Studienarbeit

Webservice Autorisierung mit Attributzertifikaten

Ingo Kampe

Sommer 2009

Institut für Informatik

Systemarchitektur

Inhaltsverzeichnis

1. Einleitung

1.1. „Das Netz ist der Computer“ (Sun Microsystems)

Das Internet verändert nicht nur die Art der Kommunikation und des Datenaustausches von Menschen und Maschinen sondern mittlerweile wesentliche Merkmale der Softwarearchitektur selbst. Während früher die Schnittstellen zur Dateneingabe und zur Ergebnisausgabe auf ein geschlossenes Rechnersystem beschränkt waren, haben sich heute auf Basis des Internet Protokolls verschiedene Mechanismen etabliert, um zwischen beliebigen Systemen über beliebige Entfernungen Interaktionen zu ermöglichen. Diese so genannten verteilten Systeme stellen völlig neue Herausforderungen an das Softwaredesign, an die eingesetzten Protokolle und an die IT-Sicherheit.

Software läuft nicht mehr unbedingt als abgeschlossene Prozedurabfolge eines lokalen Prozessors, sondern kann, in logischer Folge einer immer stärkeren Modularisierung, in ihrem Ablauf und mit ihren Komponenten auf verschiedene vernetzte Systeme verteilt werden. Hierbei werden verschiedene Ziele verfolgt. Von der Steigerung der Performance durch Parallelisierung (Loadbalancing), der Verbesserung der Verfügbarkeit durch redundante Bestandteile (High Availibity), der Umsetzung neuer Geschäftsmodelle und der Verlagerung von Administrationsaufwänden bis zur Spezialisierung auf einzelne Funktionen aus (patent-) rechtlichen oder Softwaredesigngründen (Modularität, Wiederverwendbarkeit, Neukomposition).

Die Protokolle sind meist offengelegt und standardisiert, da die beteiligten Kommunikationspartner in ihrer Anzahl und Vielfalt stark zugenommen haben. Weiterhin ermöglichen diese eine plattformunabhängige Nutzung, um flexibel beliebige Betriebssysteme auf verschiedenster Hardware zusammenarbeiten zu lassen. Um die Anforderungen an die Kommunikationssicherheit zu erfüllen, unterstützen bereits die Protokolle entsprechende Verfahren und Formate.

Völlig neue Sicherheitsanforderungen ergeben sich durch die Kommunikation über ein offenenes Medium. Einfache organisatorische Maßnahmen reichen nicht mehr aus, um die Authentizität der Beteiligten sicherzustellen. Manipulationen und Kopien sind in der digitalen Welt wesentlich vereinfacht und können ohne kryptografische Verfahren nicht entdeckt oder verhindert werden. Zahlreiche Verfahren versuchen die entsprechenden Sicherheitsziele für bestimmte Bereiche und Anwendungsfälle umzusetzen.

Eine Möglichkeit Authentizitätsregeln zu transportieren stellen X.509 Attributzertifikate dar, die in dieser Arbeit angewendet werden im Umfeld von SOAP-basierten Webservices.

1.2. Serviceorientierte Architektur (SOA) und Webservices

SOA ist aktuell ein sehr häufig genutzter Begriff mit dem eine konzeptionelle Idee zur Umsetzung der neuen vernetzten Welt bei der Abbildung von konkreten Geschäftsprozesse in die IT

Systemarchitektur beschrieben wird. In [11] wird SOA als Konzept definiert, das auf einen Service als wohldefinierte in sich geschlossenen Funktionseinheit unabhängig vom Kontext oder Zustand anderer Services basiert. Hier wird herausgestellt, dass aus Sicht der IT SOA aktuell hauptsächlich die Anforderung der Integration heterogener Systeme erfüllt. Für die Umsetzung neuer Geschäftsprozesse spielt vor allem die Möglichkeit der sogenannten Orchestration von bereits bestehenden Services eine entscheidende Rolle.

