Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften

Studienarbeit

Single Sign-On-Technologien für das World Wide Web

Marc Schönberg

5schoenb@informatik.uni-hamburg.de
Studiengang Informatik
Matr.-Nr. 4829826
Fachsemester 23

Betreuer: Prof. Dr. W. Lamersdorf

Inhaltsverzeichnis

1 Einleitung

Geschäfte werden heute zunehmend über das Internet getätigt. Um Waren oder Dienstleistungen online zu kaufen oder zu nutzen, ist beim Anbieter zuvor fast immer eine Registrierung notwendig, an deren Ende der Kunde über einen Benutzernamen und ein Passwort verfügt, mit denen er sich fortan beim jeweiligen Anbieter authentifizieren kann. Mit jeder weiteren Registrierung steigt die Anzahl der Benutzernamen und Passworte, denn es kann nicht davon ausgegangen werden, dass ein bestimmter, favorisierter Benutzername bei einem bestimmten Dienst gewählt werden kann, wenn sich ein Benutzer dort neu registriert. Bei einigen Anbietern kann der Benutzer den Benutzernamen gar nicht erst wählen, sondern bekommt einen zugewiesen, wie es häufig bei Web-Hostern vorkommt, oder am Arbeitsplatz, wo die Vergabe von Benutzernamen einem durch die Administratoren gepflegten Schema folgt. Durch alle diese Umstände wird es unwahrscheinlich, dass ein Benutzer überall, wo er Zugangsdaten erhält, den gleichen Benutzernamen bekommt. Dazu kommt die Wahl der Passwörter. Schon aus Sicherheitsgründen ist davon abzuraten, bei allen Zugängen, die ein Benutzer hat, dasselbe Passwort zu verwenden. Aber auch die Passwortrichtlinien der Anbieter können verhindern, dass ein Benutzer immer wieder dasselbe Passwort verwenden kann. Solche Richtlinien können z.B. Vorgaben über die Zusammensetzung des Passwortes aus Groß- und Kleinbuchstaben, Ziffern und Sonderzeichen enthalten oder über die erforderliche Länge. Wird der Benutzer so zur Wahl immer neuer Passwörter gezwungen, muss er sich zu den Benutzernamen auch die zugehörigen Passwörter merken. Schreibt sich der Benutzer die Daten zur Hilfe auf, wäre das wiederum unsicher. Dazu kommt meistens noch die Notwendigkeit, sich mehrstellige Geheimzahlen für Scheck- und Kreditkarten, das Mobiltelefon, Code-geschützte Türen bei der Arbeit, Codewörter für die telefonische Abwicklung von Geldgeschäften, Codewörter für die Authentifizierung beim häuslichen Wachdienst und dergleichen mehr merken zu müssen, die ebenfalls aus Sicherheitsgründen nirgendwo aufgeschrieben sein sollten.
Bestrebungen nach Single Sign-On verfolgen daher das Ziel, es Benutzern zu ermöglichen, sich nur einmal pro Sitzung mit einem einzigen Satz Zugangsdaten authentifizieren zu müssen, so dass sie sich nur einen Benutzernamen und das dazugehörige Passwort merken müssen.

1.1 Ziel der Arbeit

Die Realisierung von Single Sign-On kann grundsätzlich auf zwei Arten erfolgen. Einmal durch eine zentrale Authentifizierungsstelle, die sämtliche Benutzerkennungen in ihrem Einflussbereich verwaltet, oder durch die Verknüpfung mehrerer dezentral vorliegender Benutzerkennungen und dezentral organisierte Authentifizierungsstellen. Ziel dieser Arbeit ist, zwei Single Sign-On-Technologien vorzustellen und miteinander zu vergleichen. *Microsoft .NET Passport*, das mittlerweile unter der Bezeichnung *Live ID* läuft, ist ein zentral organisiertes System. Das *Liberty Alli-*

ance Project stellt einen Gegenentwurf dar und setzt auf die Verknüpfung mehrerer dezentral vorliegender Benutzerkennungen.

1.2 Abgrenzung des Themas

In dieser Arbeit soll die grundsätzliche Funktionsweise von Single Sign-On (SSO) dargestellt werden und zwei unterschiedliche Ansätze zur Herangehensweisen anhand zweier konkreter Beispiele deutlich gemacht werden. Naben den Entwicklern der beiden Beispiele gibt es weitere Entwickler und Entwicklergruppen, die eigene Lösungen für SSO erarbeiten. Da diese Arbeit keine Aufstellung aller SSO-Lösungen sein will, beschränkt sie sich auf die genannten Beispiele. Im Falle von .NET Passport gibt es bereits einen Nachfolger namens Windows Live ID, der zukünftig noch eine neue Technologie unterstützen soll. Sie wird in hier kurz vorgestellt.
Soweit Basistechnologien beschrieben werden, die entweder von .NET Passport oder vom LAP für Single Sign-On verwendet werden, beschränkt sich das auf den jeweils für das Verständnis von SSO relevanten Teil. Für eine Vertiefung sei auf die einschlägige Literatur verwiesen.

1.3 Organisation der Arbeit

Zum besseren Verständnis der Vorgänge beim Single Sign-On werden in Kapitel 2 zuerst die Basistechnologien beschrieben, die den beiden zu vergleichenden Single Sign-On-Technologien zu Grunde liegen. Dabei stehen die Bereiche der Basistechnologien im Vordergrund, die für die Umsetzung des Single Sign-On entscheidend sind. Die Reihenfolge ist nach Möglichkeit so gewählt, dass die Basistechnologien aufeinander aufbauend angeordnet sind. In den Kapiteln 3 und 4 folgt dann die Beschreibung von .NET Passport und des Liberty Alliance Projects und wie Single Sign-On mit der jeweiligen Technologie funktioniert. Kapitel 5 stellt beide Technologien noch einmal direkt gegenüber. Der Versuch einer Risikoanalyse soll darlegen, welche Probleme durch die Single Sign-On-Technologien gelöst werden und welche bestehen bleiben oder sich neu ergeben.

1.4 Begriffe und Schreibweisen

In der Landschaft der Single Sign-On-Technologien gibt es unterschiedliche Bezeichnungen und Schreibweisen: Microsoft verwendet den Begriff *Single Sign-in*, das Liberty Alliance Project verwendet *Single Sign-On*, Shibboleth benutzt die Schreibweise *Single-SignOn* und OpenID die Schreibweise *single-sign-on.* In dieser Arbeit wird die Schreibweise *Single Sign-On* benutzt. In Kapitel 3 erscheint vereinzelt auch die Schreibweise *Single Sign-in.*

Die Worte SOLLTE, MUSS, DARF NICHT usw. erscheinen in GROSSBUCHSTABEN, wenn sie in der gem. [RFC2119] definierten Bedeutung verwendet werden. Sie kommen hauptsächlich in Kapitel 2 bei der Beschreibung der Basistechnologien vor.

2 Technische Grundlagen

Jeder Versuch eine neue Technologie im *World Wide Web* (WWW) einzuführen erfordert, sich mit den bereits vorhandenen Technologien auseinander zu setzen, und es liegt nahe, sich die etablierten Technologien zunutze zu machen und auf ihnen aufzubauen. In diesem Kapitel werden die verwendeten Basistechnologien vorgestellt. Dabei soll der Schwerpunkt auf den für Single Sign-On relevanten Bestandteilen liegen, sodass die einzelnen Beschreibungen punktuell und nicht vollständig ausfallen. Für eine vollständige Darstellung sei stattdessen auf die einschlägige Literatur verwiesen.

Bei der Liberty Alliance setzt man bewusst auf bestehende Industriestandards, die bereits von der *Organization for the Advancement of Structured Information Standards (OASIS)*, dem *World Wide Web Consortium (W3C)* und der *Internet Engineering Task Force (IETF)* entwickelt wurden, anstatt selbst noch einmal ähnliche Technologien zu entwickeln [LAP03].

Microsoft setzt bei Live ID und CardSpace ebenfalls auf Technologien aus offenen Standards, die die Firma größtenteils selbst oder aber zumindest mit starker Beteiligung entwickelt hat und dann den Standardisierungsgremien übergeben hat. Anfangs wurden bei .NET Passport jedoch proprietäre Produkte eingesetzt.

Die *Security Assertion Markup Language* (SAML) enthält inzwischen in der aktuellen Version 2.0 einen großen Teil der Arbeit aus dem *Identity Federation Framework* (ID-FF) des Liberty Alliance Project. Nachdem das LAP sein ID-FF auf der Basis von SAML 1.x aufgebaut hatte, hat es die eigenen Erweiterungen in die laufende Entwicklungsarbeit zu SAML 2.0 einfließen lassen, um möglichst nur einen einzigen Standard für föderatives Single Sign-On entstehen zu lassen. Deshalb will das LAP künftig auf SAML 2.0 aufbauen, anstatt das eigene Rahmenwerk weiter zu entwickeln. [Wis+05]

Für die Gliederung dieser Arbeit ergibt sich aus dieser wechselseitigen Beeinflussung ein Reihenfolgenproblem: Die korrekte Beschreibung der Grundlagentechnologie SAML und ihrer Möglichkeiten nimmt die Funktionsweisen des LAP vorweg. Andererseits wäre es aber auch nicht richtig, diese Funktionsweisen ausschließlich der Arbeit des LAP zuzuordnen, indem SAML im LAP-Teil beschrieben wird. Dem Prinzip „Vom Allgemeinen zum Speziellen“ folgend, ist die Beschreibung von SAML in diesem Kapitel angesiedelt, während das LAP eine darauf aufbauende Architektur darstellt, die in Kapitel 4 beschrieben wird.

2.1 URI – URL – URN

Die Begriffe *Uniform Resource Identifier* (URI), *Uniform Resource Locator* (URL) und *Uniform Resource Name* (URN) tauchen im Kontext des Internet immer wieder auf. Teilweise scheinen insbesondere die Begriffe URI und URL synonym verwendet zu werden. Sie erscheinen auch sehr ähnlich. Da sie im Zusammenhang mit den im Folgenden beschriebenen Basistechnologien vorkommen, sollen sie hier kurz erläutert werden.

Ein ***URI*** ist allgemein ein Identifikationsmerkmal einer im Internet liegenden Ressource. Ein URI ist entweder ein URL oder ein URN. Beide Varianten identifizieren jeweils eine bestimmte Ressource. URI ist der Oberbegriff.

Ein ***URL*** ist eine spezielle Form von URI und bezeichnet *die Ressource, die unter der Adresse ABCDE zu finden ist*. Es handelt sich also um eine Ortsangabe (engl.: location). Der URL gibt zusätzlich das Protokoll an, mit dem die Ressource angesprochen werden soll.

> *„URIs which refer to objects accessed with existing protocols are known as 'Uniform Resource Locators' (URLs).“* [RFC1630]Seite 3

Also setzt sich ein URL folgendermaßen zusammen:

> *„URL = Protokoll / Maschine / Pfad zu der Ressource auf der Maschine“*
> [ACKM04] Seite 95

Beim URL kann es sein, dass sich der Inhalt der Ressource, die an der beschriebenen Stelle zu finden ist, im Laufe der Zeit verändert, eine völlig neue Ressource an dieselbe Stelle rückt oder gar keine Ressource mehr zu finden ist.

Ein ***URN*** bezeichnet *die Ressource, die in einem bestimmten Namensraum unter dem Namen XYZ bekannt ist.* Bezogen auf einen Namensraum ist der Name einer Ressource eindeutig. Ein Beispiel für URN abseits des Internet sind die *Internationalen Standard Buch-Nummern* (ISBN). Es wird ein Namensraum aus zehnstelligen Ziffernfolgen von 0-9 definiert, wobei die letzte Ziffer zusätzlich ein ‚X' sein kann.
[ACKM04] [RFC1630]

2.2 Web Redirection

Um Daten zwischen den beteiligten Providern einer Transaktion auszutauschen, wird Web Redirection benutzt. Dabei wird die Information durch das Client-Programm des Benutzers hindurch geleitet. Es entsteht also ein Kommunikationskanal zwischen den Providern. Das Benutzer-Programm (meistens ein WWW-Browser) kann auf verschiedene Arten dazu veranlasst werden, automatisch eine neue Seite anzufordern. In der Adresse der neuen Seite (URL oder URI?) können Daten für den anzusprechenden Server als Parameter untergebracht werden.
Ein Beispiel ist die Anmeldung eines Benutzers bei einem Service Provider. Der Benutzer soll über einen externen Identity Provider authentifiziert werden, doch dafür müssen die beteiligten Provider miteinander kommunizieren können und in einen gemeinsamen Kontext mit dem Clientprogramm des Benutzers (Webbrowser) gebracht werden.

Dieses wiederum zeigt aber nur Seiten an, die es zuvor ausdrücklich angefordert hat. Es bestünde also für den Identity Provider gar keine Möglichkeit, die Anmeldeseite beim Benutzer anzeigen zu lassen, wenn der Benutzer bzw. dessen Programm sie nicht ausdrücklich angefordert hätte. Ferner würden Firewalls unüberwindbare Hindernisse darstellen und das Liberty Alliance Project a priori scheitern lassen.

Web Redirection kann einmal durch HTTP-Redirect realisiert werden. Das HTTP sieht die Möglichkeit der Umleitung vor. Häufig wird sie nach dem Umzug einer bestehenden Website auf einen anderen Server oder eine andere Domäne angewendet. Benutzer, die nach dem Umzug noch versuchen die Site unter der alten Adresse zu erreichen, werden durch eine mehr oder weniger schlichte Seite über den Umzug informiert und nach einigen Sekunden Wartezeit oder aber sofort auf die neue Adresse umgeleitet. Andere Homepages starten mit einer Eingangsseite, die automatisch nach einigen Sekunden zum eigentlichen Inhalt weiterführt.

Des Weiteren kann Web Redirection auch per HTML-Formular und JavaScript im Webbrowser des Benutzers realisiert werden. Im Unterschied zu HTTP-Redirect ist Web Redirection dabei auf bestimmte Eigenschaften des Webbrowsers angewiesen, nämlich dessen Unterstützung von JavaScript. Manche Benutzer oder Administratoren deaktivieren JavaScript im Webbrowser aus Sicherheitsgründen. Für Single Sign-On bedeutet dies, dass die Verwendung von Techniken, die auf JavaScript angewiesen sind, nicht exklusiv geschehen sollte. Vielmehr sollte immer mindestens eine weitere Technik parallel dazu unterstützt werden, um auch auf solchen restriktiv konfigurierten Browsern zu funktionieren.

2.2.1 HTTP-Redirect

Im Kontext von Single Sign-On kommen bisweilen HTTP-Weiterleitungen (Statuscode 3xx) zum Einsatz. Die hierfür relevanten Punkte des HTTP werden in dieser Arbeit beschrieben. Für einen grundsätzlichen Einstieg in HTTP sei an dieser Stelle auf die Literatur verwiesen, insbesondere [RFC2616].
Das *Hypertext Transfer Protocol (HTTP)* wurde – wie es sein Name sagt – hauptsächlich zum Abruf und zur Übertragung von Hypertext, also HTML-codierten Inhalten, von WWW-Servern entwickelt. Es ist aber nicht auf die Übertragung HTML-codierter Inhalte (Dateien) beschränkt, sondern kann ganz allgemein zur Übertragung von Dateien (beliebigen Inhalts) verwendet werden. HTTP arbeitet nach dem *Client/Server-Prinzip*. Wenn ein Benutzer mithilfe seines Browsers (client) eine Seite im WWW abruft, findet zwischen Browser und WWW-Server eine Kommunikation nach dem HTTP statt. [ACKM04]

Der Client baut die Verbindung zum Server auf und sendet seine Anfrage, die aus der Anfragemethode, dem *Uniform Resource Indentifier* (URI; s.o.) der angeforderten Ressource (Datei i.w.S.)

und der verwendeten Protokollversion, gefolgt von einer MIME-artigen[1] Nachricht, die hier aber nicht weiter erläutert werden soll, besteht. [ACKM04] [RFC2616] [RFC2396]
Die Antwort des Servers setzt sich aus einer Statusmeldung und einem MIME-artigen Teil zusammen, der im Erfolgsfall das angeforderte Dokument enthält. Die Statusmeldung enthält die verwendete Protokollversion, einen dreistelligen Statuscode (Fehler oder Erfolg) und eine kurze natürlichsprachliche Meldung zum Statuscode. Nach der Übertragung beendet der Server die Verbindung wieder. [ACKM04] [RFC2616]

Durch den Statuscode informiert der Server den Client darüber, ob seine Anfrage erfolgreich bearbeitet werden konnte oder woran die Bearbeitung seiner Anfrage ggf. gescheitert ist. Im Erfolgsfall wird die Anfrage mit dem Statuscode 200 für „OK“ und der Übermittlung des angeforderten Dokuments beantwortet und abgeschlossen. HTTP 1.1 kennt fünf Statuscodeklassen, die durch die erste (linke) Ziffer des dreistelligen Statuscodes gekennzeichnet sind. Die Klassen werden wie folgt beschrieben:

1xx: **Informativ** – Anfrage erhalten und in Bearbeitung.

2xx: **Erfolg** – Die Anfrage ist erfolgreich angekommen, verstanden worden und akzeptiert.

3xx: **Um-/Weiterleitung** – Der Anfragevorgang ist noch nicht vollständig abgeschlossen. Der Client muss noch weitere Schritte einleiten.

4xx: **Fehler auf der Clientseite** – Die Anfrage ist syntaktisch falsch oder ihr kann nicht entsprochen werden.

5xx: **Fehler auf der Serverseite** – Der Server hat eine korrekt gestellte Anfrage nicht bearbeitet.

[RFC2616]

Für den Single Sign-On-Vorgang wird der *Code 302 - Found* verwendet. Dieser Code bedeutet, dass die angeforderte Ressource vorübergehend nicht an der adressierten Stelle, sondern unter einem anderen URI zu finden ist. Der neue URI SOLLTE (SHOULD gem. [RFC2119]) dabei vom Server ins „Location Field“ des Antwortpakets geschrieben und somit dem Client mitgeteilt werden, damit dieser die Ressource anschließend mit dem neuen URI abrufen kann. Für den Client bedeutet eine Antwort des Servers mit einem Statuscode von 3xx nicht nur, dass die Ressource unter einem neuen URI zu finden ist, sondern auch, dass damit seine ursprüngliche Anfrage als noch nicht vollständig bearbeitet gilt. Ferner soll sich der Client den neuen URI aber nicht als dauerhaften Ersatz merken, sondern bei einer erneuten Anfrage wieder den ursprünglichen URI

[1] MIME steht für *Multipurpose Internet Mail Extensions* und wird in RFC 2045 beschrieben. Es sei aber ausdrücklich darauf hingewiesen, dass es sich im Falle von HTTP lediglich um MIME-artige und nicht um MIME-konforme Nachrichten handelt.

verwenden. Das ist für Weiterleitungen wichtig, bei denen das Weiterleitungsziel dynamisch veränderlich ist, wie etwa beim *Liberty Alliance Project* (vgl. 4.4). [RFC2616] [HoWa03]

Die Antwort des Servers SOLLTE darüber hinaus auch eine kurze Mitteilung im Hypertextformat enthalten, die über die Umleitung an die neue Zieladresse informiert, damit der Client (meistens ein WWW-Browser) sie dem Benutzer zu dessen Information präsentieren kann. Je nachdem, ob die ursprüngliche Anfrage des Clients mithilfe des HTTP-Kommandos GET oder HEAD gestellt wurde, ist diese Mitteilung dann in der Antwort des Servers enthalten (GET) oder nicht enhalten (HEAD). In letzterem Fall DARF der Server den *Message Body* nämlich NICHT übertragen (MUST NOT gem.[RFC2119]), sondern nur den Header; daher auch HEAD-Kommando. Hierin besteht auch der einzige Unterschied zwischen dem GET- und dem HEAD-Kommando, nämlich ob der *Message Body* in der Antwort des Servers enhalten ist oder nicht. Mit dem GET-Kommando fordert der User Agent eine Informationseinheit vom Server an, die sich an dem angegebenen URI befindet (bzw. befinden soll). Die Informationseinheit kann praktisch alles sein: ein HTML-Dokument, eine Datei oder die Ausgabe eines Prozesses, der durch den URI angesprochen werden kann. Beim HEAD-Kommando wird im Unterschied zum GET-Kommando lediglich der Header der Informationseinheit abgerufen, ohne dass dabei die gesamte Einheit übertragen wird. Ob der Client (der WWW-Browser) im Falle einer Umleitung die Ressource automatisch und ohne Zutun des Benutzers unter dem geänderten URI anfordern darf, hängt von der vorgesehenen Anfragemethode ab. Er darf es nur dann, wenn die Anfragemethode für den zweiten URI ein GET oder ein HEAD ist. [RFC2616]

2.2.2 HTML-Formulare und JavaScript

Eine andere Möglichkeit basiert auf der Benutzung von HTML-Formularen, wie man sie von der Anmeldung auf Websites, Kontaktformularen oder Gästebüchern kennt. In Verbindung mit JavaScript-Code, der das Abschicken des Formulars automatisch startet, können so Daten von einem Server zum anderen übertragen werden. Im Gegensatz zum HTTP-Redirect (2.2.1) sendet der Server als Antwort auf die Anfrage des Clients ein HTML-Dokument und einen HTTP-Statuscode 2xx (Erfolg). Die Anfrage ist damit gültig beantwortet und abgeschlossen. Das HTML-Dokument enthält dieses Mal Formularfelder, die bereits mit Werten belegt sind, welche als Parameter für den anderen Server bestimmt sind. Als Versandziel für die Formulardaten ist der andere Server angegeben. Außerdem können die Formularfelder im HTML-Code als unsichtbar definiert werden, sodass der Benutzer diese in seinem Browser gar nicht angezeigt bekommt, sondern stattdessen evtl. einen kurzen Hinweistext, dass sogleich eine Weiterleitung auf eine andere Website erfolgt. Ein im HTML-Code eingebettetes JavaScript ersetzt das Drücken des „Absenden“-Buttons durch den Benutzer und sorgt so dafür, dass die Daten zum anderen Server übertragen werden.
[HoWa03][RFC2616]

2.2.3 Cookies

Als dritte Möglichkeit sei noch die Verwendung von Cookies genannt, die allerdings nicht für den Einsatz in einem Single Sign-On-System für das World Wide Web geeignet ist. Dennoch soll hier kurz die Idee, die dahinter steckt dargelegt werden, weil sie sich unter Umständen für Single Sign-On-Systeme innerhalb einer Domäne (z.B. einer Firma) anbietet. Im Besonderen wird sie in den Spezifikationspapieren des Liberty Alliance Projects erwähnt. Cookies sind nach [RFC2965] ein Mechanismus für die Zustandsverwaltung von HTTP und dienen Webservern dazu, Informationen auf dem System des Benutzers zu speichern, etwa um den Benutzer wieder zu erkennen, wenn er die Website erneut aufruft, oder um bestimmte Einstellungen des Benutzers zu speichern wie z.B. Suchbegriffe, die der Benutzer auf dieser Website bereits bei einem seiner letzten Besuche verwendet hat. Solche Cookies nennt man *persistent*, weil sie über das Ende einer Browser-Sitzung hinaus auf der Festplatte des Benutzers gespeichert bleiben, und erst dadurch die Wiedererkennung eines ansonsten anonymen Benutzers möglich wird. Cookies, die am Ende einer Browser-Sitzung wieder gelöscht werden, nennt man *temporär* oder *sitzungsbezogen.* Die Idee bei der Verwendung für Single Sign-On besteht darin, dass der Identity Provider die Daten, die für den Service Provider bestimmt sind, in einem Cookie ablegt, welches der Service Provider dann ausliest. Auf diese Weise würde der Identity Provider in einem Cookie vermerken, dass sich der Benutzer bei ihm erfolgreich authentifiziert hat. Service Provider, die dies wissen wollen, könnten diese Information dann aus dem Cookie auslesen. Es besteht aber folgendes Problem, weswegen diese Technik nicht für Single Sign-On im gesamten World Wide Web geeignet ist. Cookies tragen den Domänennamen der erstellenden Site als eine Art Kennung und üblicherweise sind Webbrowser so konfiguriert, dass sie Cookies nur an Server der Domäne zurücksenden, deren Namen sie tragen. Dahinter steckt eine Sicherheitsüberlegung. Es sollen keine Inhalte eines Cookies von Domäne A durch einen Server von Domäne B ausgespäht werden können. Zwar lässt sich die Konfiguration des Webbrowsers in den meisten Fällen so ändern, dass diese Beschränkung aufgehoben wird, aber das würde gleichzeitig allen Webservern ermöglichen, sämtliche Cookies auszulesen, die beim Benutzer gespeichert sind. Daher scheidet der Einsatz von Cookies für den Datenaustausch zwischen Identity und Service Provider aus, sobald sich beide in unterschiedlichen Domänen befinden.

