein Austausch

von Informationen zwischen einem **SP** und einem **ASP** statt, so muss dieser Informationsaustausch für den Benutzer offensichtlich sein. Dabei sollte der Benutzer nicht nur den Kommunikationswunsch erkennen können, sondern auch die zu übertragenden Daten einsehen können. Da der Umfang der Daten von Anfrage zu Anfrage variieren kann, sollte dies nicht einmalig vom Benutzer einstellbar sein, sondern individuell erfolgen. Es böte sich an, dass vor der Übertragung der Profildaten vom **ASP** zum **SP**, der **ASP** dem Benutzer eine Webseite anbietet, in der die für anfragende **SP** erlaubten Profilinformationen aufgelistet werden. Der Benutzer kann innerhalb der Webseite bestimmte Profilfelder für die Übertragung abwählen, und erst nach seinem Einverständnis würden diese Daten vom **ASP** zum **SP** übermittelt.

- Auditmöglichkeit für den Benutzer: Ein **ASP** sollte einem Benutzer jederzeit die Möglichkeit bieten, den Status seiner Sitzung einzusehen. Dazu gehört der aktuelle Authentikationsstatus des Benutzers beim **ASP**, eine Historie, wann der Benutzer beim **ASP** angemeldet war, zu welchem Zeitpunkt ein Informationsaustausch mit welchen **SP** vollzogen wurde und welche Benutzerdaten dabei übertragen wurden. Bei einem **SP** sollte ein Benutzer jederzeit einsehen können, welcher **ASP** von diesem **SP** benutzt wird und wie der derzeitige Authentikationskontext des Benutzers beim **SP** ist.

Die eben vorgestellten Forderungen können nicht das Vertrauensproblem lösen, helfen aber dem Benutzer, einen besseren Überblick zu behalten. Dennoch sind diese Maßnahmen keine Gewähr gegen einen Missbrauch, da der Benutzer weiterhin darauf vertrauen muss, dass die angezeigten Informationen auch stimmen.

Lebensdauer der Benutzerdaten

Beide untersuchten Systeme machen keine Angaben, was mit den übertragenen Daten passieren soll und wie lange deren Lebensdauer bei dem jeweiligen Teilnehmer sind. Hierbei stellen sich vorallem folgende Fragen.

- Wann werden Benutzerdaten beim **SP** gelöscht und wie kann dieses gewährleistet werden. Hat z.B. ein Benutzer einen Kauf bei einem **SP** getätigt und benötigt dieser **SP** für die Übersendung der Ware die Postanschrift des Benutzers, so bräuchten diese Daten nach dem Abschicken des Pakets nicht länger beim **SP** gespeichert zu werden. Bei Bedarf könnte der **SP** jederzeit diese Informationen beim **ASP** erneut abrufen.

- Was passiert mit den Daten, wenn ein Benutzer eine bestimmte Verknüpfung zwischen einem **SP** und einem **ASP** auflöst? Wie kann gewährleistet werden, dass die bisher angefallenen Daten beim **SP** gelöscht werden?

- Was passiert mit den Benutzerdaten bei einem **ASP**, wenn der Benutzer sein Konto bei diesem auflöst? Wie lange sind Benutzerinformationen bei einem **ASP** gültig, wenn ein Benutzer diesen **ASP** nicht mehr benutzt?

4.7.3 Informationelle Selbstbestimmung durch Identitätsmanagement

Soll die „reale Welt“ in diesen Systemen abgebildet werden, so brauchen diese Systeme eine feinere Unterteilung der Profildaten. Gerade wenn diese Systeme die herkömmlichen Informationswege und Geschäftsmodelle abbilden wollen, bedarf es der Möglichkeit einer pseudonymen oder anonymen Nutzung. Zusätzlich sollte dem Benutzer eine Möglichkeit geboten werden, unterschiedliche Rollen innerhalb des Systems annehmen zu können.

Pseudonyme oder anonyme Nutzung

Im RFC „Internet Security Glossary“ wird eine anonyme Nutzung von Internetdienstleistung folgendermaßen definiert:

> „An access control feature (or, rather, an access control weakness) in many Internet hosts that enables users to gain access to general-purpose or public services and resources on a host (such as allowing any user to transfer data using File Transfer Protocol) without having a pre-established, user-specific account (i.e., user name and secret password).“ [Shi00]

Beispiel Anonymität: Der Einkauf einer Ware in einem Kaufhaus. Hierbei kann der Benutzer die Ware bar bezahlen und bleibt somit anonym für das Kaufhaus.