Eine Möglichkeit (neben vielen weiteren) zur Realisierung einer SOA sind Webservices. Diese sind Dank verschiedener vorhandener funktionsfähigen Bibliotheken und der Standardisierung mittlerweile sehr weit verbreitet. Webservices stellen einen definierten Funktionsumfang über eine Remoteschnittstelle zur Verfügung. Als Protokoll wird SOAP ein http basiertes XML-Protokoll verwendet. Insgesamt wird an zahlreichen Stellen XML als Datenformat genutzt z.B. in der Webservice Description Language (WSDL). Ein Webservice kann interaktiv von einem Nutzer oder aber von anderen Webservices angesprochen werden. Dabei ist in den meisten Fällen eine Zugriffskontrolle notwendig. Ein etabliertes Verfahren zur Authentifikation stellen X.509 Zertifikate dar, welche über die vertrauenswürdige Instanz eines Trustcenters die Verknüpfung zwischen einem realen Subjekt und dem digitalen Nutzer herstellt. Über verschiedene Modelle (z.B. ein rollenbasiertes Zugriffsmodell) können dann die Zugriffsrechte der festgestellte Identität geprüft werden. Besondere Herausforderungen stellen die Möglichkeit der Delegation von Rechten in einer orchestrierten SOA zwischen den beteiligten Webservices dar.

1.3. Attributzertifikate als plattformübergreifende Tickets

Ziel dieser Arbeit ist die Untersuchung von X.509 Attributzertifikaten in der Anwendung als Zugriffsausweis (Ticket) in einer serviceorientierten Architektur zur Realisierung einer Zugriffskontrolle. In einem praktischen Beispiel wird eine Umsetzung erprobt und der Architekturaufbau der SOA und der komplette Lebenszyklus eines Zertifikats anhand eines konkreten Geschäftsprozesses beschrieben. Der Prototyp kann lediglich als Ideengeber für eine produktive Umsetzung genutzt werden. Es werden dort sehr vereinfachte Attribute und Sicherheitsziele umgesetzt, um die möglich Anwendung zu demonstrieren und zu testen. Als besondere Herausforderung im Vergleich zur „klassischen“ Zugriffskontrolle werden die Delegation und das Widerrufen von Rechten in verteilten Systemen sowie die Möglichkeiten einer anonymen Dienstnutzung untersucht.

2. Grundlagen

2.1. IT-Sicherheit

Wie z.B. in [13] beschrieben, kann man IT-Sicherheit aus zwei sehr unterschiedlichen Perspektiven betrachten. Zum einen gibt es die mathematische (in Teilen perfekte) Welt. Hier werden Algorithmen und Protokolle entworfen, deren Sicherheitseigenschaften anschließend bewiesen werden. Niemand kann daran im nachhinein etwas ändern (so lange der Beweis korrekt war). Die reale Welt mit existierenden IT Systemen und der Komplexität der unterschiedlichen Beteiligten und Schnittstellen ist aber weitaus schwieriger als „sicher“ zu gestalten. „Sicherheit ist

ein Prozess und kein Produkt“ sind die typischen Erkenntnisse aus den praktischen Erfahrungen der letzten 10 Jahre.

Auch die in dieser Arbeit beschriebenen Möglichkeiten sind in diesem Kontext zu sehen. Im Wesentlichen geht es um die Zugriffskontrolle auf Webservices, die wahrscheinlich mit den beschriebenen Verfahren erreicht werden kann, aber eben nur auf der technischen Ebene. An die umgebende Welt und alle beteiligten Subjekte, Systeme und Kommunikationen gibt es natürlich ebenfalls die Anforderung sicher zu sein. Das ist vor allem eine organisatorische Herausforderung und der hier beschriebene Teil nur ein kleiner Bausstein in der Sicherheitskette.

Zunächst werden die angestrebten Schutzziele und die Begrifflichkeiten erläutert. Anschließend werden theoretische Modelle zur Umsetzung einer Zugriffskontrolle und dann die praktischen Möglichkeit zur Realisierung des Zugriffsmatrixmodell vorgestellt.

2.1.1. Authentizität versus Autorisierung

Die Authentizität wird in [3] als erstes Schutzziel in einem sicheren System definiert. Dabei geht es um „die Echtheit und Glaubwürdigkeit des Objekts bzw. Subjekts, die anhand einer eindeutigen Identität und charakteristischen Eigenschaften überprüfbar ist“ [3, S. 6f]. Ein Subjekt ist dabei nicht immer direkt der aktive Benutzer, sondern umfasst ebenso Prozesse in dessen Auftrag. Diese können zum Beispiel auch Webserviceclients sein. Der Prozess des Authentizitätsbeweises ist die Authentifikation. Dieser ist wesentlicher Bestandteil und Grundvorraussetzung für viele digitalisierte und netzwerkbasierte Prozesse. Zur Authentifikation dienen heute verschiedene Verfahren, die man in drei Kategorien einordnen kann. Der Identitätsnachweis kann erfolgen über

- Besitz (z.B. klassisch Schlüssel zu einem Schloss),
- Wissen (z.B. Passwort),
- biometrische Merkmale (z.B. Fingerabdruck).