Dennoch finden Cookies im Kontext von Single Sign-On Verwendung, und zwar dann, wenn Provider lokale Sitzungsdaten oder -zustände in ihnen abspeichern. Das hat schließlich nichts mit dem eben diskutierten Problem zu tun, dass Identity und Service Provider keine Daten per Cookie austauschen können. Freilich hängt das Abspeichern lokaler Sitzungsdaten weiterhin von den Sicherheitseinstellungen im Client-Programm ab.
Zusammenfassend kann man sagen, dass Web Redirection nicht gerade eine ideale Architektur für verteilte Systeme ist. Alle drei genannten Techniken haben wohlbekannte Sicherheitslücken. Alle Daten gehen im Klartext durch das Netz. Daher können sie leicht abgehört und gefälscht werden. Verschlüsselung ist deshalb absolut notwendig (TLS oder SSL oder IPsec). Ferner stellen die Client-Programme ein fast nicht einzuschätzendes Sicherheitsrisiko dar. Denn während der

Kommunikationskanal zwischen zwei Providern durch das Client-Programm verläuft, besteht immer auch die Möglichkeit, dass die Information zwischengespeichert wird, wodurch sie auch wieder ausgelesen werden könnte. Besonders zu beachten: Die Information wird in diesem Fall sogar im Klartext abgespeichert, selbst wenn sichere Transportprotokolle benutzt werden! Dann ist die Übertragung zwischen Provider A und Client zwar geschützt, wie auch die Übertragung zwischen dem Client und Provider B, aber zwischendurch muss der Client die Daten „umpacken“, und das ist der bedenkliche Teil.
[HoWa03] [RFC2965]

2.3 Secure Sockets Layer (SSL) und Transport Layer Security (TLS)

Da es beim Single Sign-On um die Authentifizierung von Benutzern und damit um die Übertragung von Zugangsdaten geht, die aufgrund ihrer Bedeutung in besonderem Maße schützenswert sind, müssen Vorkehrungen zur Sicherung dieser Übertragung getroffen werden.
Der Zweck von SSL und TLS ist die Möglichkeit, Internetverbindungen zwischen zwei Programmen zu verschlüsseln und damit gegen Angriffe durch Mithörer, Manipulierer oder Fälscher zu schützen. [RFC4346]

> *“The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.”* [RFC4346], Seite 1

Der Zusammenhang zwischen SSL und TLS stellt sich wie folgt dar: TLS 1.0 und TLS 1.1 werden als die Nachfolger von SSL 3.0 angesehen. Sie wurden von der TLS Arbeitsgruppe der *Internet Enginieering Task Force (IETF)* auf der Basis von SSL 3.0 mit dem Ziel entwickelt, einen Standard für ein Sicherheitsprotokoll in der Transportschicht des IP-Protokoll-Stacks zu etablieren. Alle drei genannten Versionen können nicht untereinander operieren, da die Unterschiede zwischen ihnen zu groß sind, obgleich sie im Grundsatz ähnlich funktionieren. [TLSC06][RFC4346]
Da TLS 1.1 die modernste, abgeschlossene Variante ist, soll hier nur diese Version behandelt werden. TLS 1.2 befindet sich zum jetzigen Zeitpunkt in der Endphase seiner Entwicklung. Die hauptsächliche Änderung in dieser Version soll die Unabhängigkeit von den Verschlüsselungsalgorithmen MD5 und SHA-1 sein, weil diese Algorithmen durch die jüngste Forschung ganz bzw. teilweise kompromittiert wurden. Da die Spezifikationspapiere aber noch keinen finalen Status haben, sondern den Status „internet draft“, und sich somit auch immer noch etwas an der Spezifikation ändern kann, findet TLS 1.2 hier noch keine weitere Berücksichtigung. [TLSC06]

TLS ergänzt den unverschlüsselt arbeitenden TCP/IP-Protokollstapel in der Transportschicht um die Möglichkeit, die Nutzdaten aus der darüber liegenden Anwendungsschicht zu verschlüsseln – daher die Bezeichnung *Transport Layer Security*, bevor sie per *Transmission Control Protocol* (TCP) verschickt werden. TLS ist somit zwischen TCP und den Protokollen der Anwendungsschicht

anzusiedeln (Abbildung 2.1). Das ebenfalls in der Transportschicht arbeitende *User Datagram Protocol* (UDP) ist aufgrund seiner verbindungslosen Auslegung für die Verwendung durch TLS ungeeignet, da bei verbindungslosen Protokollen keine Kontrolle darüber stattfindet, ob ein verschicktes Datenpaket bzw. Datagramm tatsächlich den (gewünschten) Empfänger erreicht hat. Diese Verlässlichkeit ist aber eine Grundvoraussetzung für die verschlüsselte Kommunikation per TLS. [RFC4346][Wik07b]

<table>
<tr><td>Anwendungsschicht</td><td>HTTPS</td><td>SMTP</td><td>...</td></tr>
<tr><td>Transportschicht</td><td colspan="3">SSL/TLS
TCP</td></tr>
<tr><td>Netzwerkschicht</td><td colspan="3">IP</td></tr>
<tr><td>Netzzugangsschicht</td><td>Ethernet</td><td>Token-Ring</td><td>FDDI</td></tr>
</table>

Abbildung 2.1: SSL im TCP/IP-Protokollstapel

TLS nimmt einerseits Daten von Protokollen aus der Anwendungsschicht entgegen, verschlüsselt sie und leitet sie an TCP zur Übetrtragung an den Empfänger weiter. Umgekehrt nimmt es Daten von TCP entgegen, entschlüsselt sie und leitet sie an die in der Anwendungsschicht liegenden Protokolle weiter. Optional kann TLS die Daten auch komprimiert versenden. Der Einsatz von TLS im Protokoll-Stack ist für die höher liegenden Protokolle nicht transparent, d.h. die Protokolle der Anwendungsschicht müssen für die Verwendung von TLS ausgelegt sein, wenn sie es verwenden sollen, andernfalls würden sie direkt auf TCP zugreifen. So gibt es etwa vom oben beschriebenen HTTP noch die spezielle Variante HTTPS (S steht für Secure), die auf eine SSL/TLS-verschlüsselte Übertragung ausgelegt ist. [RFC4346]

TLS ist in zwei Schichten unterteilt, das Record Layer und das Handshake Layer. Das Handshake Layer ist für die Aushandlung der Sitzungsparameter zuständig, das Record Layer ist für die Übertragung der Nutzdaten aus der Anwendungsschicht zuständig, und das umfasst auch die Fragmentierung und Verschlüsselung beim Senden, sowie die Entschlüsselung und das wieder Zusammenfügen beim Empfangen von Daten.
Das TLS Record Protocol verleiht der Verbindung zwischen den Applikationen durch eine symmetrische Verschlüsselung *Vertraulichkeit*. Es verleiht ihr auch *Verlässlichkeit* durch eine Integritätsüberprüfung der übertragenen Nachrichten, welche mittels einer HASH-Funktion wie SHA oder MD5 codiert werden. Dabei wird in verwendet die HASH-Funktion einen geheimen, zwischen den Kommunikationspartnern ausgehandelten Schlüssel, der nur für die gerade laufende Sitzung gilt. Er wird *Message Authentication Code (MAC)* oder *Message Intergrity Code (MIC)* genannt und wird bei jedem neuen Verbindungsaufbau neu ausgehandelt. Er kann während einer laufenden Sitzung nicht erneut angefordert werden, was Spoofing entgegenwirkt, da ein Dritter nicht einfach in die Sitzung einsteigen und an sich ziehen kann.

Selbst wenn die Datenpakete abgehört werden sollten, sorgt die Verschlüsselung dafür, dass sie für den Angreifer unbrauchbar sind. Wenn der Angreifer Datenpakete abfängt, manipuliert und an den Empfänger weiterleitet, wird dieser die Manipulation dadurch bemerken, dass er die Daten nicht mehr „sinnvoll" entschlüsseln kann bzw. die (Nutz-)Daten nach der Entschlüsselung an zufälligen Stellen zufälligen Unsinn enthalten. Der Angreifer hätte ja nur die Möglichkeit, wahllos irgendwelche Bits oder Bytes zu verändern und könnte dabei kaum gezielt vorgehen. [RFC4346]

2.4 XML – eXtensible Markup Language

Single Sign-On beruht auf dem Austausch von Daten zwischen unterschiedlichen Systemen. Wann immer das geschieht ist es erforderlich, dass über das verwendete Datenformat Klarheit herrscht. Das heißt, der Empfänger muss wissen, nach welchen Regeln der Sender die Daten für den Versand zusammengestellt hat, damit er sie nach dem Empfang richtig interpretieren kann.

Beispiel „Datum":

> Die Ziffernfolge 01051970 steht bei einer Interpretation nach dem Schema TTMMJJJJ für den 1. Mai 1970, hingegen nach dem Schema MMTTJJJJ für den 5. Januar 1970. Es bedarf einer Klärung der Struktur der Ziffernfolge, damit zwei Kommunikationspartner tatsächlich denselben Tag im Kalender meinen.

Die eXtensible Markup Language (XML) wurde entwickelt, um einen Standard zu schaffen, nach dem die Struktur von Nachrichten bzw. Dokumenten definiert werden kann. Durch XML wird vorgegeben, wie die Definition der Dokumentenstruktur zu erfolgen hat, sodass jeder den Aufbau (Syntax) eines XML-Dokuments nachvollziehen kann. Durch den standardisierten Aufbau ist es möglich, solche Dokumente einfach mit einem Parser zu erfassen und automatisch Struktur und Inhalt voneinander zu trennen. Insbesondere der Austausch von Daten zwischen verschiedenen Softwaresystemen wird dadurch insofern erleichtert, als das Austauschformat unabhängig von den verwendeten Programmiersprachen und Systemplattformen der Kommunikationspartner definiert werden kann. Die im Zusammenhang mit Single Sign-On verwendeten Basistechnologien SOAP und SAML, die in den folgenden Abschnitten vorgestellt werden, sind in XML definiert. [ACKM04][Kun03]

> *„ […] XML provides a standard way to define the structure of documents that is suitable for automatic processing. "* [ACKM04], Seite 118

XML ist eine Metasprache für sogenannte *Markup Languages* (Auszeichnungssprachen), deren wohl bekanntester Vertreter HTML ist. Durch XML werden die Regeln zum strukturellen Aufbau von XML-Dokumenten festgelegt. Die XML-Dokumente bestehen aus mehreren Informationsobjekten: dem *Prolog*, der die zugrunde liegende XML-Version angibt, einem *Root-Element*, in dem

die Nutzdaten hierarchisch strukturiert untergebracht sind, und einer beliebigen Anzahl von *Leerzeichen*, *Kommentaren* oder *Prozessanweisungen* (engl.: *processing instructions*) an die Applikation, die das Dokument liest. [W3C06]

Der Prolog

Im Prolog werden die Regeln angegeben, nach denen das XML-Dokument erstellt worden ist, und nach denen es zu interpretieren ist. Es SOLLTE mit einer *XML declaration* beginnen, die die verwendete XML-Version angibt:

```
<?xml version="1.0"?>
```

[W3C06]

Darüber hinaus ist ein Dokument nur dann gültig (*valid*), wenn es

a) im Prolog auf eine Dokumenttypdeklaration verweist (*document type declaration* – DTD), und
b) strikt den dort gemachten Vorgaben entpricht.

[W3C06]

Über die XML-Deklaration und die Dokumenttypdeklaration hinaus kann bereits der Prolog Kommentare und Prozessanweisungen (s.u.) enthalten. Sie brauchen nicht erst alle am Ende des XML-Dokuments zu stehen. [W3C06][Kob99]

Document Type Definition - DTD

Die DTD legt die Struktur eines Dokumenttyps fest. Ein Dokumenttyp ist eine Klasse ähnlich aufgebauter Dokumente. Die DTD enthält die Definitionen der in einem Dokumenttyp vorgesehenen Elemente und deren Auszeichnungen (*Markups*), legt die Reihenfolge und Verschachtelung, sowie die Typen der verwendbaren Attribute fest. In der DTD können auch Verweise auf weitere Sammlungen von Markup-Definitionen eingebracht werden. Sie werden dann ähnlich wie Funktionsbibliotheken in den Programmiersprachen C/C++ oder Java eingebunden. Dadurch ist es möglich, bereits bestehende DTDs als Grundlage für eine neue DTD zu verwenden, und somit einzelne Klassen von XML-Dokumenten zu verändern oder zu erweitern. Erst im Zusammenhang mit der DTD, welche die Grammatik zur korrekten Erstellung enthält, erhält ein XML-Dokument seine Bedeutung. [W3C06] [Wiki07-a]

Die Dokumenttypdeklaration für ein HTML-Dokument im XML-Format lautet beispielsweise so:
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN//">`

[Kob99] Seite 179

XML Schema

Eine weitere Sprache, um die Struktur von XML-Dokumenten zu definieren, ist *XML Schema*. Gegenüber der oben erwähnten *Document Type Definition* (DTD) übertrifft XML Schema deren Möglichkeiten um die Verwendung eigener Namensräume (*name spaces*) und komplexe Datentypen. XML Schemata werden selbst in XML beschrieben und können als Alternative zur DTD verwendet werden. Die für diese Arbeit relevanten Basis-Technologien SOAP, SAML und die Protokolle der Liberty Alliance sind in der Definitionssprache *XML Schema* definiert. [Wik07]

Das Root-Element

Jedes XML-Dokument enthält genau ein Root-Element, das auch Dokument-Element genannt wird. Es umschließt alle weiteren Elemente des XML-Dokuments wie in einem Klammernpaar. Die anderen Elemente sind innerhalb des Root-Elements hierarchisch angeordnet und funktionieren selbst wieder wie Klammerausdrücke, die korrekt ineinander verschachtelt sein müssen. Beginn und Ende eines Elements müssen also innerhalb eines übergeordneten Elements liegen. Jedes Element wird durch ein Start-Tag und ein End-Tag begrenzt, die zusammen wie ein Klammernpaar funktionieren:

`<markup>` ***`Inhalt`*** **`</markup>`**

Der Start-Tag kann ggf. noch mit Parametern versehen werden. Der End-Tag wird durch den Schrägstrich hinter der öffnenden spitzen Klammer charakterisiert. In manchen Fällen kann es sein, dass ein Element vorhanden sein darf, aber keinen Inhalt braucht, höchstens Attribute, die direkt im Start-Tag untergebracht werden. Solche „leeren" Elemente, die also auch keine weiteren Elemente enthalten, können statt mit einem öffnenden und einem schließenden Tag auch abkürzend mit einem einzigen Tag gekennzeichnet werden, welches als Start- und End-Tag in einem gilt. Es hat zur Kennzeichnung einen Schrägstrich unmittelbar vor der schließenden spitzen Klammer, also:

`<markup/>`.

Das Root-Element bildet den Stamm in der hierarchischen Anordnung. Im Folgenden nun ein Beispiel für eine korrekt verschachtelte Anordnung von XML-Elementen:

```
1:    <root-element>
2:          <element-auf-der-zweiten-ebene>
3:                <element1-auf-der-dritten-ebene>
4:                </element1-auf-der-dritten-ebene>
```

```
5:                  <element2-auf-der-dritten-ebene>
6:                        <element-auf-der-vierten-ebene>
7:                        </element-auf-der-vierten-ebene>
8:                  </element2-auf-der-dritten-ebene>
9:                  <element3-auf-der-dritten-ebene/>
10:           </element-auf-der-zweiten-ebene>
11:     </root-element>
```

Die Elemente im vorgenannten Beispiel enthalten zunächst nur weitere Elemente der nächsten Ebene, aber keinen eigentlichen Inhalt, und es werden auch noch keine Attribute festgelegt.

Attribute werden – wie oben erwähnt – im Start-Tag untergebracht. So hat die bereits oben erwähnte XML-Deklaration ein Attribut `version`, mit dem die Nummer der XML-Version übergeben wird, in diesem Fall ist es die Versionsnummer 1.0. Attribute werden immer nach dem Schema Attributname, Gleichheitszeichen, Wert in Anführungszeichen übergeben:

`<?xml version="1.0"?>`.

Prozeßanweisungen

XML-Dokumente können Verarbeitungsanweisungen für Programme / Applikationen enthalten. Diese Anweisungen müssen beim Lesen vom Parser an die Applikation weitergeleitet werden, die durch eine bestimmte Zielangabe (*PITarget*) identifiziert wird. So kann das verarbeitende Programm beispielsweise dazu angewiesen werden, ein bestimmtes Stylesheet nachzuladen, das für die Anzeige der im XML-Dokument enthaltenen Daten entworfen wurde. [W3C06][Kob99]

Kommentare

Kommentare werden in bestimmte Tags geklammert, wobei `<!--` als linke Klammer und `-->` als rechte Klammer dient. Die Kommentare dürfen überall im XML-Dokument stehen, nur nicht innerhalb der spitzen Markup-Klammer `<markup> bzw. </markup>`. [Kob99]

2.5 SOAP

SOAP ist ein auf XML basierendes Protokoll, welches einen Rahmen für den Zugriff auf und die Interaktion zwischen *Web Services* bereitstellt, indem es den Austausch von Daten und *Remote Procedure Calls* (RPC) zwischen den beteiligten Systemen ermöglicht. Dabei wird XML zur Repräsentation der Daten und zur Strukturierung der Nachrichten eingesetzt, während zur Übertragung der Nachrichten die Protokolle der Transport- und Anwendungsschichten verwendet werden. Der Name SOAP wird seit der aktuellen Version 1.2, die im Jahre 2003 erschien, als Eigenname geführt. Er ging aus der Abkürzung für *Simple Object Access Protocol* hervor. SOAP wurde entwickelt, um verschiedene Applikationen über das Internet miteinander verbinden zu können,

bzw. einen Standard für den Daten- oder Nachrichtenaustausch zwischen Web Services zu etablieren. Die Version 1.0 von SOAP hat Microsoft entwickelt, am 30.11.1999 vorgestellt und darüber hinaus bei der Internet Engineering Task Force (IETF) eingereicht, um den Vorschlag zum Internet-Standard erheben zu lassen.
Bei der Entwicklung gab es mehrere wichtige Punkte, u.a.:

- Das Funktionieren durch Firewalls hindurch.
- Nicht-Existenz standardisierter Protokolle.
- Die Interaktion zwischen Systemen sollte möglichst lose gestaltet werden, um möglichst flexibel Änderungen an einem der beteiligten Systeme durchführen zu können.

[ACKM04] [Wiki07-b] [Kuri99]

SOAP-Nachrichten

Die SOAP-Nachricht wird ähnlich wie ein Briefumschlag (engl.: *envelope*) verwendet, der eigentliche Inhalt steht in einem Element innerhalb des Umschlages, dem *SOAP-Body*. SOAP setzt auf XML auf und daher wird der *SOAP-Envelope* zum Root-Element in einem XML-Dokument. Innerhalb des SOAP-Envelope muss es dann mindestens noch das SOAP-Body-Element geben, in dem die Information an den Empfänger enthalten ist; vergleichbar einem Stück Briefpapier, auf dem eine Nachricht steht. Innerhalb des SOAP-Bodys dürfen auch mehrere Blöcke auf derselben hierarchischen Ebene, direkt unterhalb des SOAP-Body Elements, enthalten sein. Dem SOAP-Body kann innerhalb des SOAP-Evelope noch ein *SOAP-Header* vorangestellt werden, der auch wieder aus mehreren Blöcken bestehen kann. Die darin enthaltene Information darf von allen Knoten, über die die Nachricht transportiert wird, gelesen werden und kann beispielsweise Anweisungen an diese Knoten enthalten, wie sie mit der ganzen Nachricht verfahren sollen. Die Aufteilung in Header und Body folgt dabei dem Beispiel anderer Kommunikationsprotokolle, wobei im Header Informationen zum Versand enthalten sind, die von allen Zwischenstationen gelesen und verarbeitet werden können oder sogar sollen, und im Body die Inhalte für den letztendlichen Adressaten (*Ultimate Receiver*) stehen. [ACKM04]

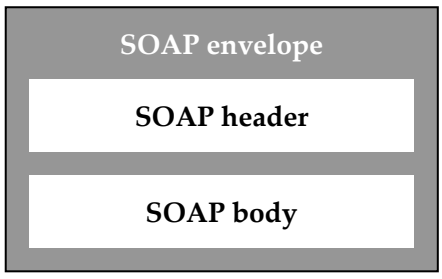

Abbildung 2.2: Aufbau einer SOAP-Nachricht

Regeln

Die Regeln darüber, wie der Transport und die Verarbeitung von SOAP-Nachrichten zu handhaben sind, werden ebenfalls im SOAP-Standard definiert. Die Spezifikation sieht hier drei Rollen vor, die ein SOAP-Knoten einnehmen kann: *None, Next* und *UltimateReceiver*. Die Blöcke im SOAP-Header können sich speziell an diese Rollenprofile richten, wobei jeder SOAP-Knoten für sich auswerten bzw. entscheiden muss, in welcher Rolle er sich im Bezug auf die jeweilige SOAP-Nachricht befindet.
Blöcke, die an die Rolle *None* gerichtet sind, sollen von keinem Knoten ausgewertet werden, sie können aber Informationen enthalten, die zur Verarbeitung anderer Header-Blöcke gebraucht werden.
Blöcke, die an die Rolle *UltimateReceiver* gerichtet sind, sind ausschließlich für den Adressaten bestimmt und dürfen von anderen Knoten nicht ver- oder bearbeitet werden.
Blöcke, die an die Rolle *Next* gerichtet sind, dürfen (aber müssen nicht) von jedem Knoten des Nachrichtenpfades verarbeitet werden. Es ist allerdings möglich einem Block das Attribut *mustUnderstand* zu geben. Dann ist es für jeden Knoten, der dem angesprochenen Rollenprofil entspricht, obligatorisch diesen Block zu verarbeiten.
Der *SOAP-Body* bedarf keiner expliziten Rollen-Zuweisung mehr, er ist per definitionem ausschließlich für den UltimateReceiver bestimmt. Er muss auch vom UltimateReceiver bearbeitet werden, oder dieser muss eine Fehlermeldung generieren und zurückschicken. [ACKM04]

Interaktionsmodi: Dokumentenaustausch und Remote Procedure Call

Im Wesentlichen kann SOAP auf zwei Arten verwendet werden, um zwei Applikationen miteinander interagieren zu lassen. Als erstes gibt es die Möglichkeit des *Dokumentenaustauschs*, bei dem die beiden Applikationen eine gemeinsame Dokumentenstruktur auf der Basis von XML verwenden. Die Vorgaben zur Struktur des Formulars macht der Server, denn er muss es nach erhalt auswerten und die enthaltenen Angaben richtig zuordnen können. Der Client stellt dann seine Anfrage an den Server, indem er quasi ein Formular, einen Vordruck, ausfüllt und im Body einer SOAP-Nachricht an den den Server sendet. Nachdem der Server die Anfrage ausgewertet und bearbeitet hat, sendet er das Ergebnis in einem ebenfalls XML-basierenden Antwortformular im Body einer SOAP-Nachricht an den Client zurück. Sollte bei der Bearbeitung ein Fehler aufgetreten sein, sendet der Server anstelle eines Ergebnisses eine Fehlermeldung auf die gleiche Weise an den Client zurück. Die zweite Möglichkeit ist die Nutzung von Remote Procedure Calls. Sie sind ein wesentliches Mittel zur Benutzung entfernt liegender Dienste in verteilten Systemen. SOAP schafft hier Konventionen für das Format von SOAP-Nachrichten, die dem Aufruf entfernt liegenden Prozeduren dienen, und umgekehrt für das Format der Antworten, die zurück an den Client geschickt werden.
Der entscheidende Vorteil bei der Verwendung von SOAP für RPCs liegt in der Unabhängigkeit von der fraglichen Kompatibilität der Middleware-Systeme von Client und Server. Mit SOAP wird auf jedem beteiligten System eine Schicht eingeführt, die die Umsetzung von SOAP-

Nachrichten in lokale Prozeduraufrufe vornimmt. Der Client muss also nicht mehr die Syntax der Middleware auf dem Server kennen, um dort RPCs absetzen zu können. Vielmehr sendet er eine entsprechend formatierte SOAP-Nachricht, die auf der Serverseite ausgewertet und von der SOAP-Schicht in einen lokalen Prozeduraufruf umgesetzt wird. [ACKM04]

Transportprotokolle

In der SOAP-Spezifikation wird das *SOAP Protocol Binding Framework* definiert, welches die Rahmenbedingungen für die Bindung von SOAP-Nachrichten an ein Transportprotokoll festlegt. SOAP hat keine eigene Transportfunktionalität, sondern bedient sich dafür bereits existierender Protokolle, z.B. HTTP oder SMTP. So wird z.B. für die HTTP-Bindung festgelegt, dass für einen RPC die SOAP-Nachricht in ein HTTP-Paket eingebettet wird (vgl. Abbildung 2.3) und mit dem HTTP-POST-Kommando versendet wird. Die Adressierung des Ultimate Receivers findet automatisch nach den Regeln des verwendeten Transport-Protokolls statt und ist beim HTTP eine andere als beim SMTP. Für letzteres Bedarf es einer E-Mail-Adresse, für ersteres einen URL. [ACKM04] [W3C07-a]