Eine anonyme Nutzung ist gegeben, wenn kein Teilnehmer auf die Identität des Benutzers schließen kann. Dies ist bei der Nutzung der vorgestellten Systeme nicht möglich und ist auch nicht sinnvoll. Gerade der **ASP** bedarf der Identifizierung des Benutzers, um abhängig von dessen Einstellungen agieren zu können. Ob aber ein Benutzer weiterhin die frei zugänglichen Ressourcen eines **SP**s benutzen kann, ohne dass der **SP** gleich eine Anfrage bei einem **ASP** stellt und somit doch die Identität des Anfragenden erfahren kann, wird bei beiden Systemen nicht behandelt. Dies wäre ein Verlust der Anonymität des Benutzers. Hat der Benutzer nämlich zuvor bei der Benutzung einer bestimmten Ressource beim **SP** sein Einverständnis gegeben, dass der **ASP** bestimmte Informationen an den **SP** übertragen kann, so kann der **SP** auch bei frei zugänglichen Ressourcen eine Anfrage an den **ASP** stellen.

Anonymität und Identität bilden gewissermaßen die beiden Pole, zwischen denen sich die verschiedenen Stufen der Pseudonymität bewegen.

> Pseudonyme werden zum Schutz der Person vergeben, deren Identität nicht beliebigen Dritten bekannt werden darf, deren Identität über das Pseudonym jedoch einem bestimmtem Personenkreis bekannt sein muss. [Kel01]

Beispiel Pseudonymität: Um eine Ware empfangen zu können, gibt man dem Sender ein Postfach an. Der Sender der Ware hat zwar eine eindeutige Empfängeradresse, an die er die Ware schicken soll, er kann aber keinerlei Hinweise auf die Identität des Empfängers erfahren.

Sowohl Microsoft Passport als auch Liberty Alliance sehen die Möglichkeit vor, ihre Systeme auch für die Webseitenpersonalisierung beim **SP** zu gebrauchen. Dabei muss häufig nicht die Identität des Benutzers offengelegt werden. Es genügt, dass der **SP** nur einzelne Informationen des Benutzers erhält, ohne damit auf dessen Identität schließen zu können. Als ein Beispiel sei hier der Wetterdienst

genannt. Um dem Benutzer die aktuellen Wetterdaten liefern zu können, reicht es vollkommen aus, dem **SP** nur den Wohnort des Benutzers zu nennen.

Bei Microsoft Passport wird der Benutzer bei jedem teilnehmenden **SP** immer mit der gleichen PUID identifiziert. Diese Kennung ist für das ganze System eindeutig. Dieser PUID wird bei jeder Kommunikation zwischen **ASP** und **SP** übertragen. Somit bietet Microsoft Passport den Benutzern keine Möglichkeit, das System in pseudonymisierter Form zu benutzen.

Liberty Alliance sieht eine Pseudonymisierung des Benutzers vor. Zunächst einmal hat der Benutzer bei jedem **SP** eine andere Kennung, somit können die **SP**s nicht anhand der eigenen Benutzerkennung den Benutzer bei einem anderen **SP** identifizieren. Zusätzlich bietet Liberty Alliance dem Benutzer die Möglichkeit, einen Dienst bei einem **SP** unter einem Pseudonym zu benutzen. Dazu generiert der **ASP** bei jeder Profilanfrage von einem **SP** eine neue zufällige Benutzerkennung und überträgt diese mitsamt der freigegeben Profilinformationen an den **SP**. Der **SP** kann diese erhaltene Benutzerkennung nicht einer bestimmten Person zuordnen, hat aber eventuell genug Profilinformationen erhalten, um dem Benutzer den angeforderten Dienst anzubieten. Natürlich muss darauf hingewiesen werden, dass je nach Umfang und Art der übertragenen Profildaten dennoch eine Zuweisung zu einer Identität beim **SP** erfolgen könnte.

Rollenbasierte Profile

Rollenbasierte Profile sind dann gegeben, wenn der Benutzer mehrfache Angaben in seinem Profildatensatz machen kann. Diese mehrfachen Angaben ordnet er zusätzlich bestimmten Rollen zu. Unterschiedliche Rollen des Benutzers könnten seine Rolle im Berufsleben sein, in seiner Familie und in seiner Freizeit (Vereinsmitglied). Je nachdem, in welchem Kontext er einen Dienst eines **SP**s benutzt, sollte er dabei vor einer Übertragung seiner freigegeben Profildaten die Wahl haben, die Rolle auszuwählen.