Neben der sehr weit verbreiteten Nutzername+Passwort Authentifikation haben sich X.509 Zertifikate [9] etabliert, die eine Kombination aus Besitz und Wissen darstellen. Der Besitz ist der private Schlüssel, das Wissen stellt das Passwort dar mit dem auf diesen Schlüssel zugegriffen werden kann. Die Zertifikate können je nach Umsetzung sehr hohe Sicherheitsstufen erreichen, in dem sie zum Beispiel auf entsprechenden Sicherheitstoken wie Smartcards oder USB-Token abgelegt werden. Dadurch ist ein Zugriff und eine Manipulation aus dem nutzenden System heraus nicht möglich. Neben diesen Eigenschaften haben X.509 Zertifikate durch die Einbettung in Public Key Infrastrukturen (PKI) weitere Vorteile. Durch hierarchische Organisation der Vertrauensstruktur und die Nutzung von Trustcentern ist es möglich auch für adhoc Kommunikation einen authentifizierten Kanal zu etablieren.

Bei der Autorisierung geht es um die Rechteverwaltung und Zugriffskontrolle eines IT Systems. Wichtige Vorraussetzung dafür stellt die vorausgehende erfolgreiche Authentifizierung aller Subjekte und Objekte dar. Erst danach können die entsprechenden Zugriffe kontrolliert werden. Es geht darum zu prüfen ob ein Subjekt S auf einem Objekt O eine Aktion A ausführen darf. Die Entscheidung eines derartigen Zugriffs (S, O, A) kann in verschiedene Modellen auf

unterschiedliche Art und Weise getroffen werden. Diese werden im nachfolgenden Abschnitt 2.1.2 dargestellt. Für die Anwendung auf Webservices ist das Subjekt S entweder der Benutzer direkt oder aber ein anderer Webservice oder Prozess des Nutzers. Das Objekt O ist typischerweise ein Funktion des Webservice und A ein ausführen (*execute*) dieser Funktion.

2.1.2. Zugriffskontrollmodelle

Die heute etablierten Sicherheitsmodelle sind zum großen Teil schon Anfang der 70er Jahre entstanden und wurden anschließend nur in Details modifiziert. Ziel der meist im militärischen Umfeld entstandenen Modelle war eine möglichst formal beweisbare Sicherheit. Dabei kann man Zugriffskontrollmodelle und Informationsflussmodelle unterscheiden. Für die Umsetzung der Webserviceautorisierung sind erstere interessant, da sie Möglichkeiten definieren den Datenzugriff zu steuern.

Aus dieser Kategorie werden in [3, S. 105ff] 4 Modelle vorgestellt. Während das Chinese-Wall Modell und das Bell-LaPadula Modell in der Praxis eher ein Nischendasein führen, findet man sehr häufig Anwendungen der Zugriffsmatrix (bzw. Ausprägungen davon) sowie des rollenbasierten Modells. In heutigen Standardbetriebssystemen (Linux, Solaris, Windows) ist eine Kombination aus Rollenmodell und Zugriffsmatrix, in der Form von Access Control Lists (ACLs), anzutreffen.

Das Zugriffsmatrixmodell definiert mit einer Matrix M_t einen Schutzzustand zu einem Zeitpunkt t als $M_t : S_t \times O_t \longrightarrow 2^R$ mit den zugreifenden Subjekten S_t auf die Objekte O_t. Die Rechte R sind zum Beispiel in klassischen Unix Systemen „`read`“, „`write`“ und „`execute`“. In der Tabelle 1 ist eine Beispielmatrix dargestellt.

Tabelle 1: Beispiel einer Zugriffsmatrix

	~/.ssh	~/public_html	/sbin/reboot	Object 4
Owner 1	read, write, execute	read, write, execute	-	-
Subject 2	-	read, execute	-	execute
Subject 3	-	read, write, execute	read, execute	-

Die Zugriffsmatrizen können dynamisch sein, dass heisst es gibt mit entsprechenden Rechten ausgestattete Subjekte, die die Möglichkeit haben diese zu verändern. Vorteil dieses Modells ist der objektorientiere Ansatz mit der Möglichkeit sehr fein granular Rechte zu vergeben. Bei entsprechend großen Systemen ist die Verwaltung dann aber auch entsprechend umfangreich. Im rollenbasierten Modell werden die Rechte daher nicht direkt an die Subjekte gebunden, sondern an Rollen, die dann wiederum Subjekten zugeordnet werden. Ein Subjekt kann zwar mehrere Rollen annehmen, in der Regel gibt es aus dieser Funktionssicht aber weniger Rollen als Subjekte und damit eine Vereinfachung.