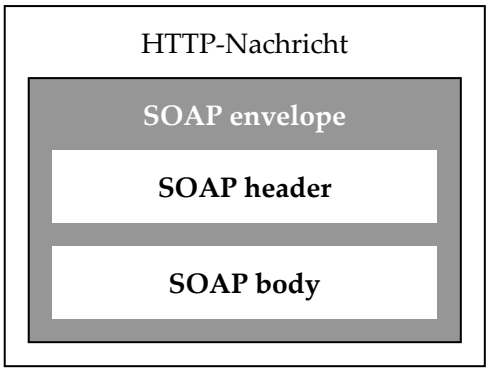

Abbildung 2.3: SOAP-Nachricht in HTTP eingebettet

2.6 WS-Security

Um dem Nachrichtenaustausch von SOAP Integrität und Vertraulichkeit zu verleihen, haben die Firmen IBM, Microsoft und Verisign *WS-Security* (kurz für *Web Services Security*) als Erweiterung entwickelt. Dabei wird dem SOAP-Header eine weiterer Block hinzugefügt, der die Bezeichnung *Security* hat und eine Signatur tragen kann. Dieser Block enthält wiederum Elemente wie *UsernameToken* oder *BinarySecurityToken*, die Sicherheitsinformationen wie Benutzername und Passwort bzw. Schlüssel und Sicherheitszertifikate enthalten. Im Unterschied zum einfachen SOAP lässt sich durch diese Erweiterung die verschlüsselte Übertragung von Sicherheitsinformationen zwischen den Endknoten der SOAP-Kommunikation auf der Anwendungsebene realisieren. Zwar können SOAP-Nachrichten mit unterschiedlichen Protokollen der Transportschicht übertragen werden, also auch mit HTTPS, der verschlüsselten Variante des HTTP, aber diese Art der Verschlüsselung findet auf der Transportebene statt und daher eine andere Qualität. HTTPS ist ein

Sicherheitsmechanismus, der nicht Zwischen den Endpunkten der Kommunikation wirkt, sondern zwischen den einzelnen *hops*[2], die an der Übermittlung der Nachrichten beteiligt sind. Jede Zwischenstation kann die Nachrichten entschlüsseln und wieder verschlüsseln und zwischenzeitlich den entschlüsselten Inhalt im Klartext einsehen.
Ferner erfasst die Verschlüsselung bei HTTPS immer die gesamte Nachricht, die der Transportschicht übergeben wird. Sicherheitsmechanismen, die dagegen auf Anwendungsebene wirken, erlauben es, auch nur bestimmte Teile eine Nachricht zu verschlüsseln. Dadurch ist es möglich, etwa nur den Body einer SOAP-Nachricht zu verschlüsseln, dessen Inhalt nur für den endgültigen Empfänger bestimmt ist, während der Header unverschlüsselt bleibt, damit er von den SOAP Intermediaries (Zwischenstationen) verarbeitet werden kann. Die Granularität, welche Teile einer Nachricht verschlüsselt werden sollen, kann praktisch beliebig gewählt werden. So lässt sich die Verschlüsselung wahlweise auch nur auf besonders sensible Informationen wie z.B. Kreditkarteninformationen anwenden, während andere Informationen innerhalb derselben SOAP-Nachricht unverschlüsselt bleiben können.
[ACKM04] [Wilk02]

2.7 SAML – Security Assertions Mark-up Language

Die *Security Assertion Markup Language* (SAML) ist ein auf XML basierendes Rahmenwerk für die Beschreibung von Sicherheitsinformationen und deren Austausch zwischen Online-Partnern, das vor dem Hintergrund von Single Sign-On, Verknüpften Identitäten (engl. federated identities) und Web Services entstanden ist. Die Sicherheitsinformationen werden in Form von Zusicherungen (engl. *assertions*) zwischen Anwendungen ausgetauscht, die in unterschiedlichen Sicherheitsdomänen laufen. Im SAML-Standard sind die Syntax und die Regeln für Anfrage, Erstellung, Übermittlung und die Verwendung der Zusicherungen festgelegt. [Rag+06]

An einer SAML-Interaktion sind mindestens zwei Parteien beteiligt: Eine, die der anderen Partei die Zusicherung macht, und die andere, welche der ersten Partei vertraut, von der die Zusicherung stammt.
SAML definiert eine Reihe von Rollen, die die SAML-Parteien bei der Interaktion untereinander annehmen können. So gibt es für das Multi Domain Single Sign-On (MDSSO) – was dem Single Sign-On im Kontext dieser Arbeit entspricht – die Rollen *Identity Provider* (IDP) und *Service Provider* (SP). Dass diese Rollenbezeichnungen genau denen aus dem Liberty Alliance Project (LAP) entsprechen, beruht darauf, dass Bestandteile aus dem *Identity-Federation Framework* (ID-FF) des LAP in die Entwicklung von SAML 2.0 eingeflossen sind; vgl. 4.1. [Wis+05][Rag+06]

Die vier Eckpfeiler von SAML sind Zusicherungen (*assertions*), Protokolle (*protocol messages*), Bindungen (*bindings*) an andere Protokolle für den Transport und anwendungsbezogene Profile (*pro-*

[2] Hop – Anfangs-, Zwischen- oder Endstation bei der Übertragung von Nachrichten per TCP/IP

files). Darüber hinaus gibt es noch zwei weitere Konzepte, die dem Aufbau und der Verwendung einer SAML-Umgebung dienlich sind, nämlich Metadaten (*metadata*) und das Konzept des Authentisierungskontexts (*authentication context*). [Rag+06]

2.7.1 Zusicherungen

Zusicherungen enthalten ein oder mehrere Statements sowie allgemeine Informationen, die entweder alle Statements gleichermaßen betreffen oder die gesamte Zusicherung als Einheit. Zusicherungen werden normalerweise in einer Antwortnachricht entsprechend einem der verschiedenen SAML-Protokolle übertragen, wobei diese Protokollnachrichten ihrerseits in ein Transportprotokoll eingebettet sind. Abbildung 2.4 zeigt die Einbettung einer SAML-Antwort in eine SOAP-Nachricht, welche dann zur Übertragung wie in Abbildung 2.3 in ein Transportprotokoll eingebettet wird. Es gibt drei Arten von Statements, die im Rahmen von Zusicherungen übertragen werden können, nämlich *Authentisierungs-Statements*, mit denen angegeben wird, zu welchem Zeitpunkt und mit welcher Methode ein Benutzer authentisiert wurde, *Attribut-Statements*, die zur Übermittlung beliebiger Attribute wie z.B. „hat eine goldene Kreditkarte“ dienen, und *Authorisierungs-Statements*, die angeben, was der Benutzer machen bzw. was er nicht machen darf – z.B. bestimmte Artikel kaufen oder bestimmte Datenbereiche einsehen. [Rag+06]

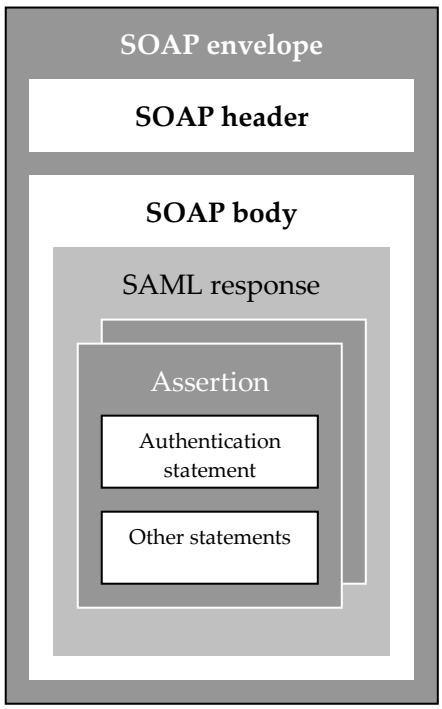

Abbildung 2.4: SAML-Antwort in einer SOAP-Nachricht

2.7.2 Protokolle

Protokolle dienen dem geordneten Ablauf bei der Datenübertragung. Für die SAML sind sechs Frage-Antwort-Protokolle definiert:

1. Das *Authentication Request Protocol* (Protokoll zur Anforderung einer Authentisierung) definiert, wie Prinzipal eine Zusicherung über seine eigene Authentisierung anfordern kann, die auch um optionale Attribut-Statements erweitert werden kann. *Das Web Browser Single Sign-On Profil* (siehe unten), welches ebenfalls für SAML definiert ist, benutzt dieses Protokoll.
2. Das *Single Logout Protocol* stellt einen Mechanismus zur Verfügung, durch den alle Sitzungen des Prinzipals auf einmal beendet werden können. Der Single Logout kann dabei entweder vom Prinzipal selbst, oder von einem der Service oder Identity Provider initiiert werden.
3. Das *Assertion Query and Request Protocol* stellt zwei Modi zur Verfügung. Der Request-Modus dient dazu, bei einer Zusicherungen ausstellenden Stelle eine bestimmte, bereits existierende Zusicherung anzufordern. Die Zusicherung, um die es dabei geht, wird über ihre *assertion ID* referenziert. Im Query-Modus können neue und bestehende Zusicherungen unter Angabe eines bestimmten Subjekts (i.d.R. ein Benutzer) in der gewünschten Form (statement type) angefordert werden.
4. Es gibt einen Mechanismus, bei dem nicht die Zusicherung direkt verschickt wird, sondern stattdessen ein Artefakt, ein künstlicher, willkürlicher Wert mit einer bestimmten Länge. Der Empfänger erhält das Artefakt typischerweise nicht direkt vom Ersteller, sondern indirekt per Weiterleitung (z.B. HTTP-Redirect). Mit Hilfe des *Artifact Resolution Protocol* wendet sich der Empfänger an den Ersteller des Artefakts und bittet ihn, das Artefakt zu dereferenzieren und stattdessen nun die eigentliche Nachricht (z.B. eine Zusicherung) zu übertragen.
5. Das *Name Identifier Protocol* dient der Verwaltung von verknüpften Identitäten zwischen einem Service Provider und einem Identity Provider. Beide Provider handeln untereinander bei der Identitätsverknüpfung einen Bezeichner aus, mit dem sie den Benutzer gegenseitig eindeutig referenzieren können. Das Name Identifier Protocol ermöglicht es, den Bezeichner auf Anforderung eines der beteiligten Provider zu ändern und neu zu setzen oder ganz zu löschen, was dann einer Beendigung der Identitätsverknüpfung entspricht.
6. Jede Verknüpfung zweier Identitäten zwischen einem Service Provider (SP) und einem Identity Provider (IDP) erzeugt einem Bezeichner wie unter Punkt 5 beschrieben. Die Bezeichner sind individuell ausgehandelt und gelten damit nur zwischen den beiden beteiligten Providern. Will ein SP mit einem IDP bzgl. eines Benutzers interagieren, benutzt er bei der Kommunikation den ausgehandelten Bezeichner. Will der SP aber mit einem anderen SP bzgl. desselben Benutzers kommunizieren, kann er das zunächst nicht über einen Bezeichner machen, weil zwischen den beiden SP gar kein gemeinsamer Bezeichner ausgehandelt worden ist. Stattdessen kann der erste SP den IDP per *Name Identifier Mapping Protocol* nach dessen gemeinsamen Bezeichner mit dem zweiten SP fragen, wobei die

Antwort nur in Abhängigkeit von entsprechend eingerichteten Zugriffsrechten erfolgen darf.

[Rag+06]

2.7.3 Bindungen

Die Bindungen legen fest, wie die verschiedenen SAML Protokollnachrichten mithilfe von Transportprotokollen übertragen werden. SAML greift hier auf bestehende Protokolle wie HTTP und SOAP zurück und definiert in Version 2.0 folgende sechs Bindungen, von denen sich für die ersten drei entsprechende Profile für das SSO in den Beschreibungen zum LAP wiederfinden (vgl. 4.4) [Rag+06]:

1. *HTTP Redirect Binding*: SAML Nachrichten können per HTTP Redirect (Statuscode 302 – vgl. 2.2) transportiert werden. Hierfür werden sie als Parameter komplett in den URL im *Location header* – also in die Zieladresse für die Weiterleitung – geschrieben. Dabei ist jedoch zu beachten, dass es zwar theoretisch keine Längenbegrenzung für URLs gibt, so dass beliebige SAML Nachrichten als Parameter übergeben werden können, es aber in der Praxis trotzdem unvorhersagbare Längenbeschränkungen, z.B. durch die Implementationen der verwendeten Programme, geben kann. Die folgenden beiden Bindungen können mit dem HTTP Redirect Binding kombiniert werden und stellen Lösungen zu dem eben genannten Längenproblem dar. Das HTTP Redirect Binding ist für Fälle konstruiert, in denen der SAML-Requester und der SAML-Responder nur indirekt z.B. durch den Browser eines Benutzers miteinander kommunizieren können. [CHK+05]

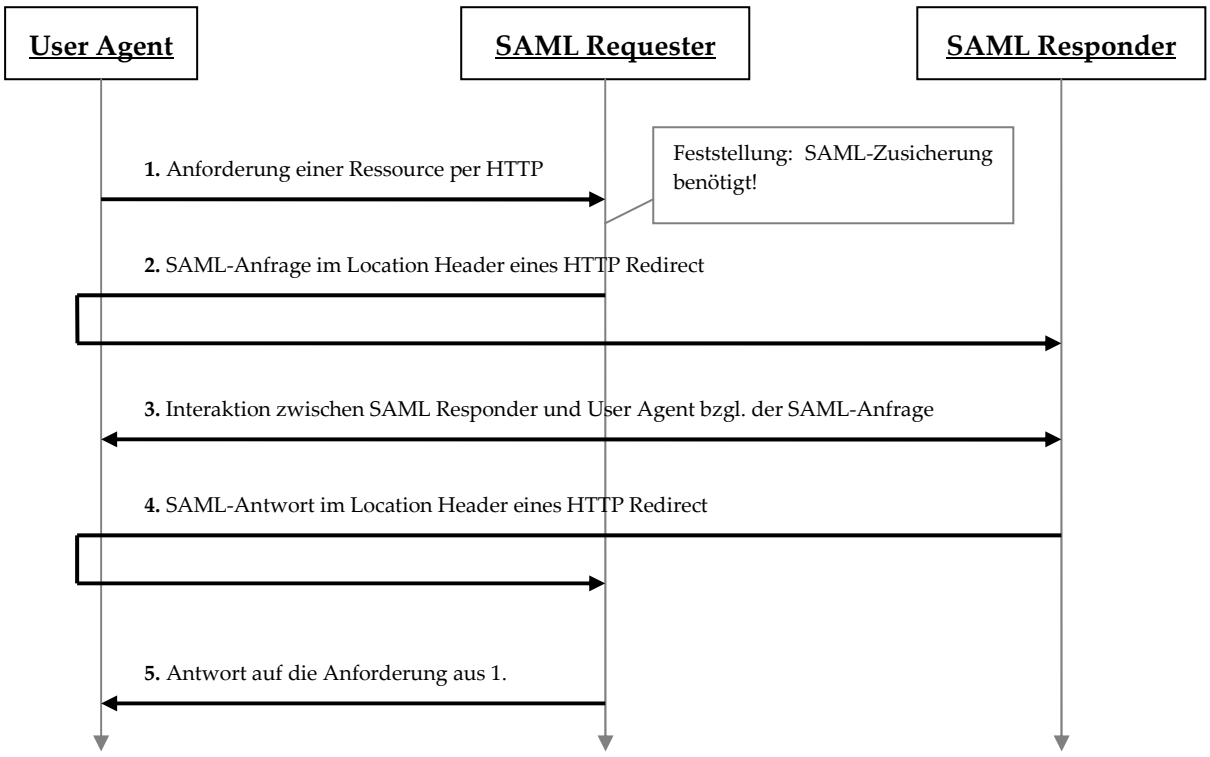

Abbildung 2.5: SAML-Nachrichtenaustausch per HTTP-Redirect

2. *HTTP POST Binding*: Wie beim HTTP Redirect Binding kommunizieren SAML-Requester und SAML-Responder auch beim HTTP Post Binding nur indirekt miteinander, und zwar ebenfalls durch den User Agent (z.B. Web-Browser) hindurch. Die Abfolge der Kommunikationsschritte ist gleich, was sich aber ändert, ist die Art und Weise, wie die Daten durch den User Agent geschleust werden. Beim HTTP POST Binding kommen XHTML-Formulare zum Einsatz. Im 2. Übertragungsschritt (vgl. Abbildung 2.6) wird die XML-Darstellung der SAML-Anfrage in Base64-Kodierung in ein verstecktes, und damit im Browser des Benutzers nicht angezeigtes HTML-Formularfeld geschrieben. Dieses Feld MUSS die Bezeichnung `SAMLRequest` bzw. `SAMLResponse` haben, je nachdem, ob es sich um eine Anfrage an oder um eine Antwort vom SAML-Responder handelt. Das `action` Attribut muss auf die HTTP Schnittstelle des eigentlichen Empfängers, also den SAML-Responder oder den SAML-Requester, verweisen. Das `method` Attribut muss auf den Wert POST gesetzt werden. Wenn der Web-Browser des Benutzers nun als Antwort auf seine eigentliche Anfrage (z.B. Abruf einer bestimmten Web-Seite) das ausgefüllte XHTML-Formular bekommt, sendet er es aufgrund der POST-Anweisung an die in action genannte Adresse, wodurch der erste Schritt der indirekten Kommunikation zwischen SAML-Requester und –Responder abgeschlossen wird. Im folgenden geht es – wie beim HTTP-Redirect Binding – mit der Interaktion zwischen SAML-Responder und User Agent weiter, und die indirekte Antwort des SAML-Responders an den –Requester erfolgt in umgekehrter Richtung wiederum per HTTP POST. Zum Schluss erhält der User Agent die Antwort auf seine ursprüngliche HTTP-Anfrage, die er eingangs gestellt hatte. [CHK+05]

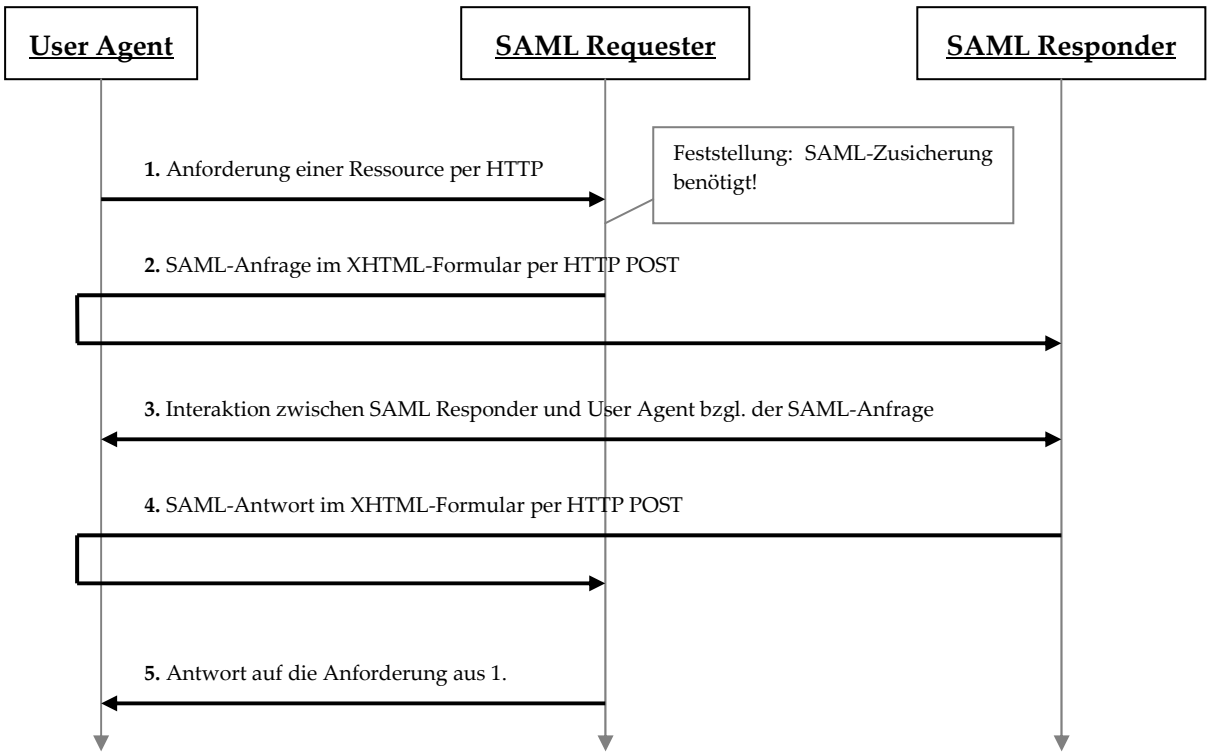

Abbildung 2.6: SAML-Nachrichtenaustausch per HTTP POST

3. *HTTP Artifact Binding*: Im Grundsatz funktioniert diese Bindung nur mit einer der beiden zuvor genannten zusammen. Ein Artefakt, das als Platzhalter zu verstehen ist, ersetzt beim Datenaustausch per HTTP-Redirect oder HTTP-POST die eigentliche SAML-Nachricht (Zusicherung). Der Empfänger des Artefakts muss sich in einem Zwischenschritt direkt an den Versender des Artefakts wenden und ihn zur Zusendung der eigentlichen SAML-Zusicherung auffordern. Abbildung 2.7 zeigt den erweiterten Ablauf. Die Schritte 3, 4, 7 und 8 sind hinzugekommen und illustrieren die zusätzlichen Zwischenschritte bei der Kommunikation zwischen SAML-Requester und –Responder. [CHK+05]

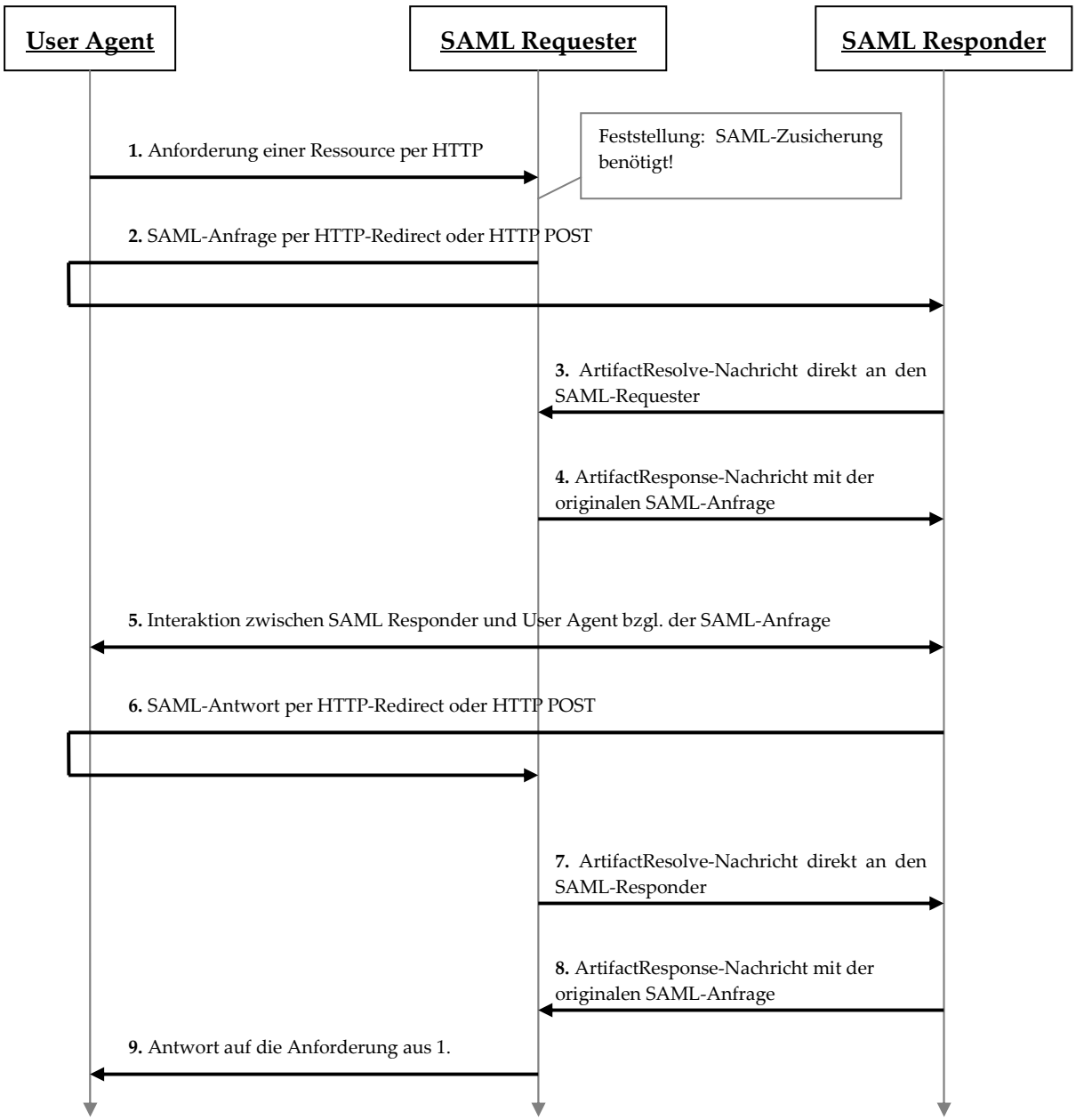

Abbildung 2.7: SAML-Nachrichtenaustausch mit Artefakt

Die anderen drei Bindungen *SAML SOAP Binding*, *Reverse SOAP (PAOS) Binding* und *SAML URI Binding* sollen hier nicht weiter erklärt werden, da sie für das Verständnis der im folgenden beschriebenen Single Sign-On-Vorgänge nicht von Bedeutung sind. Über die sechs erwähnten Bindungen hinaus können noch weitere, eigene Bindungen definiert werden. Die dafür festgelegten Richtlinien werden in [CHK+05] beschrieben.