Microsoft Passport sieht keine Rollen eines Benutzers vor. Es lassen sich keine Mehrfachangaben machen. Bei Liberty Alliance wird die Verwaltung unterschiedlicher Rollen bei einem **ASP** nicht zwingend vorgeschrieben, theoretisch kann dies aber umgesetzt werden. Entweder bietet ein einzelner **ASP** dem Benutzer eine Rollenverwaltung an oder aber der Benutzer realisiert diese über die Auswahl

des zu benutzenden **ASP** beim **SP**. Dabei wird davon ausgegangen, dass der Benutzer bei einem **ASP** die Daten für eine bestimmte Rolle hinterlegt und bei einem anderen **ASP** die Daten für eine andere Rolle. Besucht dieser Benutzer nun einen **SP**, so kann er anhand der Vorgabe, welchen **ASP** dieser **SP** kontaktieren soll, bei dem **SP** in unterschiedlichen Rollen auftreten. Voraussetzung ist, dass beide **ASP** vom **SP** benutzt werden können und dass der Benutzer bei dem **SP** vor jeder Anmeldung den zu kontaktierenden **ASP** angeben kann. Wenn der Benutzer während seiner Kommunikation mit dem **SP** zeitgleich auch bei beiden **ASP**s angemeldet ist, besteht allerdings das Risiko, dass der **SP** auch beide **ASP**s abfragen kann, um an zusätzliche Profilinformationen zu gelangen.

Kapitel 5

Zusammenfassung und Ausblick

In dieser Diplomarbeit wurden drei prominente Vertreter von Single Sign-On Lösungen vorgestellt, Kerberos, Microsoft Passport und die Liberty Alliance Spezifikation. Während Kerberos hauptsächlich für den internen Einsatz in geschlossenen Netzwerken konzipiert ist, wollen die anderen beiden Lösungen ein Verfahren für offene Netzwerke, wie das Internet, präsentieren. Dabei legen Microsoft Passport und die Liberty Alliance Spezifikation ihren Schwerpunkt auf Webservices.

Anschließend wurde eine Risikoanalyse bei Microsoft Passport und Liberty Alliance durchgeführt. Dabei wurden die Systeme vor allem auf Schwachstellen hin untersucht, die zu einem Verlust der Sicherheitsziele Verfügbarkeit, Vertraulichkeit, Integrität oder Authentisierung führen können.

5.1 Zusammenfassung

Die beiden untersuchten Systeme verfolgen das Ziel, eine Single Sign-on Lösung mit integriertem Identitymanagment anzubieten. Der Ansatz der beiden Lösungen ist jedoch sehr unterschiedlich. Während Microsoft Passport ein Produkt der Firma Microsoft darstellt, werden bei Liberty Alliance Spezifikationen veröffentlicht, die es unterschiedlichen Herstellern erlauben, Liberty Alliance konforme Produkte zu entwickeln. Dieser differierende Ansatz wurde zu Beginn dieser Arbeit bei der Informationsbeschaffung deutlich.

Da Microsoft Passport ein fertiges Produkt ist, das nur noch in eine Microsoft Infrastruktur eingebunden werden muss, finden sich sehr wenige Informationen

zu den in einer Risikoanalyse relevanten Bereichen innerhalb der Dokumentation. Bestimmte Funktionalitäten und Systemparameter werden in keinem öffentlich zugänglichen Dokument von Microsoft aufgeführt. Zudem sind die erhältlichen Informationen über viele Dokumente und unterschiedliche Orte verstreut. Ausgehend von Implementationsfehlern in Kryptoverfahren bei Microsoft Produkten[1] wollte ich das Verschlüsselungsverfahren bei Microsoft Passport genauer analysieren. Dies scheiterte jedoch am Informationsmangel in der Dokumentation von Microsoft bezüglich der eingesetzten Kryptoverfahren.

Da das Liberty Alliance Modell kein Produkt darstellt, sondern nur aus Spezifikationen und anderen Dokumenten besteht, bestand hier ein ganz anderes Problem. Liberty Alliance verfolgt das Ziel, eine Sammlung von unterschiedlichen Spezifikationen zu veröffentlichen, die eine Single Sign-On Lösung in sehr unterschiedlichen Einsatzgebieten und mit unterschiedlichen Techniken realisieren soll. Dies führt dazu, dass sich die Spezifikation über eine große Anzahl von Dokumenten verteilt und es nicht leicht zu überblicken ist, welche Anforderungen zu welchem Liberty Alliance Profil gehört. Erschwerend kommt hinzu, dass Liberty Alliance sich in ihrer Spezifikation verschiedener anderer Spezifikationen bedient, diese aber teilweise leicht abändert. Dies erschwerte die Schaffung eines Überblicks erheblich, da es nicht immer offensichtlich war, welcher Teil übernommen werden sollte und wo Änderungen von Liberty Alliance stattfanden.