2.1.3. Ticketbasierte Verfahren

Eine Möglichkeit der Umsetzung des Zugriffsmatrixmodell stellen Zugriffsausweise (andere Bezeichnungen sind auch Tickets oder Capabilities) dar. Das Subjekt S hat die Rechte R auf einem

Objekt O, wenn S Besitzer eines Tickets T_{OR} ist, in dem R für O enthalten sind. Die Matrix wird also zeilenweise aufgeteilt und die entsprechende Rechtebeziehung in ein Ticket verpackt. Genau wie bei der Matrix muss gewährleistet sein, dass das Ticket unverfälschbar ist. Tickets haben den Vorteil, dass nach erfolgter Ausstellung eine verteilte dezentrale Nutzung möglich ist. Dadurch steigt natürlich auch die Gefahr eines Diebstahls. Deshalb sind Eigenschaften notwendig wie eine begrenzte Gültigkeit und die Möglichkeit des Widerrufens eines gültigen nicht abgelaufenen Tickets. Attributzertifikate erfüllen diese Eigenschaften.

Ein bekanntes Verfahren ist Kerberos, das in der aktuellen Version 5 im [10] beschrieben wird. Die erste Version wurde bereits im Jahre 1987 von S. Miller und C. Neuman am M.I.T. entwickelt. Wesentliche grundlegende Architekturideen werden in dieser Arbeit aufgegriffen, so dass Kerberos hier kurz vorgestellt wird. Das Ziel des Verfahrens ist eine sichere Authentifizierung in einem TCP Netzwerk. Hierbei spielen im wesentlichen drei Komponenten zusammen. Zum einen das Key Distribution Center (KDC) mit Authentifizierungskomponenten und Ticketausstellung, desweiteren jeweils eine Kerberos Komponente auf dem Serviceclient C_K und eine auf dem Serviceserver C_S. Bei einem Authentifizierungsvorgang fragt C_K ein Ticket Granting Ticket (TGT) an. Dafür passiert über bekannte Verfahren zunächst eine Authentifizierung mittels Passwort oder Smartcard. Mit dem TGT kann dann für den aktuellen Zugriff des Clients auf den Server ein Session Ticket anfragen (ST). Dieses wird dann zur Authentifizierung an den C_S übergeben. Es reicht an dieser Stelle eine Ticketübergabe aus. Der C_S prüft das ST dann wiederum am KDC auf Gültigkeit. Alle beteiligten Kerberos-Komponenten sprechen dabei das spezielle Kerberosprotokoll. Vorteil dieses Verfahrens ist die Möglichkeit des TGT Caching und damit einer Wiederverwendung der erfolgreichen Authentifizierung für verschiedene Services und mehrere Sessions.

Warum also keine Webservice-Authentifizierung mit dem etablierten Kerberos Verfahren? Kerberos ist eine komplett eigene Welt mit neuen Datenformaten und Protokollen. Es ist erforderlich, dass alle beteiligten Komponenten Kerberos unterstützen. Für die aktuell gängigen Bibliotheken zur Webservice-Authentifizierung gilt dies nicht immer. Die Komplexität steigt durch Kerberos enorm an und dies soll der Versuch sein, einen einfacheren Weg zu finden. X.509 Zertifikate lassen sich in verschiedenen Welten direkt weiter- und wiederverwenden. So werden mit diesen aktuell digitale Signaturen, S/MIME basierte Emailverschlüsselungen und andere Anwendungen ermöglicht. Die Authentifizierung mit X.509 Attributzertifikaten soll sich in diese Welt einfügen und stellt damit eine direkten Gegenentwurf zu Kerberos dar.