3 Microsoft .NET Passport und Windows Live ID

Mit .NET (sprich: dot net), das während seiner Entwicklung die Bezeichnung *Next Generation Windows Services* (NGWS) trug, stellte Microsoft für PCs und mobile Endgeräte eine einheitliche Plattform für Internetanwendungen bereit. Es wurde im Sommer 2000 auf der Professional Developer Conference (PDC) vorgestellt. .NET Passport stellt einen Baustein dieser Strategie dar. Inzwischen heißt die aktuelle Internetstrategie Windows Live und im Zuge dessen gibt es mit Windows Live ID auch einen Nachfolger für .NET Passport. [Kauf06][Wes02]

3.1 .NET Passport

Der von Microsoft betriebene Anmeldedienst .NET Passport (kurz: Passport) kann von Web Sites zur Benutzer-Authentifizierung benutzt werden. Er wurde 1999 eingeführt und 2006 durch *Windows Live ID* (vgl. Abschnitt 3.2) abgelöst. Er wurde als Lösung zu dem Problem konzipiert, das Benutzer haben, wenn sie sich für immer mehr verschiedene Zugänge zu Rechnern, Programmen und Dienstleistungen immer neue Benutzernamen und Passwörter merken müssen. .NET Passport gehört zu den *.NET MyServices*, die eine der vier Säulen der gesamten *.NET-Strategie* von Microsoft bilden. Um *.NET Passport* bzw. *Windows Live ID* nutzen zu können benötigt man eine E-Mail-Adresse, die als Benutzername verwendet wird, und ein Passwort, das der Benutzer bei der Anmeldung festlegt und über dessen Sicherheit (Komplexität) er bei der Eingabe informiert wird. Nach der Anmeldung kann der Benutzer grundsätzlich alle Funktionen des Passport-Dienstes nutzen, allerdings sind für die Verwendung mancher Funktionen weitere Angaben notwendig. [KoRu00] [Wes02] [MS03]

> *„Hinter Passport steht die Idee, für (möglichst) alle Dienste im Web nur einen Anmeldenamen und ein Passwort zu benötigen." [Wes02], Seite 137*

Das Konzept basierte auf der zentralen Speicherung sämtlicher Benutzerdaten innerhalb des Passport eines Benutzers, die der Passport Server je nach Bedarf an den Server eines Dienstleisters überträgt, ohne, dass der Benutzer diese Daten erneut beim Dienstleister eingeben muss. So können in Passport beispielsweise die komplette Anschrift des Benutzers oder auch seine Kreditkartennummer, Kleidergröße usw. gespeichert werden, um so bei Einkäufen per Internet diese Daten vom Passport Server direkt an den Online-Händler zu übertragen. Der Benutzer wird vor einer solchen Datenübertragung gefragt, welche Teile der gespeicherten Informationen er an den Dienstleister übertragen lassen will. [KoRu00][Wes02]

Nachdem .NET Passport aufgrund der zentralen Datenspeicherung auf den Servern von Microsoft und gravierender Sicherheitslücken immer mehr in die Kritik geraten war, wandten sich immer mehr Partnersites, wie z.B. das Internet-Auktionshaus eBay, von .NET Passport ab und

nahmen die Möglichkeit zur Authentifizierung über .NET Passport zurück. Sie hatten das Vertrauen in das System verloren. [KoRu00] [Kauf06] [Hick03]

3.1.1 Single Sign-In

Jeder Passport Account hat einen eindeutigen Identifizierungscode, den *Passport Unique Identifier* (PUID). Er spielt die zentrale Rolle bei der Anmeldung an Passport und Passport-unterstützenden Sites. Wenn sich ein Benutzer bei Passport einloggt, wird vom Passport-Dienst ein so genanntes Ticket ausgestellt, dessen wichtigste Bestandteile der PUID und der Zeitpunkt der letzten Authentifizierung sind. Die verschiedenen Bestandteile des Tickets speichert der Passport-Server in diese fünf Typen von Cookies, sobald sich ein Benutzer einloggt:

- Profile (MSPProf): Es enthält die zentralen Attribute des Profils.
- Secure (MSPSecAuth): Dieses Cookie wird geschrieben, wenn eine Website die SSL-verschlüsselte Sign-In-Funktion aktiviert hat. Es enthält ein nicht näher spezifiziertes Token, anhand dessen der Passport-Server erkennt, dass die SSL-Sign-In-Funktion verwendet wurde.
- Ticket (MSPAuth): Es enthält die Zeitstempel des letzten manuellen Sign-In und des letzten Refresh. Ein Refresh findet immer dann statt, wenn der Passport-Server ein neues Ticket für eine Website ausstellt. Es ist mit dem .NET Passport Schlüssel verschlüsselt.
- Ticket-Granting (MSPSec): Enthält den PUID und das Passwort des Benutzers in SSL-verschlüsselter Form. – Diese Information wird für den so genannten *silent sign-in* verwendet.
- Visited Sites (MSPVis): Enthält – unverschlüsselt – alle Site-IDs der Websites, die der Benutzer während einer Session besucht, damit sie alle beim Logout des Benutzers durch den Passport-Server benachrichtigt werden.

[Cou04] [MS03]

Damit das Ticket nicht einfach mitgelesen oder gefälscht werden kann, verschlüsselt es der Passport-Server, so dass nur er es wieder lesen kann. Für jede Passport-Partner-Site erstellt der Passport-Server nach entsprechender Aufforderung ein eigenes Ticket, welches er mit dem öffentlichen Schlüssel der Partner-Site verschlüsselt und mit einer Signatur versieht, die er mit seinem eigenen privaten Schlüssel verschlüsselt. Dadurch kann eine Site leicht feststellen, ob ein Ticket tatsächlich vom Passport-Dienst ausgestellt wurde oder nicht, und ob es tatsächlich für sie bestimmt ist. Die Partner-Site fordert das Ticket vom Passport-Server an, sobald der Benutzer zugriffsbeschränkte Inhalte von ihr anfordert, für die eine Authentifizierung erfolgen muss. Damit nicht für jede einzelne Ressource, die der Benutzer von der Site abruft, ein neues Ticket vom Passport-Dienst angefordert werden muss, kann die Site das Ticket ebenfalls in Form von Cookies im Browser des Benutzers abspeichern. Von dort kann es die Site lesen, solange es dort gespeichert ist, d.h. der Benutzer eingeloggt ist. Beim Sign-Out muss das Cookie zwingend wieder gelöscht werden. Da Cookies domänenspezifisch angelegt werden, kann die Partner-Site das

Ticket, das der Passport-Server für seine eigenen Zwecke gespeichert hat, nicht lesen und muss deshalb ein eigenes Ticket zugewiesen bekommen, das sie anschließend in eigenen Cookies speichern kann.
Der Ablauf des Single Sign-On Protokolls kann in drei Fälle unterteilt werden: Der erste Login einer Session, der Login bei einer weiteren Site während einer bereits bestehenden Session und der Single-Logout zur Beendigung der Session. [Cou04] [MS03]

Erster Login einer Session

Abbildung 3.1 stellt den Ablauf beim ersten Login einer Session dar.
1. Sobald ein Benutzer auf einem Webserver mit .NET Passport eine Webseite (oder andere Ressource) aufruft und die Sicherheitsrichtlinie des Webservers es erfordert, dass sich der Benutzer vor dem Zugriff darauf authentifiziert, überprüft der Webserver, ob im Webbrowser des Benutzers bereits ein Cookie mit einem gültigen Ticket gespeichert ist. (Wäre dies der Fall, würde der Webserver als Antwort die angeforderte Ressource übertragen.)
2./3. Da der Benutzer als *nicht authentifiziert* gilt, sendet der Webserver entweder eine Seite an den Browser, die den Browser automatisch an den Passport-Server weiterleitet und dadurch den Passport-Server zur Authentifizierung des Benutzers auffordert. Alternativ kann der Webserver auch eine Seite an den Browser senden, die einen *SignIn*-Button enthält, den der Benutzer erst anklicken muss, bevor die Authentifizierungsanfrage an den Passport-Server geleitet wird.
4./5. Der Passport-Server ist jetzt dazu aufgefordert ein Ticket auszustellen, dass ihm der Benutzer bekannt ist und sich dem Passport-Server erfolgreich authentifiziert hat. Dazu überprüft der Passport-Server zunächst, ob sich im Browser des Benutzers bereits ein (gültiges) Ticket des Passport-Dienstes befindet. Dies ist beim ersten Login nicht der Fall und daher sendet der Passport-Server dem Browser die Passport SignIn-Seite, in die der Benutzer seinen Passport-Benutzernamen und das dazu gehörige Passwort eingibt und an den Passport-Server zurücksendet.
6. Nach der erfolgreichen Überprüfung der Zugangsdaten erstellt der Passport-Server ein Ticket für seine eigene Benutzerverwaltung, da sich der Benutzer jetzt beim Passport-Service eingeloggt hat. Der Passport-Server speichert die Information des Tickets in den zwei Cookies *MSPSec* und *MSPAuth* (s.o.) im Browser des Benutzers, damit er bei weiteren Authentifizierungsanfragen darauf zurückgreifen kann (vgl. 4./5.). In einem dritten Cookie *MSPVis* speichert der Passport-Server eine Liste aller Sites, die von ihm im Laufe der Session ein Ticket erhalten haben, damit er später beim Logout den Benutzer bei allen Sites stellvertretend ausloggen kann. Deshalb fügt er die Site-ID des Webservers hinzu, dessen Anfrage er soeben bearbeitet hat. Außerdem sendet der Passport-Server eine Seite an den Browser, die eine automatische Weiterleitung zurück an den Webserver enthält, von dem die Anfrage nach Authentifizierung des Benutzers kam. Im URL des Weiterleitungsziels ist ein zweites Ticket als Parameter untergebracht. Dieses Ticket ist mit dem öffentlichen Schlüssel des Webservers verschlüsselt, damit nur der echte Adressat das Ticket mit seinem privaten Schlüssel wieder entschlüsseln kann.

7. Der Browser überträgt jetzt das Ticket an den Webserver, der das Ticket wieder entschlüsseln kann und den Benutzer daraufhin als authentifiziert akzeptiert und die ursprünglich angeforderte Ressource an den Browser überträgt. Dabei speichert der Webserver das soeben erhaltene Ticket – ebenfalls verschlüsselt – in einem Cookie im Browser des Benutzers, um beim Aufruf weiterer Ressourcen den Benutzer schneller als „bereits eingeloggt" wieder zu erkennen. [Cou04] [MS03]

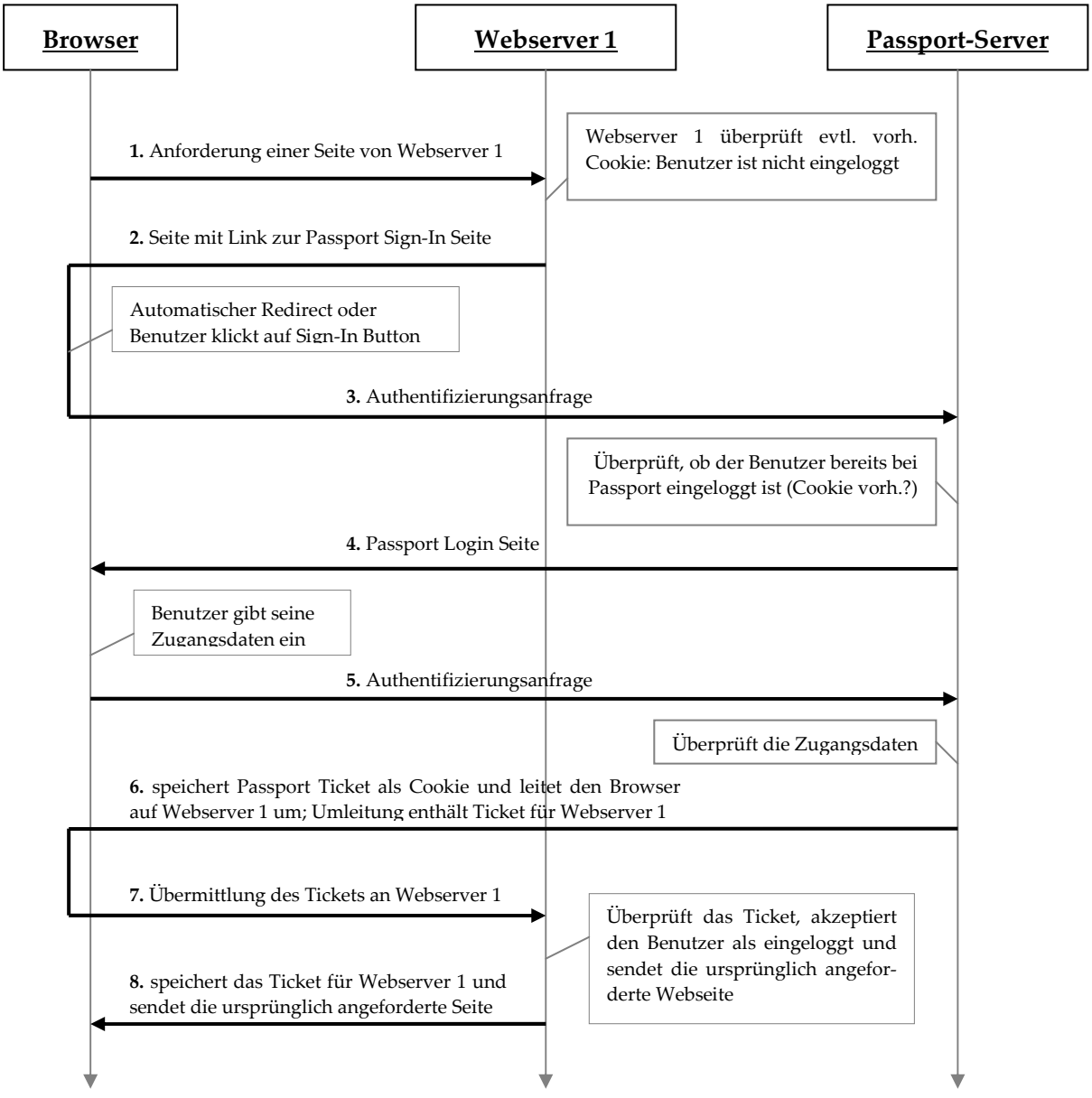

Abbildung 3.1: Erster Login einer Session

Passport-Server und Website kommunizieren in Bezug auf den Benutzer nur über den PUID, so dass man von einem gewissen Maß der Anonymität reden kann. Das kann aber nur so lange gelten, wie keine weiteren, den Benutzer identifizierenden Daten ausgetauscht werden. Der Benut-

zer kann in seinem .NET Passport Profil festlegen, welche seiner Profildaten vom Passport-Dienst an Passport-Partner-Sites übertragen werden dürfen. Dies scheint eher eine generelle Einstellung zu sein, die nicht nach Partner-Sites differenziert vorgenommen werden kann. [MS03]

Weiterer Login bei bestehender Session

In einer bestehenden Passport Session braucht der Benutzer seinen Benutzernamen und sein Passwort nicht erneut einzutippen, wenn er zugriffgeschützte Ressourcen von einem weiteren Webserver abruft. Allerdings kann eine Site jederzeit eine Re-Authentifizierung erwirken. Das kann für den Zugriff auf besonders sensible Bereiche wünschenswert sein, wie z.B. die Änderung von Stammdaten, die Bezahlung per Kreditkarte oder die Bestellung besonders hochpreisiger Waren. Eine Site kann auch festlegen, dass die letzte *manuelle Authentifizierung* – d.h. als der Benutzer zuletzt seine E-Mail-Adresse und sein Passwort eingetippt hat – nur eine bestimmte Zeitspanne zurückliegen darf, und somit ebenfalls eine Re-Authentifizierung erzwingen.
Abbildung 3.2 ähnelt sehr stark Abbildung 3.1, es fehlen lediglich die Schritte 5 und 4. Sie entfallen, weil der Passport-Server bei der Authentifizierung nicht mehr auf die Eingabe der Zugangsdaten durch den Benutzer angewiesen ist, sondern auf das gespeicherte Ticket im MSPSec-Cookie zurückgreifen kann. Der Passport-Server stellt dieses Mal ohne Interaktion mit dem Benutzer ein Ticket für *Webserver 2* aus und sendet es wiederum per Redirect via Browser an den *Webserver 2* und fügt dem MSPVis-Cookie die Site-ID von *Webserver 2* hinzu. Dieser Vorgang wird als *silent sing-in* bezeichnet.
Eine Ausnahme bildet die Anforderung einer Re-Authentifizierung. Webserver können bei der Anforderung eines Tickets von Passport-Server eine Zeitspanne angeben, die besagt, wie lange der letzte Refresh bzw. der letzte manuelle Sign-In her sein darf. Falls seit dem letzten Refresh oder manuellen Login mehr als die angegebene Zeit vergangen ist, muss der Passport-Server den Benutzer erneut zur Eingabe seiner Zugangsdaten auffordern, bevor er das angeforderte Ticket ausstellt. Diese so erzwungene Re-Authentifizierung kann beispielsweise dann sinnvoll sein, wenn der Benutzer bei einem Online-Händler Waren bestellen will. Durch diese Funktion kann sichergestellt werden, dass in diesem entscheidenden Moment auch tatsächlich der echte Benutzer am Computer sitzt und nicht vor kurzem den Rechner verlassen hat und nun ein Anderer die Bestellung ausführen will.
[Cou04][MS03]

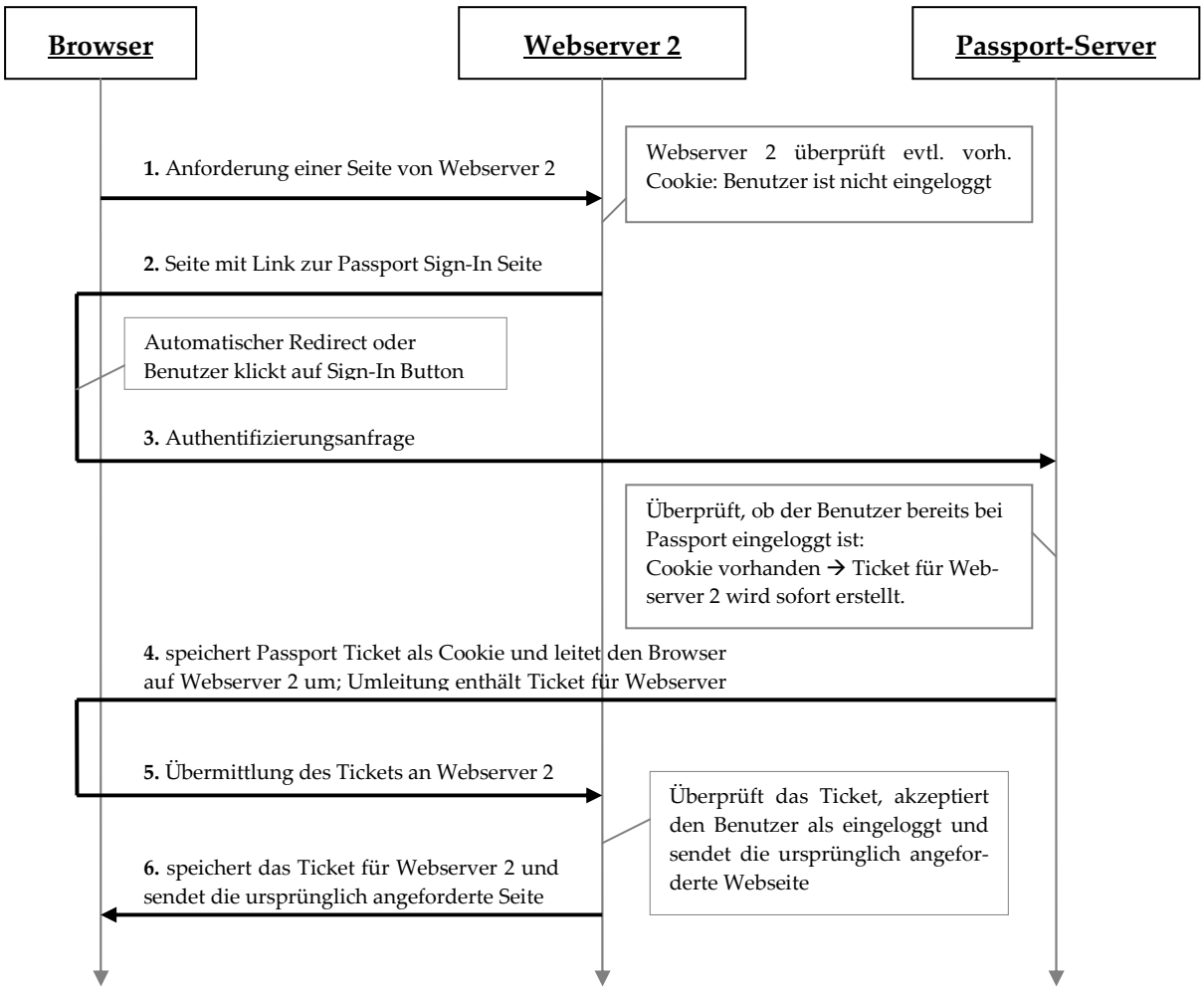

Abbildung 3.2: Weiterer Login bei bestehender Passport-Session (Single Sign-in)

[Cou04]

3.1.2 Single Sign-Out

Sobald der Benutzer während seiner Session auf irgendeiner der Seiten, auf denen er in dieser Session eingeloggt ist, auf den Sign-out Link klickt, wird er auf eine zentrale Sign-out-Seite auf dem Passport-Server umgeleitet (vgl. Abbildung 3.3). Diese Seite liest die Liste der Site IDs aus dem MSPVis-Cookie aus und löscht alle Cookies im Browser des Benutzers, die für die Domäne *.passport.com angelegt wurden, dadurch können keine weiteren *silent sign-ins* mehr durchgeführt werden. Dann erzeugt der Passport-Server eine HTML-Seite und verarbeitet dabei die Liste der Site IDs. Für jede Site wird eine Zeile mit dem Namen der Site und ein Bild eingefügt, das von der jeweiligen Site eingebunden wird. Der URL eines jeden Bildes verweist dabei nicht direkt auf eine Bilddatei auf dem Webserver der Site, sondern ruft stattdessen das Logout-Script der Site auf, welches deren .NET Passport-bezogenen Cookies aus dem Browser des Benutzers löscht. Erst nach der erfolgreichen Löschung sendet das Script als Ergebniswert die Bilddatei, die eine „check box“ zeigt. Der Benutzer sieht in seinem Browser als Ergebnis also eine Liste der besuch-

ten Sites und daneben erscheint dann jeweils die „check box“, sobald die Site das Löschen ihrer .NET Passport-bezogenen Cookies erfolgreich beendet hat. Den URL zu seinem cookie-delete Script muss jeder Site-Betreiber beim .NET Passport-Dienst hinterlegen. Er wird zusammen mit der Site-ID und allen anderen Daten zu der Site in der .NET Passport-Datenbank gespeichert. [MS03]

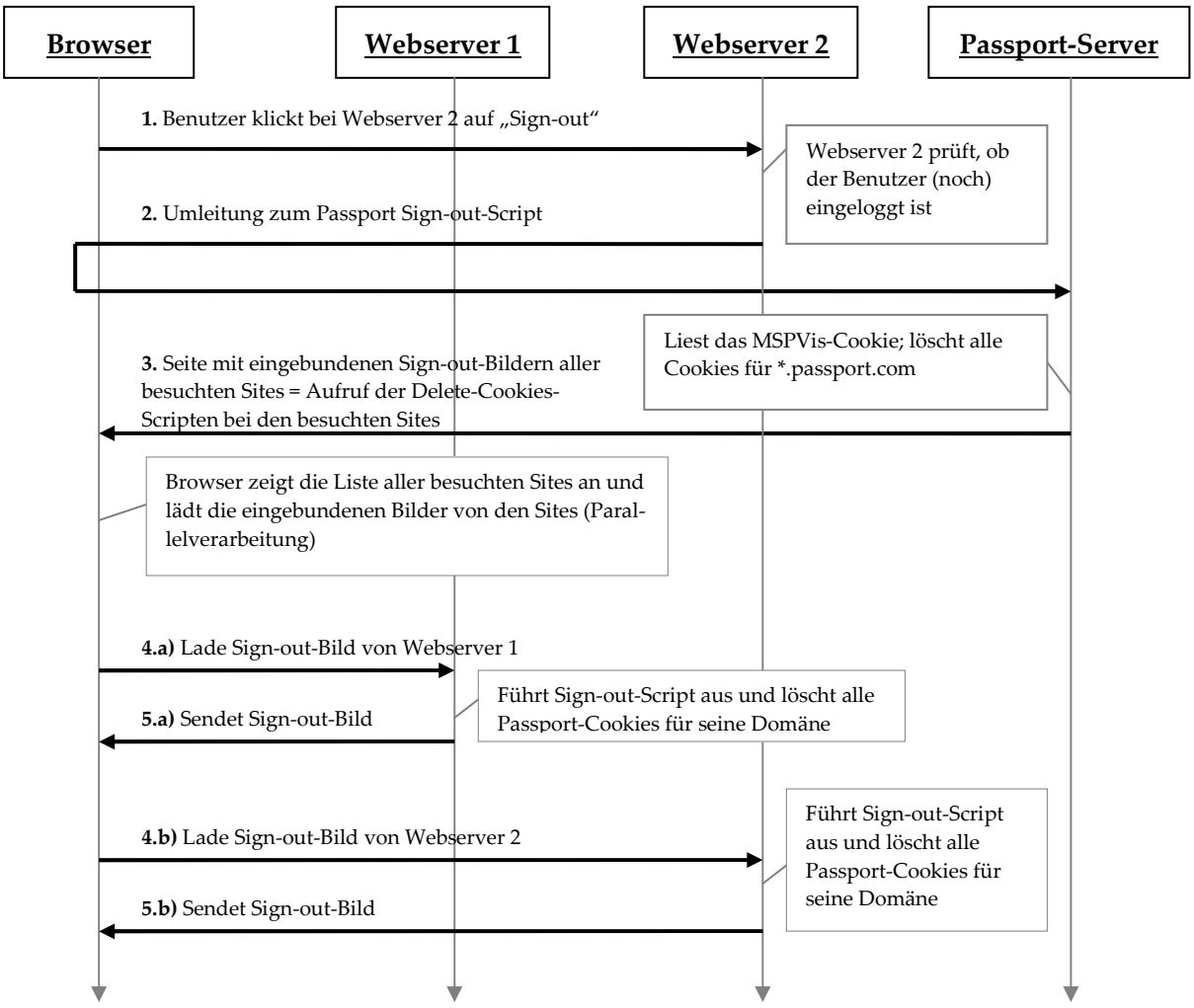

Abbildung 3.3: Sign-out Ablauf

3.2 Windows Live ID

Der Nachfolger von .NET Passport heißt *Windows Live ID* und ersetzt seit April 2006 den .NET Passport Dienst. Windows Live ID ist zu .NET Passport kompatibel, wird in Zukunft aber auch die neuere InfoCard Technologie (vgl. 3.3) unterstützen. Die Namensänderung – und der damit verbundene Abschied vom Namensbestandteil *.NET* – hat auch mit Microsofts neuer Ausrichtung auf seine Internetstrategie *Windows Live* zu tun. [Chow06][Hugh06][Kauf06]

„Unter der Haube“ hat sich aber zunächst nichts verändert. Der Protokollfluss bei der Authentifizierung hat sich gegenüber Passport nicht verändert. Es sind weiterhin die gleichen acht Schritte, die in Abbildung 3.1 dargestellt sind. [MS07]

Mittlerweile hat Microsoft seinen Authentifizierungsdienst auch für die Verwendung mit Software von Drittanbietern geöffnet. Wer Live ID für seine Website nutzen will, ist damit nicht mehr auf den Einsatz des *Internet Information Server* als Webserver-Software festgelegt, wie es zuvor Voraussetzung war. Stattdessen gibt es jetzt die Möglichkeit, außer AST.NET auch freie Scriptsprachen wie PHP oder Ruby zu verwenden. [Wolf07][MS03]

Der nächste Entwicklungsschritt ist die bereits genannte Unterstützung von InfoCard und des propagierten Identity Metasystems. Beide sind aus grundsätzlichen Überlegungen heraus entstanden, wie ein zukünftiges umfassendes Identity-System aussehen müsste, damit es eine breite Akzeptanz findet und nicht an denselben Punkten scheitert wie die Passport-Technologie.