Bei beiden untersuchten Systemen exisitieren, wie im Hauptteil dieser Arbeit beschrieben, eine Reihe von Schwachstellen. Teilweise können diese Schwachstellen zu großen Schäden führen. Deswegen sollten bei beiden Systemen Nachbesserungen erfolgen. Dies sollte bei Liberty Alliance durch Zusätze und neuere Spezifikationsversionen erfolgen. Obwohl Microsoft angeblich angekündigt hat, ihre Single Sign-On Infrastrukur nicht mehr zu vermarkten[2], sollte dieses System den-

[1] Zu nennen wäre einerseits das „Point to Point Tunneling Protokoll“ (MS-PPTP) (siehe `http://www.securityfocus.com/archive/1/12489` und [SM99]) als auch die Dokumentverschlüsselung in der Microsoft Office Suite (siehe [Wu05]).

[2] Nachdem IBM der Liberty Alliance beigetreten ist und zwei große Internetdienstleister (Ebay und Amazon) seit Begin 2005 das Microsoft Passport Verfahren nicht mehr anbieten, mehren sich die Hinweise, dass Microsoft ihr Passport System nur noch intern bei ihren eigenen Dienstleistungen anbieten werden (siehe auch: `http://www.theregister.co.uk/2004/12/30/ms_ends_pass/` und `http://www.dradio.de/dlf/sendungen/computer/317497/`).

noch nicht bei Sicherheitsbetrachtungen vernachlässigt werden. Zum einen soll es noch weiterhin innerhalb von Microsoftdienstleistungen Verwendung finden, außerdem kann das Passport-Verfahren jederzeit innerhalb eines neuen Produkts von Microsoft wieder auftauchen.

Grundsätzlich ist der Ansatz des Liberty Alliance Modells zu bevorzugen. Dies gilt zunächst aus Anwendersicht, da ein wesentlich höherer Datenschutz geboten werden kann. Aber auch die anderen Teilnehmer können mit Vorteilen rechnen. Eine offene Spezifikation erlaubt es, dass sehr unterschiedliche Herrsteller Produkte für dieses System entwickeln können. Somit kann sich eine größere Produktpalette entfalten und die Integration in bestehende Systeme kann individueller erfolgen. Durch den Umstand einer offenen Spezifikation besteht zudem die Möglichkeit, auch eigene Implementationen oder bestehende Open Source Versionen zu adaptieren und den internen Gegebenheiten anzupassen. Gerade in einem so sensiblen Bereich wie der Authentisierung und dem Identitymanagment erhöht der Einblick in den Sourcecode das Vertrauen und die Sicherheit enorm. Außerdem besteht bei Liberty Alliance eine größere Zukunftssicherheit für den Einsatz, da man nicht dem Wohlwollen einer einzigen Firma mit ihrer Vertriebspolitik ausgesetzt ist.

Liberty Alliance ist noch nicht fehlerfrei und es gibt sicherlich Verbesserungsbedarf. Dennoch kann die Vorgehensweise dieser Organisation als ein Schritt in die richtige Richtung angesehen werden. Ohne Zweifel besteht ein Bedarf an Single Sign-On Lösungen, und eine präzise, offene und herstellerunabhängige Spezifikation bildet eine Grundlage für viele unterschiedliche Produkte. Die Zukunft wird zeigen, ob Hersteller und Anwender die Spezifikation annehmen werden.

Auf jeden Fall sollte ein unabhängiges Gremium über die Güte der Produkte Auskunft geben und diese bewerten. Eine unabhängige Zertifizierungstelle sollte dabei nicht nur die Interoperabilität untersuchen, sondern auch Implementierungsmethoden und Systemumgebungen bei ihren Betrachtungen einschließen. Bei Microsoft Passport muss man der Firma Microsoft vertrauen, dass das Produkt keine Mängel aufweist. Liberty Alliance bietet einen Interoperabilitätstest mit zugehörigem Zertifikat an. Die Testanforderungen sind unter [KSST04] beschrieben. Leider betrachtet die Anforderung bisher nur das Zusammenspiel von unterschiedlichen Produkten. Diese Zertifizierungsanforderungen sollten auf je-

den Fall noch erweitert werden.

5.2 Ausblick

Single Sign-On ist kein neues Untersuchungsfeld innerhalb der Informatik. Durch die Kommerzialisierung des Internets haben sich aber neue Umgebungen entwickelt, die andere Anforderungen an ein Single Sign-On System stellen als bei herkömmlichen Lösungen. Aufbauend auf diese Arbeit bieten sich weitere Untersuchungsschwerpunkte an.