2.2. X.509 Attributzertifikate

Schon in „klassischen“ X.509 Zertifikaten ist es möglich in sogenannten non-critical extensions beliebige Zusatzinformationen zum Inhaber zu speichern. Es hat sich aber erwiesen, dass ein separierter Datencontainer, der dann wiederum eindeutig mit dem Subjekt also dem Inhaber des dazugehörigen X.509 Zertifikats verbunden ist, verschiedene Vorteile hat. Dieser Container hat praktischerweise das gleiche grundlegende X.509 Format und wird über ein X.509 Profil definiert [siehe 4]. Ein wesentlicher Vorteil des getrennten Ablegens dieser zusätzlichen Eigenschaften (Attribute) besteht in der Möglichkeit, diese einzeln mit einer Gültigkeitsdauer zu versehen und widerrufen zu können ohne die komplette Identität des Subjekts zu widerrufen. Am Beispiel der Campuskarte der Technischen Universität Berlin [14] wird deutlich, wie wichtig

diese Eigenschaft ist. Dort sind zum Beispiel Attribute wie die Zugangserlaubnis zum Gebäude und auch der Bibliotheksausweis umgesetzt. Eine Sperre des Bibliothekszertifikats aufgrund verspäteter Bücherrückgabe hat nun nicht zur Folge, dass der Student das Gebäude nicht mehr betreten darf. Typischerweise ist die Lebenszeit einer ausgewiesenen Identität sehr viel höher als ein detailliertes Recht zu einer bestimmten Aktion.

Während X.509 Zertifikate als Bestandteil einer Public Key Infrastruktur dazu dienen ein Subjekt zu authentifizieren, leisten die Attributzertifikate darüber hinaus die Rechteverknüpfung um eine Autorisierung konkreter Aktionen zu ermöglichen. Aus der PKI leitet sich dabei wie z.B. in [1] beschrieben die vergleichbar aufgebaute Privilegien Management Infrastruktur (PMI) ab.

Tabelle 2: Gegenüberstellung PKI und PMI

	PKI	PMI
Zertifikat	Public Key Zertifikat	Attribut Zertifikat
Herausgeber	Certification Authority (CA)	Attribute Authority (AA)
Inhaber	Subjekt	Halter
Bindung	Name des Subjekts an öffentlichen Schlüssel	Name des Halters an Privileg(ien)
Widerrufen	Certificate Revocation List (CRL)	Attribute CRL (ACRL)
Vertrauenswurzel	Wurzelzertifikat	Autoritätsquelle
Ausstellende Unterinstanz	Sub CA	Attribut Autorität (AA)

Man findet wie in der PKI eine streng hierarchische Struktur mit einer Vertrauenswurzel, die im allgemeinen von einer dritten Instanz verwaltet wird, der beide (oder alle) beteiligten Komponenten vertrauen. Diese Autorität wird in PKI Umgebungen von Sicherheitsabteilungen, externen Trustcentern oder aber auch von staatlichen Institutionen wahr genommen. Da die Verwaltung einer PMI sehr viel spezifischer ist, wird sich in diesem Bereich wohl schwer eine vergleichbar übergeordnete Struktur etablieren können. Die Rechte in einer PMI sind auf bestimmte Prozesse und deren Sicherheitsdomäne beschränkt, so dass sich wohl höchstens branchenweite Attributtrustcenter entwickeln werden. Die bestehenden Datenformate, Protokolle und Verfahren aus der PKI Welt können für eine PMI genutzt werden. Das macht die Anwendung so attraktiv und ist gegenüber anderen Ticketverfahren (wie z.B. Kerberos) ein großer Vorteil in Umgebungen, wo PKI und PMI gemeinsam genutzt werden. Da Public-Key-Verfahren heute sehr weit verbreitet angewandt werden (z.B. als TLS Client/Server Authentifizierung oder in S/MIME) ist es eine interessante Herausforderung Attributzertifikate zur Rechteverwaltung in die bestehenden Systeme zu integrieren. Eine sehr gute Einführung zu Attributzertifikaten mit Ableitung und Vergleich zu PKI im PKIX IETF Standard und SPKI ist in [6] zu finden.