3.3 InfoCard

Einen ganz anderen Ansatz als bei .NET Passport verfolgt Microsoft bei InfoCard. Die Basis ist ein Papier mit dem Titel *The Laws of Identity* (vgl. 3.4.1) von Microsoft-Mitarbeiter Kim Cameron und die Idee, dass es im Internet eine Art Metasystem für die Handhabung von Identitäten geben sollte, da alle bisherigen Systeme mindestens eine der sieben Gesetzmäßigkeiten aus [Cam05] verletzten. Der Grundgedanke, dass Personen sich in verschiedenen Kontexten bewegen und dabei jeweils unterschiedliche Identitäten haben und diese in ihrer Vielfalt in die Online-Welt abgebildet werden sollen, ist die Abkehr von dem zentralistischen Ansatz, der hinter .NET Passport stand. Das InfoCard-System funktioniert ohne Benutzernamen und Passwörter. Stattdessen ist es von bestimmter Software abhängig, die auf dem Client-Computer installiert sein muss. Das Modell hinter der Verwendung von InfoCard ist aus dem realen Leben gegriffen, wo sich Personen in unterschiedlichen Situationen mit verschiedenen Dokumenten ausweisen, die sie situationsabhängig aus der Menge ihrer mitgeführten Dokumente wählen. Bei der Benutzung öffentlicher Verkehrsmittel ist das z.B. die Monats- oder Jahreskarte mit einem Lichtbild. Ein solches Dokument wird vom Verkehrsbetrieb für den Fahrgast ausgestellt, der es bei Bedarf einem Fahrkartenkontrolleur präsentiert. Der Kontrolleur interessiert sich nur für die Gültigkeit der Fahrkarte für die gerade durchgeführte Fahrt und ob das Lichtbild darauf die Person darstellt, die die Fahrkarte benutzt. Bei InfoCard gibt es drei Rollen, nämlich *Identity Provider* (IDP), *Client* und *Relying Party* (RP). [WHN06][WNB06][Wat07]

3.3.1 Karten

Der Benutzer legt Karten an, die er entweder selbst erstellt, oder bei einem IDP erstellen lässt. Im zweiten Fall fordert der Identity Provider den Benutzer zum Download der Karte (.CRD-Datei)

auf oder sendet sie dem Benutzer an die registrierte E-Mail-Adresse. Nach dem Erhalt importiert der Benutzer die Karte in ein lokal installiertes Kartenverwaltungsprogramm. Die Karten enthalten verschiedene Identitätsdaten, also Merkmale über den Benutzer. Selbst erstellte Karten entsprechen bei einem herkömmlichen System der freien Wahl von Benutzername und Passwort während der Registrierung. Im Prinzip kann der Benutzer bei einer Registrierung beliebige Daten eintragen, die er sich gerade ausgedacht hat. Dadurch kann er sich z.B. unter einem Pseudonym anmelden anstatt mit seinem echten Namen. Diese Möglichkeit bleibt durch selbst erstellte Karten erhalten, weil die Daten durch keine andere Instanz als den Benutzer selbst kontrolliert werden können. Sie entsprechen in gewisser Weise auch privaten Visitenkarten. Geschäftliche Visitenkarten unterscheiden sich von privaten dadurch, dass die Firma mit ihrem Namen für die Richtigkeit der abgedruckten Daten steht. Außerdem enthält sie die implizite Aussage: „Diese Person arbeitet bei uns." Bei InfoCard werden die Karten als .CRD-Datei auf dem Computer des Benutzers gespeichert. Sie enthalten zur Sicherheit nur Metadaten und nicht die echten Identitätsdaten. Eine Karte hat einen Namen, ein Bild oder Logo und die Kontaktdaten für den *Security Token Service* (STS), von dem die eigentlichen Identitätsdaten bezogen werden. Die Kontaktinformationen zum STS bestehen aus einer Liste, in der für jede Authentifizierungsmöglichkeit, für die der STS ein Token erstellen kann, ein URL des jeweiligen Dienstes beim STS. Gegenüber dem STS muss sich der Benutzer auch authentifizieren. Dafür kann u. U. auch eine selbst ausgestellte Karte genommen werden. Das Security Token, das der STS bereitstellt, wird zur Authentifizierung gegenüber einer Relying Party (z.B. Website, Web Service) verwendet.
Die Karte enthält darüber hinaus noch zwei Listen: Die erste Liste führt auf, welche Arten von Identitätsdaten der STS bereitstellen kann. Bei der selbstausgestellten Karte sind die möglichen Attribute allgemeingültig festgelegt; z.B. Name, Vorname, Adresse, Postleitzahl, Ort etc. Alle Daten, die auf einer selbstausgestellten Karte gespeichert werden können, sind Daten die auch sonst öffentlich zugänglich sein könnten, wie z.B. aus dem Telefonbuch. Die zweite Liste auf der Karte enthält alle Token-Typen, die der STS ausstellen kann. Das ist wichtig für den späteren Abgleich mit den Anforderungen der Relying Party, die für eine Authentifizierung Vorgaben dazu machen kann, mit welcher Art von Token die Authentifizierung erfolgen soll. Die gesamte Karte wird vom Identity Provider signiert und die Signatur enthält auch ein Sicherheitszertifikat für den IDP. Bevor die Karte im System des Benutzers installiert wird, kann er sie sich genau ansehen und auch das Sicherheitszertifikat der Zertifizierungsstelle überprüfen um sicherzustellen, dass die Karte tatsächlich von diesem IDP ausgestellt wurde. Nach dem Import in das InfoCard-Verwaltungstool, das auf dem Computer installiert sein muss, steht die Karte zur Verfügung. Für die Windowswelt stellt Microsoft eine entsprechende Software namens *CardSpace* zur Verfügung. In Windows Vista ist sie bereits integriert und für Windows XP und Windows Server 2003 ebenfalls erhältlich. [WHN06][WNB06][Wat07]

3.3.2 Relying Party

Die Relying Party legt eine Richtlinie (Policy) fest, welche Teilmenge der Identitätsdaten eines Benutzers sie benötigt, welchen Identity Providern sie vertraut und in welcher Form sie das Security Token haben möchte (X.509, SAML 1.1, Kerberos), bevor sie ihm Zugriff auf eine geschützte Resource gewährt.
Die InfoCard-Software hat die Funktion eines *Identity Selectors*, arbeitet als Browser Plug-In und tritt an diesem Punkt für den Benutzer in Erscheinung. Sie nimmt die Anforderungen der Relying Party entgegen und gleicht sie mit den gespeicherten Karten ab. Karten, die den Anforderungen genügen, werden hervorgehoben angezeigt, die anderen Karten werden zwar angezeigt, weil sie ja vorhanden sind, werden aber ausgegraut dargestellt und sind nicht auswählbar, weil sie die Anforderungen nicht erfüllen können. Der Benutzer kann dann wählen, welche der geeigneten Karten er der Relying Party präsentieren möchte. Ein Beispiel aus der realen Welt (im Ggs. zur virtuellen Welt des Internet) wäre der Altersnachweis mithilfe eines mitgeführten Dokuments. Dafür kommen u. a. Personalausweis oder Führerschein in Frage, da es sich um Dokumente mit Lichtbild handelt, auf denen das Geburtsdatum der abgebildeten Person vermerkt ist. Die Kreditkarte ist aber nicht geeignet, da erstens das Geburtsdatum nicht enthält (zumindest in menschenlesbarer Form) und häufig auch kein Bild des Inhabers. Die Person, die nun einer anderen Person gegenüber einen Altersnachweis erbringen soll, kann sich überlegen, ob sie den Personalausweis verwendet und damit preisgibt, in welchem Land sie Bürger ist, oder ob sie den Führerschein verwendet und damit preisgibt, dass sie im Besitzt einer Fahrerlaubnis ist und in welcher Gemeinde dieses Dokument ausgestellt worden ist.
Erst nachdem der Benutzer in InfoCard die Karte ausgewählt hat, die er der Relying Party präsentieren möchte, fordert InfoCard vom *Security Token Service* (STS), der in der Karte für die angeforderte Art von Token vermerkt ist, ein entsprechendes Token an; also z.B. ein SAML 1.1-Token. Der STS muss nicht zwangsläufig mit dem IDP identisch sein, es besteht aber eine gegenseitige Vertrauensbeziehung zwischen den beiden. Ein wichtiger Punkt ist, dass der STS nicht erfährt, für welche Relying Party das Token verwendet werden soll, ebenso wenig wie das Straßenverkehrsamt erfahren würde, dass eine Person den Führerschein als Altersnachweis in einer Videothek vorgelegt hat.
Bei der Anforderung des Security Tokens muss sich der Benutzer gegenüber dem STS authentifizieren, was wieder mit einer Karte geschieht. Eine Relying Party kann schließlich in ihrer Policy festlegen, ob sie selbst erstellte Karten für die Authentifizierung akzeptiert oder nicht.

InfoCard nimmt das Token entgegen, präsentiert es dem Benutzer und markiert dabei die Bestanteile (Daten), die die Relying Party angefordert hat, als notwendig. Falls das Token darüber hinaus noch weitere Identitätsdaten über den Benutzer enthält, kann er entscheiden, ob er diese Daten als optionale Daten an die Relying Party mit übertragen haben möchte oder nicht.

Wichtig ist, dass die Kommunikation zwischen den beteiligten Systemen immer verschlüsselt erfolgt. Zum Einsatz kommt das SSL-Verfahren mit asymmetrischen Schlüsselpaaren aus öffentlichen und privaten Schlüsseln. [WHN06][WNB06]

3.4 Identity Metasystem

Windows CardSpace (Entwicklungsname: InfoCard) ist Microsofts Implementierung für das propagierte Identity Metasystem. Das Identity Metasystem soll der Internetkommunikation eine eigene Schicht für Authentifizierung und Identitätsmanagement hinzufügen, welches den *Laws of Identity* gehorcht.

3.4.1 The Laws of Identity

Der Begriff *law* (engl.: Gesetz) sollte eher als *Gesetzmäßigkeit* wie bei „Naturgesetz“ angesehen werden. Der Microsoft-Mitarbeiter Kim Cameron hat versucht zu analysieren, warum .NET Passport nicht die erhoffte Akzeptanz gefunden hatte, und welche Parameter künftige Systeme bzw. eine Art Metasystem erfüllen müssten.
Blog . Öffentlichkeit – Diskussion – Resultat – hier zusammengefasst.

1. **Kontrolle durch den Benutzer und Zustimmung zur Informationsweitergabe**
   Der Benutzer muss es in der Hand haben, welche Informationen über sich selbst an wen weitergegeben werden.
2. **Minimale Datenweitergabe für einen bestimmten Verwendungszweck**
   Wenn jemand eine Wettervorhersage für seinen Wohnort abruft, braucht der Dienst nicht dessen Namen zu erfahren um seine Dienstleistung zu erbringen, sondern nur dessen Wohnort. Oder wenn jemand bestimmte Waren kaufen möchte, wie z.B. Alkohol, reicht die (bestätigte) Aussage, dass diese Person mindestens das erforderliche Alter hat, aber das genaue Geburtsdatum ist dabei nicht von Bedeutung.
3. **Berechtigte Parteien**
   Nur die Parteien, die direkt miteinander interagieren, sind die einzigen am Austausch der Identitätsdaten berechtigten Parteien. In diesem Punkt ist das .NET Passport-System gescheitert, weil nicht zu vermitteln war, wieso Microsoft am Datenaustausch beteiligt sein sollte, wenn ein Benutzer Waren in einem Online-Versandhaus kauft.
4. **Gerichtete Identität**
   Wenn eine Person in einer Stadt Schuhe kaufen will, sieht sie sich nach dem Ladenschild eines Schuhladens um. Das Schild hängt ständig da und bekundet, dass hier ein Schuhgeschäft ist. Die Person aber hält ihre Identität bedeckt und trägt gewöhnlich kein öffentlich sichtbares Namensschild mit sich herum. Wenn sie dann im Schuhgeschäft einkauft und die Ware per Kreditkarte bezahlt, teilt sie dem Schuhgeschäft einen (relevanten) Teil ihrer Identität mit, aber nicht allen anderen Kunden, die sich dort ebenfalls aufhalten.

5. **Pluralismus der Betreiber und der Technologien**
   Ein universelles Identity-System muss in der Lage sein, unterschiedliche Identitäts-Technologien verschiedener Identity Provider miteinander zu verbinden, die zudem auf unterschiedlichen Rechnersystemen laufen können. Einen einzelnen Betreiber könnte es für ein solches universelles System gar nicht geben, weil damit Punkt 3 nicht erfüllt wäre. Wenn dadurch nämlich alle Informationsfäden bei einem einzelnen Betreiber zusammen liefen, würde das Misstrauen bei den Personen hervorrufen, die das System benutzen sollen, und zur Ablehnung des Systems führen.
6. **Einbeziehung des Menschen**
   Der menschliche Benutzer sollte immer Handlungssicherheit bei der Bedienung eines Systems haben. In Bezug auf die Handhabung von Identitätsdaten bedeutet Handlungssicherheit gleichzeitig einen besseren Schutz vor dem Diebstahl der Daten durch Täuschung, z.B. Phishing-Attacken.
7. **Gleiche Systembedienung, unabhängig vom Benutzungskontext**
   Auch wenn ein Benutzer in unterschiedlichen Kontexten agiert, z.B. einmal als Angestellter einer Firma und ein anderes Mal als Privatperson, soll die Bedienung des Identitäts-Systems auf die gleiche Art und Weise geschehen, um die Handlungssicherheit aufrecht zu erhalten.

[Cam05][Wat07]

4 Das Liberty Alliance Project

Im September 2001 schlossen sich knapp 20 Firmen aus der ganzen Welt zur Liberty Alliance zusammen, um einen offenen Standard zur Verwaltung verknüpfter Identitäten (federated identities) zu erstellen. Unter der Verknüpfung von Identitäten versteht man die Herstellung einer Beziehung zwischen zwei sonst als unabhängig zu sehenden Identitäten, die ein Benutzer bei verschiedenen Anbietern haben kann.

Unter den Mitgliedern der Liberty Alliance finden sich heute Dienstleistungsfirmen, Technologie-Unternehmen, Regierungs- und andere politische Organisationen. Um einige Beispiel zu nennen: Sun Microsystems, NEC, T-Com, RSA Security, die Fraunhofer-Gesellschaft, das Dänische Ministerium für Wissenschaft, Technologie und Innovation. Insgesamt beläuft sich die Zahl der Mitglieder laut [LAP07a] heute auf knapp 150, während es zwischenzeitlich – Anfang 2003 – schon einmal über 160 Firmen waren [LAP03].

Der Vorstand besteht aus 10 Sponsorenmitgliedern, die die Allianz (mit?)gegründet haben. Er zeichnet für die Gesamtleitung, die Finanzen, rechtliche Belange und??? (engl. operations) verantwortlich und ist die entscheidende Instanz bei der Verabschiedung der Spezifikationen, die in drei Expertengruppen innerhalb der Liberty Alliance, erarbeitet werden. Die drei Expertengruppen existieren für die Bereiche „Business Marketing“, „Technologie“ und „Öffentliche Strategie“ (Public Policy). Letztere berät den Vorstand in Fragen des Datenschutzes, der Sicherheit und der allgemeinen öffentlichen Darstellung, steht in Kontakt mit Datenschutzgruppen und Regierungsbehörden und entwickelt außerdem Richtlinien und Verfahren zum Datenschutz, die später veröffentlicht werden sollen. [LAP06] [LAP07a]

Die Gruppe der Technologieexperten entwickelt technische Architekturen und Spezifikationen und definiert die Programme zur Interoperabilität und Übereinstimmung, damit später die Produkte, die nach dem Liberty-Entwurf implementiert werden, auch tatsächlich untereinander kommunizieren können.

In der Gruppe der Business Marketing-Experten schließlich werden die Anforderungen und die „use cases“ (Anwendungsfälle) entwickelt. Sie ist auch für die Verbreitung (engl. evangelism) (Verbreitung wovon?) und die Öffentlichkeitsarbeit zuständig, entwickelt Geschäftsvorlagen und -richtlinien und beschleunigt die Entwicklung von Märkten.

Zu guter Letzt tragen alle Mitglieder der Liberty Alliance durch Rückmeldungen, Verbesserungsvorschläge und Anregungen zu Entwürfen der Expertengruppen bei. [LAP06]

Die Begriffe *Liberty Alliance* und *Liberty Alliance Project* sind insofern als synonym anzusehen, als es derzeit keine anderen Ziele der Liberty Alliance gibt als das Liberty Alliance Project. Eine ebenfalls gebräuchliche Bezeichnung, die dasselbe meint, ist *Project Liberty*. So ist zum Beispiel die Homepage des Liberty Alliance Project unter dem URL http://www.project-liberty.org zu erreichen.

4.1 Ziele

Das Liberty Alliance Project (kurz: LAP) ist ein Standard schaffendes Projekt. Es ist im Gegensatz zu .NET Passport kein konkretes Produkt, sondern die Entwicklung einer Art Blaupause oder Rahmenvorgabe, nach der sich konkrete Endprodukte später richten sollen. Dadurch soll gleichzeitig die Interoperabilität zwischen verschiedenen solcher Endprodukte sichergestellt werden.
Die Ziele der Liberty Alliance sind im Einzelnen:

1. Bereitstellung eines offenen Standards und Geschäftsrichtlinien für die Verwaltung verknüpfter Identitäten für alle Arten von Netzwerkgeräten.
2. Bereitstellung eines offenen und sicheren Standards für Single Sign-On (SSO) mit dezentralisierter Authentifizierung und offener Autorisierung.
3. Firmen und Verbrauchern soll es ermöglicht werden, persönliche Information sicher und nach eigenen Vorstellungen zu handhaben.

[LAP06]

4.2 Das Konzept

Das Konzept sieht vor, dass bereits bestehende lokale Identitäten eines Benutzers erhalten bleiben, und die Single Sign-On-Funktionalität durch *Verknüpfung dieser Identitäten* hinzugefügt wird. Es gibt keine einzelne, übergreifende, globale Benutzerdatenbasis, auf die alle Dienstanbieter im Rahmen des Vertrauenskreises zurückgreifen, um ihre Benutzer (Kunden) zu identifizieren/authentifizieren, sondern jeder behält seine eigene Benutzerverwaltung. Die Möglichkeit zur lokalen Anmeldung beim einzelnen Dienstanbieter bleibt damit erhalten. Es kommt also für das Single Sign-On eine neu Art Anmeldevorgang hinzu, für den aber keine neue, gesonderte Identität benötigt wird. Dadurch bleiben dem Benutzer bereits erworbene Vorteile, wie etwa besondere Konditionen als treuer Kunde, Bewertungsprofile etc. erhalten; Vorteile, die er möglicherweise verlieren würde, wenn er gezwungen wäre, eine neue Identität für das Single Sign-On aufzubauen. Bevor Identitäten miteinander verknüpft werden können, müssen die Provider untereinander geschäftliche Beziehungen eingehen und vertragliche Vereinbarungen zur Zusammenarbeit treffen, in denen geregelt wird, wer Service Provider (SP) und wer Identity Provider (IDP) ist oder wer sogar beide Rollen einnimmt, und welche Formen der Authentifizierung von Benutzern gegenseitig akzeptiert wird. Gruppen von Providern, die untereinander solche Vereinbarungen getroffen haben, werden *Circles of Trust (Vertrauenskreise)* genannt. In einer solchen Gruppe gibt es demnach Service Provider (SP), die einem Benutzer bestimmte Dienste anbieten, und Identity Provider (IDP), die im Auftrag der Service Provider die Identitäten der Benutzer verlässlich authentifizieren. Ohne Vertrauensbeziehungen ist eine Zusammenlegung der Accounts und damit Single Sign-On im Sinne des Liberty Alliance Project nicht möglich. [HoWa03]

4.3 Die Architektur

Die Architektur des Liberty Alliance Project besteht im Wesentlichen aus drei Modulen: Dem *Identity Federation Framework (ID-FF)*, dem *Identity Web Services Framework (ID-WSF)* und den *Iden-*

tity Services Interfaces Specifications (ID-SIS). Dazu kommt noch eine Ansammlung von Basistechnologien, die als ein weiteres Modul angesehen wird. [LAP03]

4.3.1 Identity Federation Framework (ID-FF)

Im Mittelpunkt der Architektur des LAP steht das *Identity Federation Framework* (ID-FF), welches die Funktionen zur Verfügung stellt, die zur Verknüpfung und Verwaltung von Identitäten durch den Benutzer (Opt-in) benötigt werden, sowie Funktionen für *Simplified Sign-On* und ein grundlegendes Session Management inklusive einer *Global Sign-Out*-Funktion. Dadurch bildet es die Kernideen des LAP-Konzepts ab (vgl. 4.2). Das ID-FF ist so konzipiert, dass es sowohl als eigenständige Lösung arbeiten kann oder aber auch in Verbindung mit anderen bestehenden Identity Management Systemen, und es ist so ausgelegt, dass es mit allen Arten von Netzwerkgeräten und Softwareplattformen funktionieren kann. [LAP03]

Die Entwicklung des ID-FF wurde auf der Basis von SAML 1.x begonnen, welches um zusätzlich benötigte Funktionen ergänzt wurde, die in SAML 1.x noch nicht zur Verfügung standen. Inzwischen hat die Liberty Alliance die Version 1.2 des ID-FF als Anregung für die Entwicklung von SAML 2.0 beigesteuert, da somit leichter ein einziger Standard für Single Sign-On mit verknüpften Identitäten zu erreichen ist. (vgl. 2.6.) [LAP03] [Wis+05]

Verknüpfung von Identitäten (Account Linking)