Es wäre u. a. eine Untersuchung des allgemeinen Themenkomplexes der Vertrauensbeziehung möglich. Dabei könnte die Frage untersucht werden, wie Vertrauensbeziehungen formal ausgedrückt werden können und ob es Verfahren gibt, die Vertrauensbeziehungen verifizierbar und beweisbar machen können. Gerade der Einsatz von Single Sign-On Lösungen in Authentication Infrastructures mit verteilten Verwaltungsstrukturen benötigt Hilfsmittel, um Sicherheitsbeziehungen zu validieren und überschaubar zu machen. Eventuell lassen sich Verfahren aus diesem Bereich mit Produkten visualisieren und böten somit einem Benutzer Werkzeuge an, sein Identitymanagment verwaltbarer und übersichtlicher zu gestalten.

Eine anderer Aspekt wäre die Untersuchung von Integrationsmöglichkeiten von sogenannter „Privacy Enhanced Technology“ (PET) in Single Sign-On Systeme. Welche dieser unterschiedlichen Technologien, wie Crowds, Anonymisierer, Mix-Netze oder Onion-Routing-Verfahren lassen sich einbinden und nutzen? Eventuell müssen auch neue datenschutzgewährleistende Verfahren, speziell für den Einsatz bei Single Sign-On, gefunden werden.

Ein weiterer Themenkomplex wäre das Entwerfen, Validieren oder Bewerten von Testumgebungen für diesen Bereich. Dies kann sowohl ein Zertifizierungskonzept beinhalten, als auch eine davon unabhängige Sicherheitsuntersuchung. Der Validierungsaspekt darf gerade bei so komplexen Systemen nicht vernachlässigt werden. Schon jetzt besteht die Gefahr von Implementationsfehlern bei der Umsetzung der Liberty Alliance Spezifikation.

Single Sign-On Verfahren für verteilte Systeme werden auch zukünftig benötigt. Neben abgeschlossenen Lösungen für spezielle Bereiche wird auch innerhalb

der sogenannten „zero-footprint“ Techniken ein Bedarf existieren. Eventuell sollten aber die Anforderungen an die Ausstattung des Anwendersystems bei diesen „zero-footprint“ Lösungen erhöht werden, um somit ein sichereres System anzubieten. Der Einsatz von Java oder Javascript beim Anwender würde das Generieren von zusätzlichen Sicherheitsmerkmalen erlauben. Schon heute bieten fast alle aktuellen Webbrowser und Mobiltelefone die dazu notwendigen Technologien an. Es sei aber auch erwähnt, dass Verfahren, die Systeme sicherer und verbraucherfreundlicher machen, häufig diese auch verkomplizieren und dadurch fehleranfälliger machen. Deswegen bedarf es in diesem Bereich großer Transparenz und Offenheit bei Konzeption, Entwicklung, Implementation, Einsatz und Betrieb dieser Systeme.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[ABCS03] Aarts, Robert, John Beatty, Conor Cahill und Xavier Serret: *Liberty ID-FF Protocols and Schema Specification*. In: Cantor, Scott und John Kemp (Herausgeber): *Liberty Alliance Project Version 1.2*. Liberty Alliance Project, 2003.

[AKW03] Aarts, Robert, Slava Kavsan und Tom Wason: *Liberty ID-FF Bindings and Profiles Specification*. In: Cantor, Scott und John Kemp (Herausgeber): *Liberty Alliance Project Version 1.2*. Liberty Alliance Project, 2003.

[Ant04] Anti-Phishing Working Group: *Phishing Activity Trends Report December 2004*. Technischer Bericht, Anti-Phishing Working Group, Dezember 2004.

[Art02] Artikel 29 Datenschutzgruppe der Europäischen Union: *Arbeitspapier zu Online-Authentifizierungsdiensten*, Januar 2002.

[BBF+02] Bartel, M., J. Boyer, B. Fox, B. LaMacchia und E. Simon: *XML-Signature Syntax and Processing*. W3C Recommendation, World Wide Web Consortium, Februar 2002.

[BLFF96] Berners-Lee, T., R. Fielding und H. Frystyk: *Hypertext Transfer Protocol – HTTP/1.0*. Request for Comments 1945, Internet Society - Network Working Group, Mai 1996.

[BLuLM98] Berners-Lee, T. und R. Fielding und L. Masinter: *Uniform Resource Identifiers (URI): Generic Syntax*. Request for Comments 2396, Internet Society - Network Working Group, 1998.

[Bra97] Bradner, S.: *Key words for use in RFCs to Indicate Requirement Levels*. Request for Comments 2119, Internet Society - Network Working Group, 1997.

[Bry88] Bryant, Bill: *Designing an Authentication System: a Dialogue in Four Scenes*, Februar 1988.