Wenn ein Benutzer bei mehr als einem Partner desselben Vertrauenskreises registriert ist, hat er die Möglichkeit, diese unterschiedlichen Identitäten miteinander zu verknüpfen. Diese richtet sich danach, welche Partner untereinander Verträge über eine Zusammenarbeit im Sinne des LAP abgeschlossen haben. Nach der lokalen Anmeldung beim SP präsentiert dieser dem Benutzer die Möglichkeiten zur Verknüpfung seines Accounts mit denen auf Partnersites. Um die Verknüpfung durchführen zu können, ist es notwendig, sich (nacheinander) bei beiden Providern mit den jeweiligen Zugangsdaten anzumelden. [HoWa03]
Das LAP benutzt sowohl für den Single Sign-On Vorgang, als auch die Account-Verknüpfung dasselbe Protokoll, wobei die Account-Verknüpfung als Option ausgewählt werden kann.
Der Ablauf beginnt immer beim SP. Er sendet eine Authentifizierungsanfrage `<AuthnRequest>` an den Identity Provider und fordert eine Echtheitsbestätigung über den Benutzer an *(authentication assertion)*. Im gleichen Zug kann der SP den IDP auch noch zur Verknüpfung der Identität des Benutzers beim IDP mit dessen Identität beim SP auffordern. Dazu setzt der SP in seiner Authentifizierungsanfrage das Attribut <lib:Federate> auf *true* (siehe Abbildung 4.1).
Nachdem der IDP die Zugangsdaten vom Benutzer abgefragt und überprüft hat, schickt er dem SP eine Antwort `<AuthnResponse>`, die entweder die gewünschte Zusicherung selbst enthält oder ein Artefakt, mit dessen Hilfe die gewünschte Zusicherung in einem zweiten Schritt dereferenziert werden kann, d.h. beim IDP abgeholt werden kann (vgl. 2.7.3). Darüber hinaus verknüpft der IDP die Identitäten des Benutzers, sofern dies angefordert war. [BeKe03]

```
<lib:AuthnRequest id="12345" RequestID="RPCUk211+GVz+t1lLURp51oFvJXk" Ma-
jorVersion0"1" MinorVersion="0" IssueInstant="2001-12-17T21:42:4Z">
      <ds:Signature> … </ds:Signature>
      <lib:ProviderID>http://ServiceProbider.com<lib:ProviderID>
      <lib:ForceAuthn>false</lib:ForceAuthn>
      <lib:IsPassive>false</lib:IsPassive>
      <lib:Federate>true<lib:Federate>
      <lib:ProtocolProfile>http://projectliberty.org/profiles/brws-
post</lib:ProtocolProfile>
      <lib:AuthnContext>
      <libAuthnContextClass-
Ref>http://projectliberty.org/schemas/authctx/classes/Password-
ProtectedTransport</lib:AuthnContextClassRef>
      </lib:AuthnContext>
      <lib:RelayState>R01GODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</lib:RelayState>
      <lib:AuthnContextComparison>exact</lib:AuthnContextComparison>
</lib:AuthnRequest>
```

Abbildung 4.1: Authentifizierungsanfrage mit Verknüpfungsanforderung

Die Verknüpfung der Identitäten erfolgt dabei durch den Austausch von so genannten *user handles*, das sind Zeichenketten mit quasi-zufälligem Charakter. Der IDP erstellt einen solchen user handle und teilt ihn dem SP mit. Der SP kann nun den user handle in der Kommunikation mit dem IDP verwenden, wenn er sich auf den betreffenden Benutzer bezieht. Die Provider erlangen keine Kenntnis darüber welche Benutzernamen miteinander verknüpft werden. Sie kennen und verwenden ausschließlich den user handle, der innerhalb der Domäne des IDP eindeutig sein muss. Optional kann der SP noch seinen eigenen user handle generieren und ihn dem IDP mitteilen. Der IDP verwendet dann bei der Kommunikation mit dem SP den user handle, welchen der SP generiert hat, und der SP verwendet den vom IDP generierten user handle, wenn er mit dem IDP kommuniziert. Die Provider legen alle user handles in ihrem jeweiligen Benutzerverzeichnis ab. Abbildung 4.2 zeigt beispielhaft, wie die Einträge der Benutzerverzeichnisse nach der Verknüpfung der Identität *Jack456* beim IDP mit der Identität *JackMiller* beim SP aussehen. Die user handles stehen bei jeweils beiden Providern im Benutzerverzeichnis eingetragen; auf der einen Seite unter alias, auf der anderen Seite unter Name. Dagegen sind die eigentlichen Benutzernamen, die der Benutzer kennt und zum Einloggen verwendet, jeweils nur dem IDP bzw. dem SP bekannt. Der beidseitige Austausch von user handles ist nicht verbindlich, sondern stellt lediglich eine Option des *Liberty Single Sign-On and Federation Protocol* dar. Verbindlich ist dagegen nur die Erstellung eines user handles durch den IDP und die Übermittlung an den SP. Dieses Konzept ermöglicht Szenarios, in denen ein SP kein eigenständiges Benutzerverzeichnis führt, sondern sich ausschließlich auf das Benutzerverzeichnis des IDP stützt. [HoWa03]

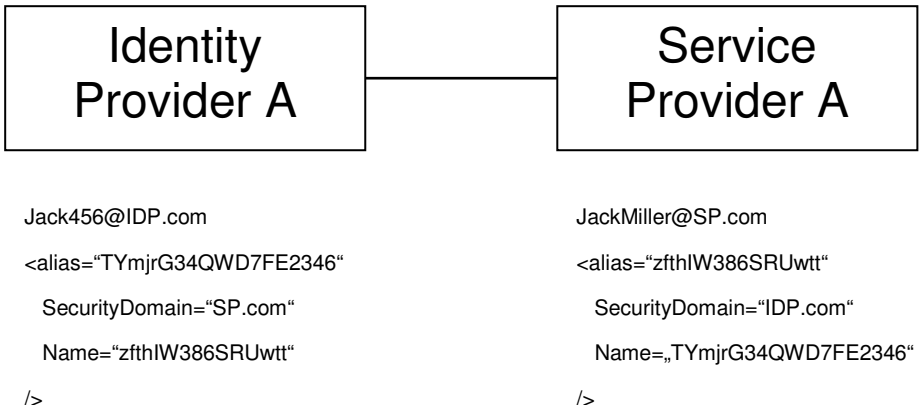

Abbildung 4.2: Benutzerverzeichnisse nach einer Identitätsverknüpfung

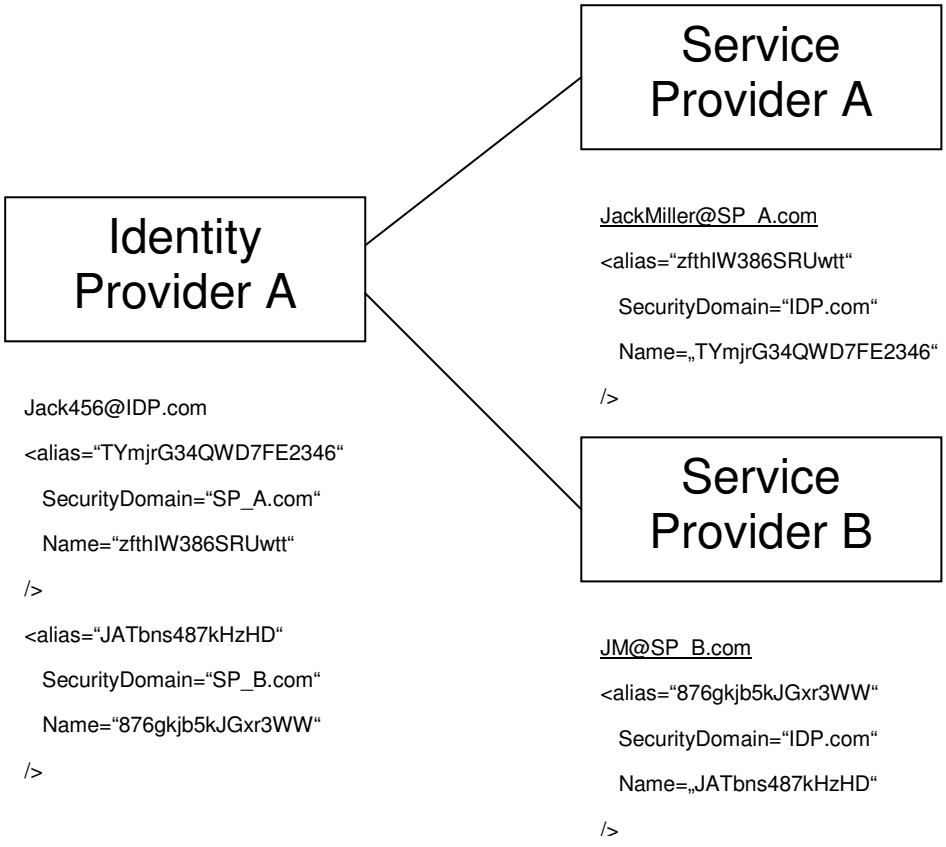

Abbildung 4.3: Verknüpfung von zwei SP-Identitäten mit einer IDP-Identität

Abbildung 4.3 zeigt die Benutzerverzeichnisse nach der Verknüpfung der Identitäten bei zwei SP mit der Identität bei einem IDP. Es sei darauf hingewiesen, dass die SP jeweils nur mit dem IDP bezüglich des Benutzers *Jack Miller* kommunizieren können, aber nicht untereinander, da die Identitäten nicht miteinander verknüpft sind.

Es gibt ebenfalls das Szenario, in dem der Benutzer eine Identität bei einem SP mit seinen Identitäten bei mehreren Identity Providern verknüpft. Dahinter steckt die Überlegung, dass der Be-

nutzer aus unterschiedlichen Kontexten heraus den Dienst des SP in Anspruch nehmen wollen könnte. Ein möglicher Kontext wäre, wenn sich der Benutzer an seinem Arbeitsplatz befindet und dort angemeldet (eingeloggt) ist, wofür er ggf. schon einen IDP verwendet hat. Mit dem Ziel des Single Sign-On vor Augen wäre es für den Benutzer günstig, wenn er sich bei den SP, die er im Laufe seiner Arbeitssitzung (Session) verwenden möchte, über denselben IDP authentifizieren zu können.
In einem weiteren Kontext könnte sich der Benutzer auf einer Geschäftsreise befinden oder an einem anderen Standort seiner Firma, wo er sich nicht über den IDP authentifizieren kann, den er an seinem Hauptarbeitsplatz verwendet. Wenn er sich stattdessen über einen andern Liberty-fähigen IDP authentifizieren kann, wäre es für den Benutzer sinnvoll, seine Identitäten bei den SP, die er benutzt, mit der Identität bei dem zweiten IDP zu verknüpfen. Dadurch könnte er Single Sign-On auch innerhalb dieses Kontexts verwenden. Abbildung 4.4 zeigt die Verknüpfung der Identität bei einem SP mit den Identitäten bei zwei IDP. [HoWa03]

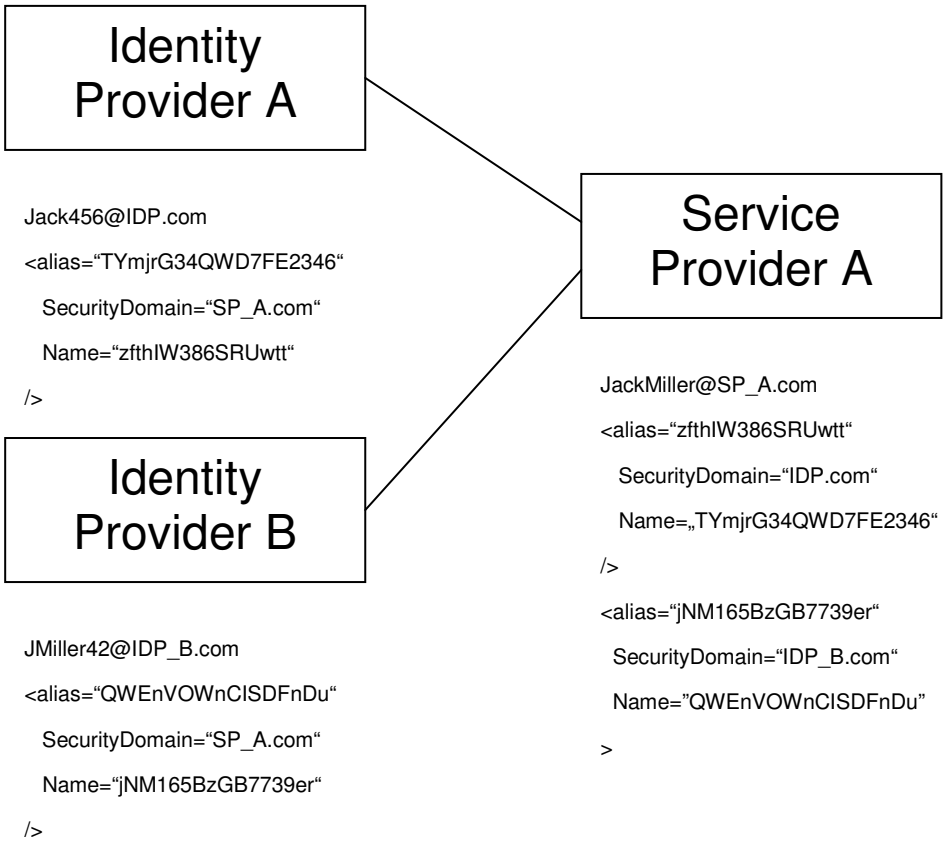

Abbildung 4.4: Verknüpfung einer SP-Identität mit zwei IDP-Identitäten

Ein globaler Namensraum für eindeutige Benutzernamen, die bei allen beteiligten Providern für denselben Benutzer stehen, wird nicht benötigt. Die Provider tauschen sich über die Benutzer immer nur mithilfe der user handles aus, die bei der Verknüpfung erstellt worden sind bzw. zu einem späteren Zeitpunkt aktualisiert worden sind. Das in diesem Abschnitt beschriebene Konzept geht sogar so weit, dass die Benutzernamen zwischen den Providern überhaupt nicht ausgetauscht werden müssen. In den vier vorangegangenen Abbildungen ist das daran zu erkennen,

dass die Benutzernamen immer nur bei dem Provider stehen, bei dem sie für die direkte Anmeldung (Login bzw. Sign-On) gelten. Bei den anderen Providern taucht der Benutzername nirgendwo auf.

Aufhebung von Verknüpfungen

Der Benutzer erhält die Möglichkeit, einmal eingerichtete Verknüpfungen wieder aufzuheben. Dafür gibt es beim LAP ein eigenes Protokoll: das *Federation Termination Notification Protocol*., also das „Protokoll zur Unterrichtung über die Beendigung einer Verknüpfung". Die Aufhebung einer bestehenden Identitäts-Verknüpfung kann sowohl beim SP als auch vom IDP gestartet werden. Sie wird vom Benutzer ausgelöst, der bei einem Provider einen entsprechenden Aufhebungs-Link aktiviert. Der weitere Verlauf der Verknüpfungsaufhebung unterscheidet sich danach, ob sie bei einem IDP oder bei einem SP initiiert wurde.

1. Ist die Verknüpfungsaufhebung bei einem IDP initiiert worden, so informiert der IDP den SP darüber, dass er dem SP fortan keine Identitätsinformation über den betreffenden Benutzer zur Verfügung stellen wird und dass er auf keine Anfragen des SP mehr reagieren wird, die diesen Benutzer betreffen.
2. Ist die Verknüpfungsaufhebung bei einem SP initiiert worden, informiert der SP den IDP darüber, dass der Benutzer verfügt hat, dass der IDP dem SP fortan keine Identitätsinformation mehr über ihn zur Verfügung stellen soll, und dass der SP den IDP künftig zu keinerlei Aktion im Auftrag des Benutzers mehr auffordern wird.

Neben einer Sicherheitsabfrage, die vor Ausführung der Aufhebung den Benutzer zur Bestätigung auffordert, sollte der Benutzer direkt durch den Provider authentifiziert werden, bei dem die Aufhebung initiiert wird, d.h. der Benutzer sollte nach seinen lokalen Zugangsdaten gefragt worden sein. Dies stellt zum einen sicher, dass der Benutzer auch tatsächlich er selbst ist, zum anderen stellt es sicher, dass sich der Benutzer auch nach Beendigung der Verknüpfung bei diesem Provider lokal anmelden kann und sich nicht durch die Beendigung der Verknüpfung „ausschließt"; das gilt insbesondere für die Fälle, in denen die Beendigung der Verknüpfung bei einem SP initiiert wird, bei dem sich der Benutzer zuletzt nur noch über den IDP authentifiziert hat und möglicherweise die Zugangsdaten für den SP vergessen hat.
Weitere Fälle für die Beendigung von Verknüpfungen, die sogar automatisch und ohne Benutzereingriff ausgelöst werden könnten, könnten etwa zeitlich begrenzte Zugänge sein, die SP Ihren Benutzern einrichten. Das könnten zeitlich befristete Testzugänge oder Abonnements sein, die auslaufen, d.h. die Zugangsdaten werden danach ungültig, und nach Ungültigwerden, könnte der Vorgang zur Beendigung der Verknüpfung u. U. nicht mehr ordnungsgemäß durchgeführt werden, weil ein Zugang (eine Identität) nicht mehr gültig und aus dem Benutzerverzeichnis gelöscht worden sein kann.
Noch ein anderer Fall für die Aufhebung von Verknüpfungen entsteht, wenn zwei Provider ihre vertraglich geregelte Zusammenarbeit beenden. Dann entfällt die Grundlage für die Verknüp-

fung von Identitäten zwischen diesen beiden Providern und die bestehenden Verknüpfungen müssten entfernt werden.
[HoWa03]

Simplified Sign-On

Nachdem sich ein Benutzer bei einer *Liberty-fähigen* Site, die also das ID-FF unterstützt, eingeloggt hat, wird er bei anderen Liberty-fähigen Sites, die er im weiteren Verlauf einer Sitzung (Session) aufruft, >>automatisch und im Hintergrund eingeloggt<< ← (Das scheint der Unterschied zu sein), ohne sich erneut authentifizieren zu müssen. Simplified Sign-On funktioniert nicht nur innerhalb von Vertrauenskreisen, sondern auch über die Grenzen von Vertrauenskreisen hinaus. [LAP03]

In [LAP03] wird von *Simplified Sign-On* anstelle von *Single Sign-On* gesprochen. Es gibt darüber hinaus auch noch eine Sammlung von Präsentationsfolien [LAP06], die auf den Webseiten des LAP als Tutorial zur Verfügung gestellt wird. In diesem Dokument ist eingangs ebenfalls von Simplified Sign-On die Rede, und zwar im Rahmen einer Auflistung der Ziele des LAP. Insofern kann die Bezeichnung Simplified Sign-On als eine Art Vorstufe zum Single Sign-On verstanden werden. Weiterhin lässt die Tatsache, dass die Worte Simplified Sign-On in den englischsprachigen Quellen stets am Anfang eines Satzes oder als eigenständiger Punkt einer Auflistung erscheinen, die Vermutung zu, dass es sich hierbei nicht unbedingt um einen feststehenden Begriff mit Großschreibung handelt wie bei Single Sign-On, sondern zunächst um die Formulierung der Zielsetzung ein „vereinfachtes Sign-On“ zu ermöglichen, also: *simplified Sign-On*.

Fundamentales Session-Management

Session Management macht Single Sign-On möglich, indem es im Hintergrund den Anmeldestatus des Benutzers überwacht. Wenn der Benutzer das Angebot eines SP nutzen möchte, muss im Hintergrund geprüft werden, ob der Benutzer bereits bei einem IDP eingeloggt ist, dem der SP vertraut. Abhängig davon wird dem Benutzer der Zugriff auf das Angebot des SP gewährt oder nicht, und stattdessen eine Anmeldeseite präsentiert.
Beim Sign-Out muss das Session Management dafür sorgen, dass der Benutzer a) tatsächlich abgemeldet wird, d.h. die Sitzung beendet wird, und b) im Falle eines Single Sign-Out alle an der zu beendenden Sitzung beteiligten Provider informiert werden und diese den Benutzer ggf. lokal abzumelden und den Status auf „nicht angemeldet“ zu setzen. [LAP03]

Partnerschaften (Affiliations)

Verbünde von Providern ermöglichen, dass ein Benutzer seine Identität gleich mit einer ganzen Gruppe verbündeter Provider verknüpfen kann, statt mit einem einzelnen. Das macht es dem Benutzer leichter, alle Angebote verbündeter Provider wahrzunehmen, als würde er die Provider erst nach und nach kennenlernen und seine Identitäten untereinander zu verknüpfen. [LAP03]

Anonymität

In gewissen Situationen benötigt ein SP ganz bestimmte Detailinformationen über den Benutzer, um seinen Dienst zu erbringen, aber durchaus nicht das gesamte Paket an Detailinformationen, die der Benutzer dem IDP anvertraut hat. So könnte der Benutzer etwa für automatisierte Bezahlvorgänge seine Bankverbindung beim IDP hinterlegt haben, damit ein (berechtigter) SP diese Daten vom IDP beziehen kann, wenn der Benutzer beim SP einen erworbenen Artikel per Lastschriftverfahren bezahlen möchte. Diese Art Information ist nicht anonym, sondern ein direkter Bestandteil der Identität de Benutzers. Es gibt aber Dienste, die die Identitäten ihrer Benutzer nicht kennen müssen, sondern mit einer anonymen Teilinformation auskommen. So reicht für einen Wettervorhersagedienst die Angabe des Ortes, für den die Vorhersage gelten soll. Der Name des Ortes oder die Postleitzahl sind anonyme Informationen, die nicht auf die Identität des Benutzers schließen lassen. Das ID-FF bietet die Möglichkeit der anonymen Übermittlung bestimmter Attribute an Dienste, für die eine Identifizierung des Benutzers nicht erforderlich ist. [LAP03]

Protokoll zum Auffinden und zum Austausch von Meta-Daten in Echtzeit

Wenn die beteiligten Provider miteinander kommunizieren, müssen sie zuvor bestimmte Meta-Daten voneinander haben. Dazu zählen z.B. X.509-Zertifikate oder öffentliche Schlüssel, damit die Authentizität der ausgetauschten Nachrichten gewährleistet werden kann. [LAP03]

4.3.2 Identity Web Services Framework (ID-WSF)

Mit dem *Identity Web Services Framework* (ID-WSF) lassen sich Dienste erstellen, auffinden und benutzen, die Identitäts-bezogene, personalisierte Leistungen erbringen. Sie werden über Schnittstellen angesprochen, die in den *Identity Services Interface Specifications* (ID-SIS) (vgl. 4.3.3) definiert sind. Das ID-WSF stellt auch Mechanismen zur Weitergabe von Attributen bereit, deren Weitergabe der Benutzer zugestimmt hat, und Funktionen zum Einholen dieser Zustimmung. Ein Wetterdienst kann dadurch z.B. sehr einfach eine Wettervorhersage für den Wohnort des Benutzers machen, wenn er die Adresse aus den Attributdaten des Benutzers erfährt. Der Provider, der die Attributdaten zur Verfügung stellt, wird in diesem Zusammenhang auch *Attribut-Provider* genannt. Wie das zuvor beschriebene ID-FF, setzt auch das ID-WSF auf führende Industrie-Sicherheits-Standards auf und versucht, diese weiter zu entwickeln. [LAP03]

4.3.3 Identity Services Interfaces Specifications (ID-SIS)

Dieses Modul definiert eine Reihe von Schnittstellen zum ID-WSF (vgl. 4.3.2), mit dessen Hilfe Dienste aufgebaut werden können. Eine der ersten Schnittstellen-Spezifikationen ist die des Personal Profile Identity Service (ID-Personal Profile), der grundlegende Profil-Informationen über einen Benutzer bereitstellen kann. Alle diese Dienste sollen in ihrem Design auf Web Service

Standards aufsetzen, so dass sie mit SOAP und HTTP angesprochen und mithilfe von WSDL beschrieben werden können. [LAP03]

4.3.4 Industrie-Standards

Die drei oben genannten Module benutzen eine Reihe von bestehenden Standards. Das Aufgreifen bestehender Industriestandards – und ggf. die Erweiterung derselben – ist eine erklärte Vorgehensweise bei der Entwicklung der Liberty-Spezifikation. Zum einen wäre es mühsam und ineffizient, das Rad immer wieder neu zu erfinden, zum anderen dürfte die Akzeptanz der Spezifikation bei den Entwicklern und später auch bei den Benutzern dadurch von vorne herein höher sein, dass etablierte Standards verwendet werden. Wie SAML beim ID-FF setzt die Liberty Alliance auch WS-Security, HTTP, WSDL, XML, SOAP, XML-ENC, XML-SIG, SSL/TLS und WAP ein, die zum Teil in Kapitel 2 beschrieben wurden. Sie stammen entweder von der *Organization for the Advancement of Structured Information Standards* (OASIS), dem *World Wide Web Consortium* (W3C) oder der *Internet Engineering Task Force* (IETF).[LAP03]

4.4 Der Single Sign-On Vorgang

Beim LAP gibt es vier Varianten dafür, mit welcher Technik die Nachrichten des *Single Sign-On and Federation Protocol* zwischen Service Provider, Identity Provider und Client-Programm (User Agent) ausgetauscht werden. Diese Varianten werden als Profile bezeichnet und heißen im Einzelnen: *Browser Artifact Profile*, *Browser POST Profile*, *WML POST Profile* und *LECP (Liberty-enabled Client or Proxy) Profile* [LAP07b].

Im Grundprinzip funktionieren alle vier Profile nach demselben Schema in elf Schritten, das in Abbildung 4.5 dargestellt ist. Die Unterschiede bestehen in den Nachrichten, die zwischen den drei beteiligten Parteien ausgetauscht werden, und darin, welche der elf Schritte ausgeführt oder übersprungen werden. IDP und SP kommunizieren miteinander nur indirekt, durch das Client-Programm (WWW-Browser) hindurch. Es wird wie eine Relaisstation für die Nachrichten zwischen IDP und SP benutzt. Eine Ausnahme hiervon ist das *Browser Artifact Profile*, da hierbei die Schritte 8 und 9 vorkommen, in denen das Artefakt, das der IDP ausgegeben hat, vom SP direkt an den IDP zurückgesendet wird. Der IDP sendet daraufhin – ebenfalls direkt – die eigentliche Zusicherungsnachricht an den SP. [RoWa03]

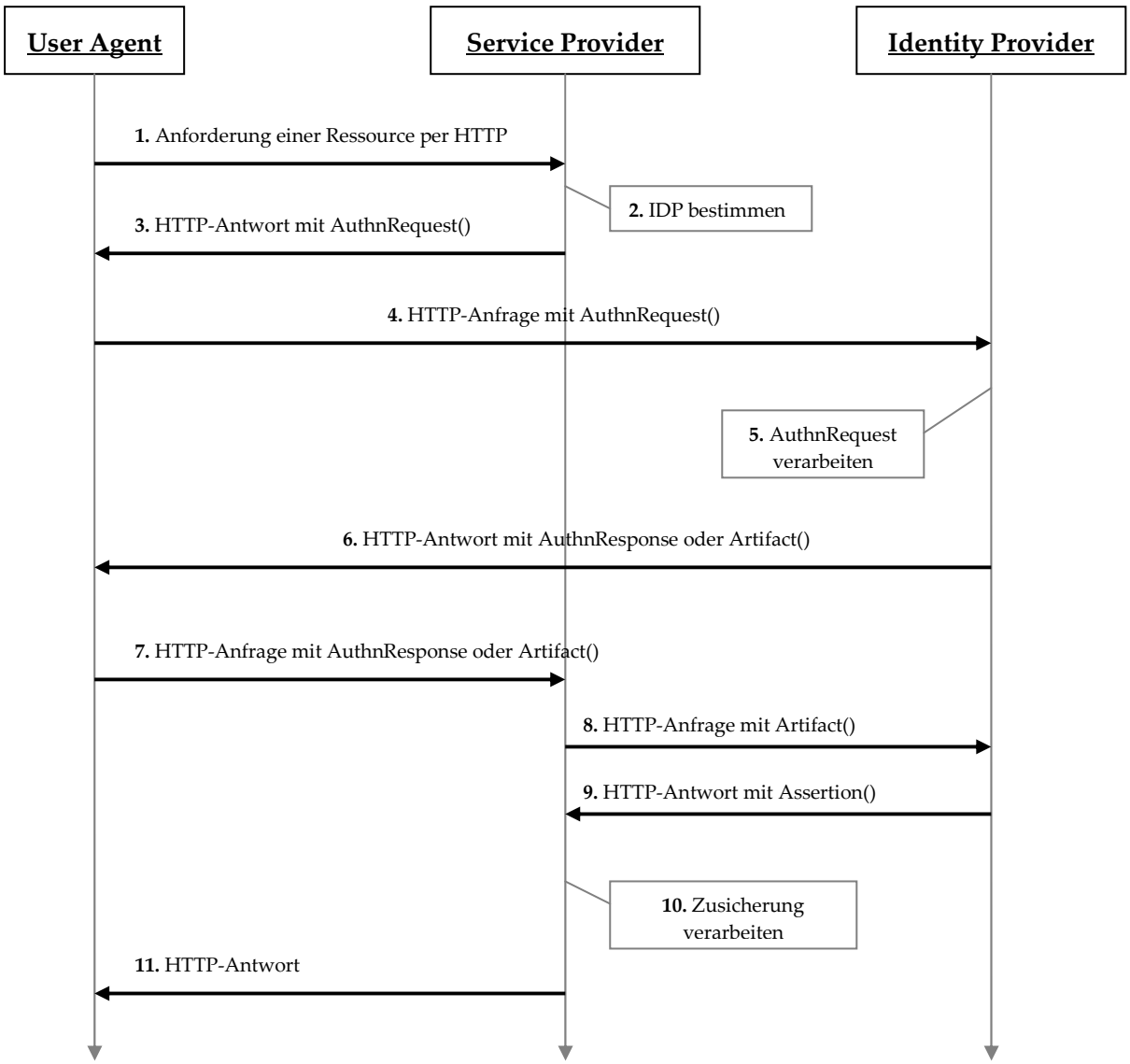

Abbildung 4.5: Ablaufschema beim Single Sign-On im LAP

[RoWa03]

Der Ablauf findet in bis zu elf Schritten statt:

1. **HTTP-Anfrage:**
   Der Client fordert vom SP eine zugriffsgeschützte Ressource an. Der SP stellt fest, dass der Benutzer bisher nicht authentifiziert ist, und daher noch keinen Zugriff auf die gewünschte Ressource haben darf.
2. **Bestimmen des Identity Providers, über den der Benutzer authentifiziert werden soll**
   Wie der Identity Provider bzw. der URL zu dessen Single Sign-On-Dienst ermittelt wird, bleibt dem Service Provider überlassen, hier gibt es keine Vorschriften durch das LAP.
3. **HTTP-Antwort mit der Authentifizierungsanforderung an den Client**
   Mit seiner Antwort auf die Anfrage aus 1. verweist der SP den Client an den Single Sign-On-Dienst des (ermittelten) IDP und fordert somit auf indirektem Weg eine Zusicherung

über die Authentizität des Benutzers beim IDP an. Form und Inhalt der Antwort hängen vom verwendeten Profil ab.

4. **HTTP-Anfrage an den Identity Provider mit der Authentifizierungsanforderung des SP**
5. **Bearbeitung der Authentifizierungsanforderung durch den Identity Provider**
   Falls sich der Benutzer gegenüber dem IDP noch nicht authentifiziert hat, kann der IDP den Benutzer jetzt dazu auffordern. Da beim LAP die Verknüpfung von Identitäten im Rahmen des Protokolls für Single Sign-On stattfindet, sei an dieser Stelle erwähnt, dass in so einem Fall der IDP in diesem Schritt auch die Bestätigung des Benutzers zur Verknüpfung zweier Identitäten einholen kann.
6. **HTTP-Antwort mit der Authentifizierungsantwort bzw. einem Artefakt an den Client**
   In diesem Schritt sendet der IDP dem Client eine Antwort auf die Anfrage aus Schritt 4. Form und Inhalt hängen vom verwendeten Profil ab. Im Fehlerfall muss der IDP auch eine Antwort schicken, die dann eine Fehlermeldung enthält, und so den Benutzer über das Scheitern informiert.
7. **HTTP-Anfrage mit der Authentifizierungsantwort bzw. einem Artefakt an den Service Provider**
   Der Client wendet sich nun mit der Authentifizierungsantwort bzw. dem Artefakt, das er in Schritt 6 vom IDP übermittelt bekommen hat an den SP. Form und Inhalt der Anfrage hängen vom verwendeten Profil ab.
8. **HTTP-Anfrage mit Artefakt durch den SP an den IDP (Auflösung / Dereferenzierung)**
   Dieser Schritt gilt nur in Verbindung mit dem *Browser Artifact Profile*, da es nur in diesem Profil ein SAML-Artefakt gibt, das der SP beim IDP gegen die eigentliche Authentifizierungsnachricht eintauschen muss. Der SP kommuniziert hierbei direkt mit dem IDP.
9. **HTTP-Antwort mit der Zusicherung des IDP an den SP**
   Der IDP hat das SAML-Artefakt dereferenziert und gegen eine für den SP verwendbare Zusicherung getauscht und sendet diese zurück an den SP.
10. **Verarbeitung der Zusicherung durch den SP**
    Der SP hat eine SAML-Zusicherung erhalten und prüft sie auf ihre Gültigkeit. Dann entscheidet sich für den SP wie er auf die ursprüngliche Anfrage des Client reagiert, der eine zugriffsgeschützte Ressource angefordert hatte.
11. **HTTP-Antwort an den Client**
    Im letzten Schritt bekommt der Client eine HTTP-Antwort auf seine ursprüngliche Anfrage. Hiermit wird dem Client entweder der Zugriff auf die ursprünglich angeforderte Ressource gestattet oder verweigert, falls sich der Benutzer nicht als zugriffsberechtigter Benutzer authentifizieren konnte.

[RoWa03]

Browser Artifact Profile

Dieses Verfahren entspricht dem *HTTP Artifact Binding* aus 2.7.3. Dabei antwortet der IDP dem WWW-Browser des Benutzers nach erfolgreicher Anmeldung mit einer speziellen http-Seite, die mit dem http-Code 302 versehen ist und somit eine Umleitung (Redirect), zu einem bestimmten anderen URI enthält. In diesem Fall besteht der URI aus dem URL für den Dienst beim SP, der für die Liberty-konforme „Assertion Consumer URL" steht, erweitert um zwei Argumente:

1. URL für den in Schritt 1 angeforderten Dienst oder Seite beim SP.
2. Artefakt - als künstlichen Parameter.

Das Artefakt.ist eine Art virtueller Gegenstand, das die Information über den aktuellen Login-Status des Benutzers beim IDP referenziert. Bei der Weiterleitung sendet der SP das Artefakt an den IDP zurück, diesmal auf dem direkten Weg, nicht durch den Browser des Benutzers, wodurch sich der Kreis schließt: Der IDP erkennt das von ihm selbst erzeugte Artefakt wieder, welches genau für diesen einen Zusicherungsvorgang gültig ist, und kann nun dem SP die Zusicherung über die korrekte Authentisierung des Benutzers machen. Die Zusicherung in Form einer SAML-Assertion. Das Artefakt ist verschlüsselt und quasi-zufällig und es kann nur einmal verwendet werden. Der Zufallsfaktor verhindert, dass das Artefakt von einem möglichen Angreifer erraten werden kann, und die einmalige Verwendbarkeit verhindert den Missbrauch eines abgehörten Artefakts durch Duplizieren und erneutes Zurücksenden an den IDP. [LAP07b][LAP06]

Browser POST Profile

Browser, die JavaScript oder ECMAscript unterstützen, können die erforderliche Weiterleitung alternativ durch den Einsatz von HTML-Formularen durchführen, um den Datenaustausch zwischen IDP und SP zu ermöglichen. Hierzu werden die weiterzuleitenden Daten in einer HTML-Seite in FORM-Elemente geschrieben und anschließend mit dem POST-Kommando verschickt, welches automatisch per JavaScript (bzw. ECMAscript) aufgerufen wird. Als Weiterleitungsziel wird der jeweils andere Provider angegeben, so dass auch hier die Daten zwischen SP und IDP durch den Browser des Benutzers als Relaisstation hindurch übertragen werden. [LAP07b][LAP06]

Diese Methode funktioniert aber nur, wenn der Benutzer einen Client mit (aktivierter) JavaScriptUnterstützung einsetzt. Clients, die diese Unterstützung nicht bieten können oder dürfen, müssen die Daten zwischen SP und IDP nach dem *Browser Artifact Profile* übertragen. Deshalb sollte das *Browser POST Profile* nicht als alleinige, sondern nur als zusätzliche Authentifizierungsmöglichkeit neben dem *Browser Artifact Profile* angeboten werden, um nicht Benutzer auszugrenzen, die kein JavaScript verwenden können – etwa weil das Clientprogramm keine Scriptunterstützung bietet oder diese aufgrund von Sicherheitsgründen abgeschaltet ist – und dadurch die Klientel a priori einzuschränken. [HoWa03]

WML POST Profile

Dieses Profil ist für den Einsatz auf Geräten gedacht, die einen WML-Browser (WML – Wireless Markup Language) verwenden, z.B. Mobiltelefone. Der Hauptunterschied zum *Browser POST Profile* ist, dass die Antworten des SP bzw. IDP an das vom Benutzer verwendete Programm nicht in HTML sondern in WML codiert ist. Gegenüber dem *Liberty-enabled client or proxy Profile* ist das WML POST Profile für die Verwendung mit unveränderten WML-Browsern ausgelegt. [HoWa03]

Liberty-enabled client or proxy (LECP)

Als *Liberty-enabled clients or proxies* werden solche Programme bezeichnet, die mit einer gewissen Intelligenz bzw. einem bestimmten Wissen ausgestattet sind, welches sie dazu befähigt, Funktionen der Liberty-Protokolle direkt zu unterstützen. Zum einen wissen sie, welchen IDP der Benutzer für welchen SP verwenden möchte, bzw. wie sie diese Information beziehen können. Zum anderen senden und empfangen sie in HTTP-Nachrichten verpackte Liberty-Nachrichten direkt. Dadurch sind sie nicht auf die sonst benutzten HTTP-Redirects angewiesen und unterliegen auch nicht den sonst gebotenen Längenbeschränkungen für Liberty-Nachrichten, da sie die Liberty-Nachrichten nicht als Parameter in URLs unterbringen müssen, wie es beim HTTP-Redirect Profile der Fall wäre. [HoWa03]

4.5 Single Logout

Der Single Logout[3] kann entweder beim IDP oder bei einem SP initiiert werden. In beiden Fällen führt das dazu, dass der IDP eine Logout-Nachricht an alle SP sendet, bei denen zur Zeit eine Session für den Benutzer läuft, die durch den IDP authentifiziert wurde. Wird der Single Logout beim IDP initiiert, kann dieser sofort die Logout-Nachrichten an die betreffenden SP senden. Wenn der Single Logout beim SP initiiert wird, muss dieser anschließend den IDP von diesem Vorgang unterrichten, so dass der IDP daraufhin die Logout-Nachrichten an die SP sendet. Single Logout bezieht sich immer auf die Sitzungen, die von einem bestimmten IDP authentifiziert wurden. Wenn ein Benutzer mehrere Single Sign-On Sitzungen mit unterschiedlichen IDP hat, bedeutet Single Logout nicht, dass damit sämtliche Sitzungen aus beiden SSO-Kontexten beendet werden, sondern nur die Sitzungen aus einem SSO-Kontext. [RoWa03]

Es gibt wie beim *Single Sign-On* drei verschiedene Verfahren (Profile), wie die Anfrage nach dem Single Logout gehandhabt werden kann. Es handelt sich dabei um die Profile *HTTP-Redirect*,

[3] Im Gegensatz zum Begriff *Single Sign-On* wird in den Spezifikationspapieren des LAP für den Abmeldevorgang der Begriff *Logout* und nicht *Sign-Out* verwendet.

HTTP-GET und *SOAP-über-HTTP*. Sie werden im Folgenden beschrieben. Außer in der verwendeten Technologie unterscheiden sie sich noch in ihrer Einsatzmöglichkeit: Während der Single Logout mit allen drei Verfahren beim Identity Provider initiiert werden kann, kann er beim Service Provider nur mit dem HTTP-*Redirect*- oder dem *SOAP über HTTP*-Verfahren eingeleitet werden. Service Provider hinterlegen, nach welchem Profil der Identity Provider den Single Logout bei ihnen aufrufen soll. So kann es sein, dass bei einem Single Logout eine Kombination der im Folgenden beschriebenen Profile zum Einsatz kommt. Sie sind hier zunächst für den Fall beschrieben, dass der Single Logout beim IDP initiiert wird. Danach wird kurz erläutert, worin die Unterschiede bestehen, wenn der Single Logout bei einem SP initiiert wird. [LAP07b]

HTTP-Redirect

Die folgende Abbildung 4.6 zeigt das Ablaufschema beim Single Logout mithilfe von HTTP-Redirects, wenn der Vorgang beim IDP ausgelöst wird. Der erste Schritt entspricht dem Klicken des Benutzers auf den *Logout*-Link auf der Webseite des IDP. In der Folge leitet der IDP den Browser des Benutzers zum Single-Logout-Service des ersten SP per HTTP-Redirect um. Der SP loggt den Benutzer lokal aus und leitet den Browser zurück an den IDP. Die Schritte 2 bis 6 wiederholen sich nun für jeden weiteren SP, dem der IDP im Laufe der Session eine Authentifizierung des Benutzers gegeben hat und der nach dem HTTP-Redirect-Profil bedient werden möchte. Am Schluß sendet der IDP dem Browser noch eine Bestätigungsmeldung, dass der Single Logout ausgeführt wurde.

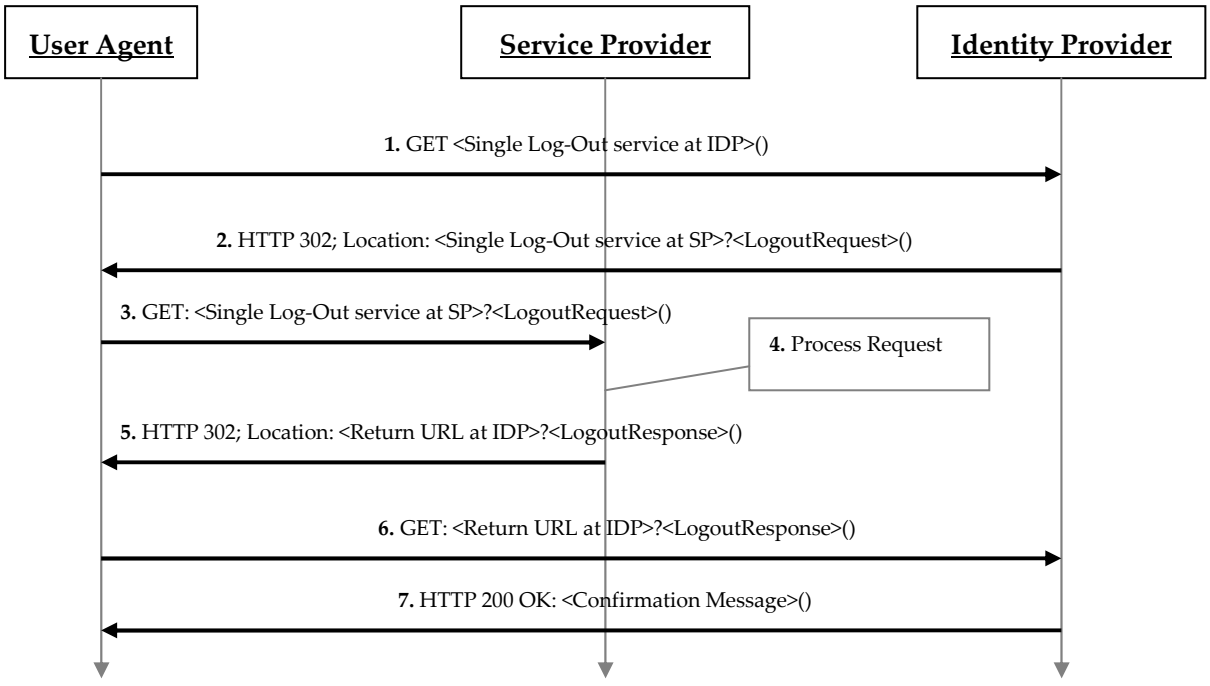

Abbildung 4.6: Single Logout beim IDP per HTTP-Redirect

[RoWa03]

HTTP-GET

Der Single Logout funktioniert auch mit HTTP-GET Befehlen, wenn er beim IDP initiiert wurde. Abbildung 4.7 zeigt das Ablaufschema. Es unterscheidet sich dadurch von der Methode mit HTTP-Redirect, dass der URL des Single-Logout-Services des SP durch den Browser des Benutzers nicht dadurch aufgerufen wird, weil der Browser umgeleitet wird, sondern weil der Browser vom IDP eine HTML-Seite bekommt, die einen Verweis auf ein Bild enthält. Beim Nachladen des Bildes wird nicht direkt eine Grafikdatei abgerufen sondern der Single-Logout-Service des SP aktiviert. Als Ergebnis des lokalen Logout beim SP sendet der Single-Logout-Service eine Grafik zurück, die den Erfolg des lokalen Logout dokumentiert. Beim Senden der Grafik kommt wieder ein HTTP-Redirect zum Einsatz, wodurch die Grafik an den IDP übertragen wird. Nachdem die Schritte 2 bis 6 für jeden SP wiederholt wurden, der nach dem HTTP-GET-Profil bedient werden möchte und vom IDP in der laufenden Session eine Authentifizierung des Benutzers erhalten hat, wird eine Seite zur Bestätigung des Single-Logout an den Browser des Benutzers geschickt.

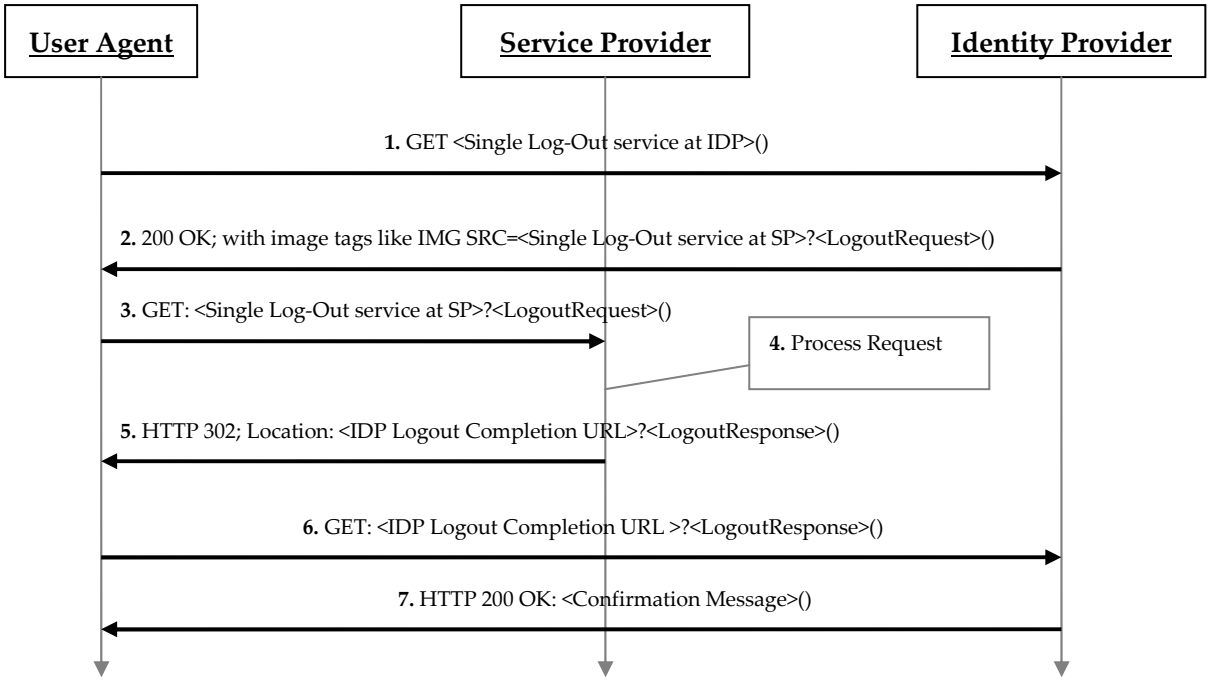

Abbildung 4.7: Single Logout beim IDP per HTTP-GET

[RoWa03]

SOAP-über-HTTP

Nachdem der Benutzer den Single-Logout beim IDP initiiert hat, sendet der IDP an jeden SP eine in HTTP verpackte SOAP-Nachricht auf direktem Wege, ohne Umleitung durch den Browser des Benutzers (vgl. Abbildung 4.8). Wenn der SP den Benutzer bei sich lokal ausgeloggt hat, sendet er ebenfalls eine in HTTP verpackte SOAP-Nachricht an den IDP zurück. Zum Schluss sendet der IDP eine Seite an den Browser, die den erfolgten Single-Logout bestätigt.