[Bun03] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutzhandbuch*. Bundesamt für Sicherheit in der Informationstechnik, 2003.

[CHKT03] Cantor, Scott, Jeff Hodges, John Kemp und Peter Thompson: *Liberty ID-FF Architecture Overview*. In: Wason, Thomas (Herausgeber): *Liberty Alliance Project Version 1.2*. Liberty Alliance Project, 2003.

[CKPS00] Courtois, Nicolas, Alexander Klimov, Jacques Patarin und Adi Shamir: *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*. In: *Eurocrypt*, Seiten 392–407, 2000.

[Cle02] Clerq, Jan De: *Single Sign-on Architectures*. In: Davida, George (Herausgeber): *Infrastructure Security*, Band 2437 der Reihe *InfraSec 2002*, Seiten 1–18. Springer LNCS, Oktober 2002.

[CP02] Courtois, Nicolas und Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*. Cryptology ePrint Archive, Report 2002/044, 2002.

[DA99] Dierks, T. und C. Allen: *The TLS Protocol*. Request for Comments 2246, Internet Society - Network Working Group, 1999.

[Dob96a] Dobbertin, Hans: *Cryptanalysis of MD4*. In: *Fast Software Encryption*, Seiten 53–69, 1996.

[Dob96b] Dobbertin, Hans: *The Status of MD5 After a Recent Attack*. RSA Laboratories - CryptoBytes, 2(2), 1996.

[FGM+99] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*. Request for Comments 2616, Internet Society - Network Working Group, Juni 1999.

[FKK96] Freier, Alan O., Philip Karlton und Paul C. Kocher: *The SSL Protocol*. Draft, Netscape, 1996. URL: `http://wp.netscape.com/eng/ssl3/ssl-toc.html`.

[Fro03] Froehling, Willem Joaquin: *Konzept und exemplarische Implementation eines gesicherten Kanals zur Übertragung biometrischer Daten*. Studienarbeit, Universität Hamburg, Januar 2003.

[FSW01] Ferguson, Niels, Richard Schroeppel und Doug Whiting: *A Simple Algebraic Representation of Rijndael*. In: *Selected Areas in Cryptography*, Seiten 103–111, 2001.

[GHM+03a] Gudgin, M., M Hadley, N. Mendelsohn, J. Moreau und H. F. Nielsen: *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation, World Wide Web Consortium, Juni 2003.

[GHM+03b] Gudgin, M., M Hadley, N. Mendelsohn, J. Moreau und H. F. Nielsen: *SOAP Version 1.2 Part 2: Adjuncts*. W3C Recommendation, World Wide Web Consortium, Juni 2003.

[HL98] Howard, John D. und Thomas A. Longstaff: *A Common Language for Computer Security Incidents*. Technischer Bericht, Sandia National Laboratories, Oktober 1998.

[HPFS02] Housley, R., W. Polk, W. Ford und D. Solo: *Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile*. Request for Comments 3280, Internet Society - Network Working Group, April 2002.

[IDS02] Imamura, T., B. Dillaway und E. Simon: *XML Encryption Syntax and Processing*. W3C Recommendation, World Wide Web Consortium, Dezember 2002.

[JtM01] Jendricke, Uwe und Daniela Gerd tom Markotten: *Identitätsmanagement: Einheiten und Systemarchitektur*, Seiten 77–85. Vieweg, Wiesbaden, September 2001.

[Kas98] Kassovic, Marian: *Vortragsfolien - Risikoanalyse des Homebanking Standards HBCI*. In: *VTC-Seminar des Arbeitsbereiches AGN des Fachbereiches Informatik an der Universität Hamburg*. 1998.

[Kel01] Kelter, H.: *Das Ende der Anonymität? Datenspuren in modernen Netzen*. Bundesamt für Sicherheit in der Informationstechnik, 2001.

[KN93] Kohl, J. und C. Neuman: *The Kerberos Network Authentication Service(V5)*. RFC, Internet Engineering Task Force (IETF), September 1993.

[KR00] Kormann, David P. und Aviel D. Rubin: *Risks of the Passport Single Signon Protocol*. In: *IEEE Computer Networks*. IEEE, Juli 2000.

[KSST04] Kemp, J., J. Sergent, R. Sullivan und E. Tiffany: *Liberty ID-FF 1.2 Interoperability Conformance Testing Procedures*. In: *Liberty Alliance Project Version 1.1*. Liberty Alliance Project, April 2004.

[Lek93] Lektorat des B.I.-Wissenschaftsverlags (Herausgeber): *Informatik Duden, 2., vollständig überarbeitete und erweiterte Auflage*. Dudenverlag, 1993.