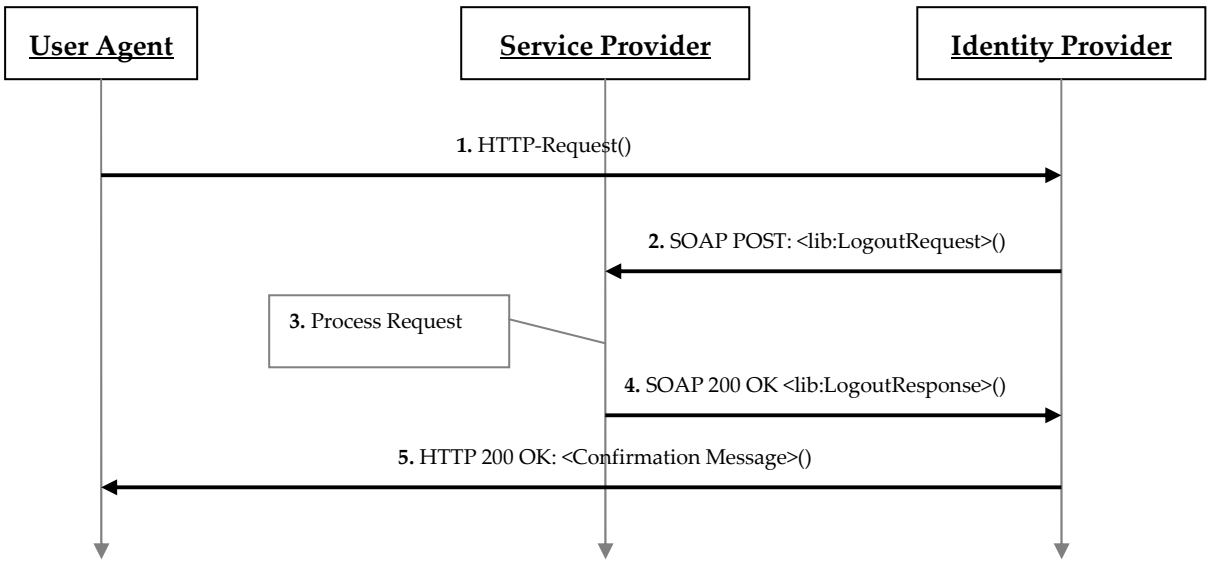

Abbildung 4.8: Single Logout beim IDP per SOAP-over-HTTP

[RoWa03]

Single Logout beim Service Provider initiiert

Beim SOAP-über-HTTP-Profil sendet der SP, bei dem der Benutzer den Single Logout initiiert hat, ein SOAP-Nachricht an den IDP. Der IDP bearbeitet die Anfrage nach dem Single Logout und sendet allen anderen SP, die innerhalb dieser Session eine Authentifizierung vom IDP erhalten haben eine Logout-Aufforderung nach deren bevorzugtem Profil. Nachdem alle Logouts erfolgt sind, antwortet der IDP dem SP, bei dem der Benutzer den Single Logout initiiert hat, und dieser sendet eine Bestätigungsseite an den Browser.

Beim HTTP-Redirect-Profil funktioniert ein beim SP initiierter Single Logout so, dass der SP den Benutzer bei sich lokal abmeldet und dann den Browser des Benutzers an den Single Logout-Service des IDP umleitet. Der IDP bearbeitet die Anfrage wiederum für alle SP, die eine Authentifizierung erhalten haben, und sendet ihnen jeweils eine Logout-Aufforderung gemäß dem bevorzugten Profil.

5 Zusammenfassung und Ausblick

Single Sign-On-Technologien versprechen dem Nutzer des World Wide Web eine Vereinfachung bei der Authentifizierung gegenüber Websites und Webservices, da er sich nur noch einmal pro Sitzung authentifizieren muss, statt sonst einzeln gegenüber jeder Website/jedem Webservice. Dahinter steht das Problem der Handhabung einer immer größer werdenden Menge an Zugangsdaten, der sich jeder Benutzer des WWW gegenüber sieht, der sich bei verschiedenen Webseiten und Webservices registrieren muss, um die angebotenen Leistungen wahrnehmen zu können.

Den Technologien, die Single Sign-On ermöglichen, liegen verschiedene Ansätze mit Gemeinsamkeiten und Unterschieden zugrunde. Am Single Sign-On sind immer drei Rollen beteiligt: ein *Client*, mit dem der Benutzer interagiert, ein *Identity Provider*, der Identitätsdaten bereitstellt und für die Richtigkeit garantiert, und ein *Service Provider* – auch *Relying Party* genannt – (Website oder -service), der dem Identity Provider vertraut. Bei jeder SSO-Technologie müssen sich IDP und SP darüber verständigen können, um welchen Benutzer es beim Ein- oder Ausloggen geht. Dazu werden abstrakte Identifikatoren oder Handles erzeugt, die innerhalb ihres Kontextes eindeutig sind. Benutzernamen garantieren diese Eigenschaft nur für den Kontext eines einzelnen Providers und können daher nicht zur Abstimmung zwischen zwei Providern herangezogen werden. Darüber hinaus kann die Verwendung abstrakter Identifikatoren für eine gewisse Anonymisierung gegenüber dem Service Provider sorgen.

Beim zentralistisch ausgelegten .NET Passport wird ein einziger Identifikator pro Benutzerkonto erzeugt. Er wird beim SSO an den Service Provider übermittelt, sodass dieser für den Benutzer in seinem Benutzerverzeichnis einen Eintrag mit demselben Identifikator führen kann. Damit ist der Datenabgleich zwischen den verschiedenen Service Providern zwecks Profilerstellung möglich. Andererseits kann sich ein Benutzer von .NET Passport bei Service Providern einloggen und deren Dienste nutzen, ohne sich explizit bei ihnen zu registrieren. Das geschieht dort nämlich im Hintergrund.

Das Liberty Alliance Project setzt auf die Verknüpfung der verschiedenen Identitäten eines Benutzers, die er aufgrund seiner Registrierungen bei unterschiedlichen Service Providern hat. Hier gibt es keinen Identifikator, der im gesamten Umfeld eines Identity Providers Gültigkeit hat. Ein Identity Provider erzeugt pro Benutzer und Service Provider einen eigenen Identifikator, der nur zwischen diesen beiden Providern gültig ist. Überhaupt kann es beliebig viele Identity Provider geben. Jeder Service Provider kann entscheiden, welchem Identity Provider er das nötige Vertrauen entgegenbringt, die Benutzer zuverlässig zu authentifizieren. Er darf beliebig vielen Identity Providern vertrauen. Es gibt dadurch aber die Möglichkeit, dass sich viele Cluster bilden, die sich jeweils um einen oder mehrere Identity Provider gruppieren. Sie werden auch *Circles of Trust* (Vertrauenskreise) genannt. Solange sich ein Benutzer innerhalb eines Vertrauenskreises bewegt, ist für ihn Single Sign-On Realität. Sobald er sich aber in mehreren Vertrauenskreisen bewegt und er sich pro Vertrauenskreis einmal authentifizieren muss, ist Single Sign-On für ihn nicht mehr

gegeben, obwohl alle Provider, mit denen er zu tun hat, nach den Spezifikationen des Liberty Alliance Projects arbeiten. Hier ist die zentralistische Ausrichtung von .NET Passport im Vorteil. Wann immer ein Service Provider .NET Passport unterstützt, nutzt er die zentral bei Microsoft gespeicherte Benutzerdatenbasis und befindet sich dadurch automatisch in einem Vertrauenskreis mit allen anderen Providern im Kontext von .NET Passport. Eine einzige Datenbasis hat auch den Vorteil, dass es keine Konsistenzproblem durch verteilte Datenhaltung gibt.
Aber spätestens seitdem .NET Passport offiziell als gescheitert gilt, ist klar geworden, dass es keine einzelne Institution – weder politisch noch wirtschaftlich noch gemeinnützig – geben darf, die die alleinige Kontrolle über die Identitätsprofile der Internetnutzer hat und von allen gleichermaßen akzeptiert ist. Das politische und wirtschaftliche Potenzial, das einer nahezu vollständigen Sammlung aller Profildaten sämtlicher Internetnutzer innewohnt, ist einfach zu groß als dass nicht bei den Benutzern Zweifel an der Reichweite der Integrität des Betreibers aufkommen würden. Denn egal, um welche Single Sign-On-Technologie es sich auch immer handelt, sie funktioniert nur mit dem Vertrauen der Benutzer. Sie müssen nämlich entscheiden, ob sie solch einem umfassend konzipierten System ihre persönlichen Daten anvertrauen oder nicht. Daher muss es im Interesse eines jeden SSO-Anbieters liegen, die Funktionsweise seines Systems offenzulegen, damit jeder Benutzer die Möglichkeit erhält es zu überprüfen und zu entscheiden, ob er dem System vertraut.

Mit dem Fortschreiten der Entwicklung der verschiedenen Ansätze zeichnet sich eine Konvergenz der Entwürfe ab. Das ist unter anderem den Pressemeldungen über Kooperationen der Entwickler von Identitätsmanagementsystemen und Single Sign-On-Technologie zu entnehmen, aber auch der Beobachtung, dass sowohl die Liberty Alliance als auch Microsoft die Weiterentwicklung offener Sicherheitsstandards voran treibt. Besonders Microsoft ist mit *InfoCard* auf ein dezentral organisiertes SSO-System umgestiegen. Zwar läuft der .NET Passport-Dienst unter dem Namen *Windows Live ID* noch weiter, aber die Entwicklungsarbeit wird in InfoCard, eine Art Metasystem, gesteckt. Das Internet wurde ohne integriertes Identifizierungs- und Authentifizierungssystem entwickelt und sämtliche Mechanismen, die es im Internet in dieser Richtung gibt, sind spezielle, anwendungsbezogene Lösungen. Die Idee hinter dem *Identity Metasystem* ist, in die Internetkommunikation eine Abstraktionsebene einzuführen. Es soll dabei den *Laws of Identity* genügen, einer Ansammlung von sieben Anforderungen, die sich in Bezug auf Identität herauskristallisiert haben und unseren täglichen Umgang mit Identität im reellen Leben abseits des Internet widerspiegelt.
Danach sollte auch die weitestgehend anonyme Nutzung des Internet weiterhin möglich sein, ebenso wie man anonym durch eine Fußgängerzone gehen kann. Aber immer mehr Dienstanbieter verlangen zur Nutzung ihrer Dienste die vorherige Registrierung und Authentifizierung des Benutzers. Die anonyme Nutzung ist dann nur durch das Anlegen spezieller *Dummy-Accounts* mit bewusst falschen Angaben zur Person möglich. An dieser Stelle unterstützt der InfoCard-Entwurf den Benutzer, indem er vorsieht, dass der Benutzer selbst in die Rolle eines Identity Providers schlüpfen kann und sich selbst Identitätskarten ausstellt.

Mittlerweile wird schon viele Jahre an Single Sign-On-Lösungen gearbeitet und es gibt auch Produkte, z.B. *One* von Sun Microsystems oder .NET Passport von Microsoft, aber bislang hat sich noch keine Lösung flächendeckend etablieren können. .NET Passport bzw. heute Windows Live ID hat zwar nach eigenen Angaben über 300 Mio. Nutzer, aber diese hohe Zahl ist sicherlich dem Umstand geschuldet, dass alle Benutzerzugänge von Microsoft-Diensten wie Hotmail oder MSN-Network in *Passports* umgewandelt wurden, die dann auch für SSO genutzt werden konnten. Eine sehr große Benutzerzahl hat sich also nicht von sich aus für ein Passport registriert und es bleibt Spekulation, ob sie diesen Schritt von sich aus gegangen wären.
Die Nutzung von SSO scheint einerseits zum Greifen nah zu sein, andererseits bekommt man als Internetnutzer derzeit nicht den Eindruck, als sei SSO überall verfügbar und unterstützt. Das mag auch daran liegen, dass die Entwicklung in dem Bereich auch noch ziemlich in Bewegung ist, und dass die Betreiber von Websites deshalb noch zögern, bevor sie SSO implementieren. Aus Betreibersicht ist es nachvollziehbar, dass keiner voreilig auf das „falsche Pferd“ setzen will, indem er sich jetzt für eine Technologie entscheidet, die vielleicht noch von einer anderen überholt wird. Außerdem nützt die SSO-Technologie einem einzelnen Provider sowieso nichts, wenn nicht seine Geschäftspartner auf ihren Websites dieselbe SSO-Technologie einsetzen. Es bedarf vermutlich noch etwas Zeit bis zur flächendeckenden Verfügbarkeit von SSO. Der jüngste, von Microsoft initiierte Ansatz mit dem Identity Metasystem und InfoCard klingt vielversprechend, weil er ungewohnt offen daher kommt. Skepsis gegenüber einer neuen Technologie ist aber durchaus angebracht, besonders wenn sie in diesem Fall mit der Authentifizierung per Benutzername und Passwort bricht. Dass damit der Sumpf der Phishing-Attacken ausgetrocknet werden soll, ist sicherlich aller Ehren wert, aber dafür muss die neue Technologie umso genauer auf ihre Schwachstellen überprüft werden.

5.1 Chancen und Risiken von Single Sign-On

So verheißungsvoll die Vorstellung auch ist, sich bequem im World Wide Web zu bewegen, so sehr sollten die damit verbundenen Vor- und Nachteile ins Bewusstsein derjenigen rücken, die Single Sign-On verwenden. Die Parameter Sicherheit und Bequemlichkeit hängen von einander ab. Ein einfach zu merkendes oder zu tippendes Passwort ist eine bequeme Sache, aber meistens ist es dann auch einfach zusammengesetzt, ein lexikalisches Wort oder schlicht zu kurz und damit nicht sicher. Wer SSO benutzt sollte sich darüber im Klaren sein, dass seinen Zugangsdaten zu so einem System eine noch größere Bedeutung zukommt als handelte es sich nur um die Zugangsdaten zu einem Internetforum zu seinem Hobbythema. Andererseits kann die Bündelung durch SSO auf ein Zugangsdatenpaar die Sicherheit bei richtiger Anwendung auch verbessern, wenn nämlich die Reduzierung dazu führt, dass der Benutzer sich ein etwas komplexeres Passwort merken kann, weil er sich nicht mehr mehrere merken muss, die er deshalb leichter wählen würde.

Gerade die zentrale Bedeutung eines SSO-Zugangs macht ihn wiederum für Angreifer interessant. Die mehr oder weniger umfangreichen Sammlungen von Identitätsdaten bei den Identity Providern sind besonders interessante Angriffsziele und insofern mehr denn je zu schützen. Hier stehen die Identity Provider in der Pflicht, einerseits ihre Leitungen und ihre Server bestmöglich zu schützen. Andererseits sollten sie den Benutzer bei der Absicherung seines Zugangs z.B. in der Wahl des Passwortes insoweit unterstützen, dass sie dessen Komplexität bestimmen und mitteilen, als wie sicher es einzustufen ist.
Ein anderer Punkt ist die Profilerstellung über SSO-Benutzer, wenn diese ihre Online-Tätigkeiten über einen zentralen Zugang abwickeln. Dass die Bewegungen im Cyberspace nachvollziehbar sind, sollte nicht verdrängt werden. Möglicherweise besteht aber gerade durch einen zentralen SSO-Zugang die Chance, genau dieses Bewusstsein zu stärken.

5.2 Offene Fragen

In den Anfängen der Single Sign-On-Bemühungen gab es gerade bei .NET Passport mit dem Kids-Passport eine Initiative für spezielle SSO-Zugänge für Kinder. Davon ist bei anderen Anbietern noch nicht viel angekommen. Das liegt möglicherweise daran, dass die SSO-Technologie sich bislang noch nicht hat durchsetzen können. Spezielle Sicherheit für Kinder und Jugendliche, die die Folgen von abfließenden Identitätsdaten noch viel weniger absehen können als Erwachsene, sollte jedoch ein Aspekt sein, auf den die Entwicklung eingeht. Mit diesem Blickwinkel lassen sich eventuelle Schwachpunkte der Entwürfe möglicherweise besser erkennen.
Offen ist bislang auch, wie es realisiert werden soll, dass Benutzer die Authentizität von Providern selbst überprüfen können. Sicherheitszertifikate sind bislang für die meisten Anwender unverständlich und somit zur Überprüfung ungeeignet.

5.3 Weitere Ansätze

Neben den in dieser Arbeit behandelten Ansätze .NET Passport (bzw. Windows Live ID), InfoCard und Liberty Alliance Project gibt es wie eingangs bereits erwähnt noch weitere. An dieser Stelle soll noch kurz auf Shibboleth von Internet2 (http://shibboleth.internet2.edu) und OpenID (http://www.openid.net) verwiesen werden. Sie werden in den einschlägigen Medien häufig erwähnt. OpenID hat zuletzt mit der Zusammenarbeit mit Microsoft in Sachen SSO Schlagzeilen gemacht.

Literaturverzeichnis

[ACKM04] Alonso, Gustavo; Casati, Fabio; Kuno, Harumi; Machiraju, Vijay : *Web Services*. . Berlin, Heidelberg, New York : Springer, 2004. - ISBN 3-540-44008-9

[BeKe03] Beatty, John D.; Kemp, John: *Liberty Protocols and Schema Specification, Version 1.1.* - Aktualisierungsdatum: 15.01.2003.

[Cam05] Cameron, Kim: The Laws of Identity. URL: http://msdn2.microsoft.com/en-us/library/ms996456.aspx.- Aktualisierungsdatum: 09.09.2007.

[CHK+05] Cantor, Scott et al.: *Binding for the OASIS Security Assertion Markup Language (SAML) V2.0*. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf.- Aktualisierungsdatum: 15.03.2005 13.29 Uhr.

[Chow06] Chow, Trevin: Windows Live ID and Passport. URL: http://trevin.spaces.live.com/blog/cns!C97E92F3E4DAD144!686.entry.- Aktualisierungsdatum: 17.09.2007.

[Cou04] Couvreur, Julien: Curiosity is bliss. URL: http://blog.monstuff.com/archives/000167.html.- Aktualisierungsdatum: 17.09.2007.

[HaMa02] Hallam-Baker, Phillip; Maler, Eve: The Organization for the Advancement of Structured Information Standards [OASIS] (Hrsg. Bd.) : *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML).* , 2002

[Hick03] Hicks, Terry Allan: Security Flaw Shows Microsoft Passport Identities Can't Be Trusted, 2003. - FT-20-0850

[HoWa03] Hodges, Jeff; Wason, Tom (Hrsg.): Liberty Architecture Overview - Version 1.1, 15 Jauary 2003.

[Hugh06] Hughes, Greg: Windows LiveID - It went live and you didn't even know it.. URL: http://www.greghughes.net/rant/WindowsLiveIDItWentLiveAndYouDidntEvenKnowIt.aspx.- Aktualisierungsdatum: 20.04.2007.

[Kauf06] Kaufmann, Joachim: Microsoft: Passport wird zu Windows Live ID. URL: http://www.zdnet.de/news/print_this.htm.?pid=39141612-39001023c.- Aktualisierungsdatum: 17.09.2007.

[Kob99] Kobert, Thomas: *XML.* : Das bhv Taschenbuch. 1. Aufl. Kaarst : bhv, 1999. - ISBN 3-8287-5044-3

[KoRu00] Kormann, David P.; Rubin, Aviel D.: *Risks of the Passport Single Signon Protocol.* URL: http://avirubin.com/passport.- Aktualisierungsdatum: 15.05.2003.

[Kun03] Kunze, Christian Philip: *Digitale Identität und Identitäts-Management.* Hamburg, Universität Hamburg, Fachbereich Informatik, 2003

[Kuri99] Kuri, Jürgen: *Standard für Web-Anwendungen vorgeschlagen.* URL: http://www.heise.de/newsticker/meldungen/print/7102.- Aktualisierungsdatum: 16.09.2007.

[LAP03] Liberty Alliance Project: *Introduction to the Liberty Alliance Identity Architecture - Revision 1.0, March 2003.* URL: http://www.projectliberty.org/index.php/liberty/content/download/383/2708/file/LAP%20Identity%20Architecture%20Whitepaper%20Final.pdf.- Aktualisierungsdatum: 29.03.2007.

[LAP06] Liberty Alliance Project: *Liberty Specs Tutorial.* URL: http://www.projectliberty.org/index.php/liberty/content/download/423/2832/file/tutorialv2.pdf.- Aktualisierungsdatum: 01.11.2006.

[LAP07a] Liberty Alliance Project: *Current Members / Membership / Home - Liberty Alliance.* URL: http://www.projectliberty.org/liberty/membership/current_members.- Aktualisierungsdatum: 29.03.2007.

[LAP07b] Liberty Alliance Project: *Liberty ID-FF Architecture Overview.* URL: http://www.projectliberty.org/liberty/content/download/318/2366/file/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf.- Aktualisierungsdatum: 30.03.2007.

[Min03] Mink, Martin: *Sicheres Single Sign-On für Webdienste.* Darmstadt, Technische Universität Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, 2003. - KOM-D 197

[MS03] Microsoft: .NET Passport Service Guide Kit – Version 2.5.
Kompilierte HTML Hilfedatei aus dem .NET Passport SDK 2.5 vom 16.05.2003

[MS07] Microsoft: *Windows Live: Architectural Overview of Web Authentication.* URL: http://msdn2.microsoft.com/en-us/library/bb676631(d=printer).aspx.- Aktualisierungsdatum: 24.09.2007.

[Rag+06] Ragouiz, Nick et al.: *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. URL: http://www.oasis-open.org/committees/download.php/20645/sstc-saml-tech-overview-2%200-draft-10.pdf.- Aktualisierungsdatum: 09.10.2006 bzw. 22.04.2007 21:13 Uhr.

[RFC1630] Berners-Lee, T.: Universal Resource Identifiers in WWW. URL: http://www.ietf.org/rfc/rfc1630.txt.- Aktualisierungsdatum: 20.12.2006.

[RFC2119] Bradner, S.: Key words for use in RFCs to Indicate Requirement Levels. URL: http://www.ietf.org/rfc/rfc2119.txt.- Aktualisierungsdatum: 10.11.2006.

[RFC2396] Berners-Lee, T.: Uniform Resource Identifiers (URI): Generic Syntax. URL: http://www.ietf.org/rfc/rfc2396.txt.- Aktualisierungsdatum: 10.11.2006.

[RFC2616] Fielding, R. et al.*: RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1*. URL: http://www.ietf.org/rfc/rfc2616.txt.- Aktualisierungsdatum: 19.04.2007 16:55.

[RFC2965] Kristol, D.; Montulli, L.: HTTP State Management Mechanism. URL: http://www.ietf.org/rfc/rfc2965.txt.- Aktualisierungsdatum: 10.11.2006.

[RFC4346] Dierks, Tim; Rescorla, Eric: *The Transport Layer Security (TLS) Protocol Version 1.1*. URL: http://www.ietf.org/rfc/rfc4346.txt.- Aktualisierungsdatum: 31.03.2007 - 10:54.

[RoWa03] Rouault, Jason; Wason, Tom: Liberty Bindings and Profiles Specification - Version 1.1. URL: http://www.project-liberty.org/liberty/content/download/1469/9656/file/liberty-architecture-bindings-profiles-v1.1.pdf.- Aktualisierungsdatum: 15.01.2003.

[TLSC06] Internet Engineering Task Force (Hrsg.): *Transport Layer Security (tls) Charter*. URL: http://www.ietf.org/html.charters/tls-charter.html.- Aktualisierungsdatum: 31.03.2007 10:55.

[W3C06] Bray, Tim et al.: *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. URL: http://www.w3c.org/TR/2006/REC-xml-20060816/.- Aktualisierungsdatum: 31.03.2007 11:44.

[W3C07-a] Gudgin, Martin et al.: *SOAP Version 1.2 Part 1: Messaging Framework*. URL: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.- Aktualisierungsdatum: 14.04.2007 12:09 Uhr.

[Wat07] Watling, Nigel: Windows CardSpace and the Identity Metasystem. URL: http://cardspace.netfx3.com/files/folders/7653/download.aspx.- Aktualisierungsdatum: 15.09.2007.

[Wes02] Westphal, Ralf : *.NET kompakt / Ralf Westphal.* . Heidelberg ; Berlin : Spektrum, Akad. Verlag, 2002. - ISBN 3-8274-1185-8

[WHN06] Watling, Nigel; Harjanto, Andy; Nanda, Arun: *InfoCardExplained*. URL: mms://wm.microsoft.com/ms/msnse/0602/27129/InfoCard_Explained_Final_MBR.wmv - Aktualisierungsdatum: 22.09.2007

[Wik07] Wikipedia: XML Schema. URL: http://de.wikipedia.org/wiki/XML_Schema.- Aktualisierungsdatum: 13.04.2007.

[Wik07b] Wikipedia: Transport Layer Security. URL: http://de.wikipedia.org/wiki/Transport_Layer_Security.- Aktualisierungsdatum: 31.03.207.

[Wiki07-a] Wikipedia: Dokumenttypdefinition. URL: http://de.wikipedia.org/wiki/Document_Type_Definition.- Aktualisierungsdatum: 13.04.2007 19:19 Uhr.

[Wiki07-b] Wikipedia: SOAP. URL: http://de.wikipedia.org/wiki/SOAP.- Aktualisierungsdatum: 06.04.2007 14:28 Uhr.

[Wilk02] Wilkens, Andreas: Neuer Web-Sicherheitsstandard von IBM, Microsoft und Verisign. URL: http://www.heise.de/newsticker/data/anw-11.04.02-000/.- Aktualisierungsdatum: 04.07.2003.

[Wis+05] Wisniewski, Thomas et al.: *SAML V2.0 Executive Overview.* URL: http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf.- Aktualisierungsdatum: 22.04.2007 21:10 Uhr.

[WNB06] Watling, Nigel; Nanda Arun; Bhargava, Ruchi: URL: InfoCard – Deep Architecture. mms://wm.microsoft.com/ms/msnse/0605/27723/Arun_Deep_InfoCard_mbr.wmv - Aktualisierungsdatum: 26.09.2007

[Wolf07] Wolff, Markus: Microsoft gibt SSO-Service für Drittanbieter frei. URL: http://www.php-center.de/beitraege/detail.php?a_id=1259.- Aktualisierungsdatum: 28.08.2007.

Abbildungsverzeichnis

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Außerdem erkläre ich, dass ich mit der Einstellung dieser Studienarbeit in den Bestand der Bibliotheken der Universität Hamburg einverstanden bin.

Hamburg, den 28.09.2007

[Marc Schönberg]