[Lib03a] Liberty Alliance Project: *Liberty Authentication Context 1.2*. Internet, August 2003.

[Lib03b] Liberty Alliance Project: *Liberty ID-FF 1.2 Errata*. In: *Liberty Alliance Project Version 1.1*. Liberty Alliance Project, 2003.

[Lib03c] Liberty Alliance Project: *Liberty ID-FF 1.2 Static Conformance Requirements*. In: *Liberty Alliance Project Version 1.1*. Liberty Alliance Project, 2003.

[Lib03d] Liberty Alliance Project: *Liberty Technical Glossary - Version 1.3*. In: *Liberty Alliance Project*. Liberty Alliance Project, 2003.

[Lib04a] Liberty Alliance Project: *Current Members*. Website, September 2004.

[Lib04b] Liberty Alliance Project: *Liberty Metadata Description and Discovery Specification - Version 1.0*. In: Davis, Peter (Herausgeber): *Liberty Alliance Project*. Liberty Alliance Project, 2004.

[May04] Mayo, Drew: *Eine Einführung in die Sitepersonalisierung durch Microsoft Passport*. MSDN, 2004.

[Mes04] MessageLabs: *MessageLabs Intelligence Annual Email Security Report 2004*. Technischer Bericht, Dezember 2004.

[Mic04a] Microsoft: *Building trust in Internet privacy: The New .Net Passport*. Internet, 2004.

[Mic04b] Microsoft: *Collecting User Data*. .NET Passport SDK, August 2004.

[Mic04c] Microsoft: *Component Configuration Document*. MSDN - .NET Passport SDK, August 2004.

[Mic04d] Microsoft: *Core Profile Table*. MSDN - .NET Passport SDK, August 2004.

[Mic04e] Microsoft: *Hard Sign-In: Step By Step*. .NET Passport SDK, August 2004.

[Mic04f] Microsoft: *Implementing Sign-Out and Deleting Cookies*. .NET-Passport SDK, August 2004.

[Mic04g] MICROSOFT: *Microsoft .NET Passport - Review Guide*. review guide, Microsoft, Januar 2004.

[Mic04h] MICROSOFT: *Microsoft .NET Passport-Datenschutzbedingungen*. http://www.passport.com, April 2004.

[Mic04i] MICROSOFT: *.NET Passport Cookies*. MSDN - .NET Passport SDK, August 2004.

[Mic04j] MICROSOFT: *.NET Passport Glossary*. .NET-Passport SDK, August 2004.

[Mic04k] MICROSOFT: *.NET Passport System Requirements*. MSDN - .Net Passport SDK, August 2004.

[Mic04l] MICROSOFT: *Passport Manager Administration Utility*. MSDN - .NET Passport SDK, August 2004.

[Mic04m] MICROSOFT: *PassportIdentity.IsAuthenticated*. MSDN - .NET Passport SDK, August 2004.

[Mic04n] MICROSOFT: *Registering Your .NET Passport Site*. MSDN - .NET Passport SDK, August 2004.

[Mic04o] MICROSOFT: *Saving State*. MSDN - .NET Passport SDK, August 2004.

[Moc87a] MOCKAPETRIS, P.: *Domain Names - Concepts and Facilities*. Request for Comments 1034, Internet Society - Network Working Group, November 1987.

[Moc87b] MOCKAPETRIS, P.: *Domain Names - Implementation and Specification*. Request for Comments 1035, Internet Society - Network Working Group, November 1987. URL: `ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt`.

[MVO96] MENEZES, ALFRED J., SCOTT A. VANSTONE und PAUL C. VAN OORSCHOT: *Handbook of Applied Cryptography*. CRC Press, Inc., 1996.

[NS78] Needham, Roger M. und Michael D. Schroeder: *Using Encryption for Authentication in Large Networks of Computers*. Technischer Bericht, Xerox Palo Alto Research Center, Dezember 1978.

[OAS03a] OASIS Security Services Technical Committee: *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*. Technischer Bericht, OASIS, September 2003.

[OAS03b] OASIS Security Services Technical Committee: *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1*. Technischer Bericht, OASIS, September 2003.

[PM03] Pashalidis, Andreas und Chris J. Mitchell: *A Taxonomy of Single Sign-on Systems*. In: Safavi-Naini, Rei und Jennifer Seberry (Herausgeber): *Information Security and Privacy*, Nummer 2727 in *ACISP*, Seiten 249–264, Juli 2003.

[PW03] Pfitzmann, Birgit und Michael Waidner: *Analysis of Liberty Single-Signon with Enabled Clients*. IEEE Internet Computing 7(6) Nov/Dec 2003, Seiten 38–44, 2003.

[Res00] Rescorla, E.: *HTTP Over TLS*. Request for Comments 2818, Internet Society - Network Working Group, Mai 2000.

[Sch96] Schneier, Bruce: *Angewandte Kryptographie - Protokolle, Algorithmen und Sourcecode in C*. Addison-Wesley, Mai 1996.

[Shi00] Shirey, R.: *Internet Security Glossary*. Request for Comments 2828, Internet Society - Network Working Group, Mai 2000.

[SM99] Schneier, Bruce und Mudge: *Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)*. Lecture Notes In Computer Science Proceedings of the International Exhibition and Congress on Secure Networking - CQRE (Secure) '99 Springer-Verlag, 1740:192 – 203, 1999.

[SNS88] Steiner, J. G., B. C. Neuman und J. I. Schiller: *Kerberos: An Authentication Service for Open Network Systems*. Technischer Bericht, Project Athena - Massachusetts Institute of Technology, Januar 1988.

[Sta99a] Stallings, William: *Cryptography and Networking Security: Principles and Practice*. Prentice Hall, zweite Auflage, 1999.

[Sta99b] Standard, ISO: *Safety aspects – Guidelines for their inclusion in standards*. ISO Standard, Dezember 1999.

[Ste02] Stelzer, Dirk: *Risikoanalysen als Hilfsmittel zur Entwicklung von Sicherheitskonzepten in der Informationsverarbeitung*, Seiten 37–54. 2002.

[The97] The Open Group: *X/Open Single Sign-On Service (XSSO) Pluggable Authentication*. Specification, The Open Group, Juni 1997.

[uLM00] L. Montulli, D. Kristol und: *HTTP State Management Mechanism*. Request for Comments 2965, Internet Society - Network Working Group, Oktober 2000.

[WFLY04] Wang, Xiaoyun, Dengguo Feng, Xuejia Lai und Hongbo Yu: *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*. Technischer Bericht, Shandong University of China, August 2004.

[Wu05] Wu, Hongjun: *The Misuse of RC4 in Microsoft Word and Excel*. Technischer Bericht, Institute for Infocomm Research, Singapore, Januar 2005.

[WYY05] Wang, Xiaoyun, Yiqun Lisa Yin und Hongbo Yu: *Collision Search Attacks on SHA1*. Technischer Bericht, Februar 2005.

5.3 Anhang

Es folgt eine ausgewählte Liste von Sicherheitsproblemen bei populären Webbrowsern, die im Jahre 2004 bekannt worden. Diese Liste besteht nur aus einem kleinen Ausschnitt von Veröffentlichungen die Webbrowserschwachstellen betreffen. Weitere Schwachstellen lassen sich bei Mailinglisten nachforschen, wie BUGTRAQ `http://www.securityfocus.com/archive/1`, FULL-DISCLOSURE `http://lists.grok.org.uk/pipermail/full-disclosure/` oder VULNWATCH `http://archives.neohapsis.com/archives/vulnwatch/`.

- URL-Spoofing beim Internet Explorer 6 und Outlook Express 6 trotz installiertem Service Pack 2 möglich. (`http://www.securityfocus.com/archive/1/359139`)
- Nach der Installation des Sammelpatchs MS04-011 von Microsoft benutzt der Internet Explorer für eine SSL-Verbindung nur noch eine Verschlüsselungsstärke von 0 Bit anstelle von 128 Bit. (`http://lists.grok.org.uk/pipermail/full-disclosure/2004-April/020246.html`)
- Durch einen Fehler in einem ActiveX Control beim Internet Explorer 6 mit Service Pack 2 lässt sich beliebiger HTML und Java Script Code in eine dargestellte Webseite einschleusen, u.a. können somit Cookies beim Benutzer ausgelesen werden. (http://freehost07.websamba.com/greyhats/abusiveparent-discussion.htm)
- Anhand eines Fehlers bei Mozilla Firefox ist es möglich, beliebige Webseiten im Kontext eines fremden Zertifikats anzuzeigen. (`http://www.cipher.org.uk/index.php?p=advisories/Certificate_Spoofing_Mozilla_FireFox_25-07-2004.advisory`)
- Die Webbrowser Internet Explorer 6, Mozilla Firefox, Opera und Konqueror senden unter bestimmten Vorraussetzungen Cookies an unautorisierte Server. (`http://www.westpoint.ltd.uk/advisories/wp-04-0001.txt`)

Erklärungen:

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Quellen oder Hilfsmittel nicht bedient habe.

Hamburg, den 29. März 2005 Willem Froehling

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereichs einverstanden

Hamburg, den 29. März 2005 Willem Froehling