Attributzertifikate sind in ihrer Spezifikation völlig offen gehalten, betreffend der abgespeicherten Attribute. Hier können beliebige Strings (z.B. Schlüssel/Wert Paare) an den Halter gebunden und damit alle heute bekannten Autorisierungsverfahren abgebildet werden. Die direkte Ablage einer Zugriffsmatrixzeile mit den Zuordnungen von Objekten und Rechten ist natürlich nur bis zu einer begrenzten Komplexität auch tatsächlich sinnvoll. Die Datenmenge würde die Auswertung und den Transport der Attribute sonst schnell ineffizient werden lassen. Bei sehr vielen zu verwaltenden Rechten bietet sich daher eher die Anwendung eines rollenbasierten Verfahrens an, so dass im Zertifikat nur die Rollenzugehörigkeit(en) abgelegt werden. Man verliert dabei auf dem auswertenden System etwas an Unabhängigkeit, weil die Auflösung der Rolle auf die Zugriffsrechte nur entweder über einen zusätzlichen entfernten Dienst oder über eine Replikation der Rollendefinitionen über alle Systeme der verwalteten Domäne möglich ist. Bei einer derartig zentralisierten Rollenverwaltung können die Rollenprivilegien geändert werden ohne kursierende Zertifikate anzupassen. Dies kann bei sehr vielen Rechtehaltern ein schwerwiegender Verwaltungsvorteil sein, widerspricht aber der hier angenommen Zielumgebung von offenen verteilten Systemen. Die Änderung der Zertfikatsattribute lässt sich über zurückziehen (revoke) oder zeitliches Ablaufen lassen und erneute Ausstellung bewerkstelligen. Dies kann eine höhere Latenzzeit bis zur Umsetzung der Rechteänderung zur Folge haben als ein zentraler Rollenservice. Je nach Anwendungsfall muss dies beim Design berücksichtigt werden.

2.3. Webservices

2.3.1. Einführung

Webservices sind ein Mechanismus zur Nutzung verteilter Programmkomponenten und einer einheitlichen Art der Kommunikation zwischen diesen. Man kann sie als weitere Evolutionsstufe von Remote Procedure Calls für die es in verschiedenen Programmiersprachen unterschiedlich umgesetzte Ansätze gibt. Das Besondere an Webservices ist die Gestaltung als platformübergreifender und sprachunabhängiger Standard. Einen wesentlichen Beitrag dazu leisten die ausgiebig verwendete Datenbeschreibungssprache XML und die „Internetprotokolle“ HTTP und SMTP als Transportmedium.

Webservices sind keine an einem Ort zu einer Zeit gemachte Erfindung, sondern eine Entwicklung in vielen Bereichen. Es existieren dadurch unzählige Definitionen und sogar Diskussionen zur Übertragung der Schreibweise ins Deutsche. Sehr informativ dazu ist die Seite von Prof. Jeckle[1]. Eine dort zu findende Definition ist:

„Webservices sind selbstbeschreibende, gekapselte Software-Komponenten, die eine Schnittstelle anbieten, über die ihre Funktionen entfernt aufgerufen, und die lose durch den Austausch von Nachrichten miteinander gekoppelt werden können. Zur Erreichung universeller Interoperabilität werden für die Kommunikation die herkömmlichen Kanäle des Internets verwendet. Webservices basieren auf den drei Standards WSDL, SOAP und UDDI: Mit WSDL wird die Schnittstelle eines Webservices spezifiziert, via SOAP werden Prozedurfernaufrufe übermittelt und mit UDDI, einem zentralen Verzeichnisdienst für angebotene Web Services, können andere Webservices aufgefunden werden.“

[1] `http://www.jeckle.de/webServices/`

Die am weitesten verbreiteten Implementierungen sind .NET von Microsoft und AXIS von der Apache Foundation. Mittlerweile werden Webservices auch direkt im SUN Java SDK 1.6 unterstützt. Da Webservices an sich keinerlei Sicherheitsmechanismen definieren, wurde dafür der separate WS-Security Standard definiert.

2.3.2. OASIS WS-Security, der Standard

Sicherheit ist eine grundlegende Voraussetzung, um Webservices als vollwertigen Ersatz für bestehende Anwendungsarchitekturen einzusetzen. Die Integrität und Vertraulichkeit von Nachrichten muss sichergestellt sein. Zudem muss die Identität der beteiligten Parteien geprüft werden können. Im Standard zur Beschreibung des Kommunikationsprotokolls SOAP sind dafür keinerlei Mechanismen vorgesehen. Deshalb ist die Ergänzung Web Service Security Spezifikation die Basis für eine Realisierung von sicheren Lösungen. Die Organization for the Advancement of Structured Information Standards (OASIS) hat die 1. Version der WS-Security Spezifikation am März 2004 als Standard unter dem Namen OASIS Web Service Security (WSS) bestätigt. Mittlerweile ist die Nachfolgeversion [7] erschienen.

Es werden im wesentlichen drei grundlegende Verfahren definiert: die Signatur von SOAP-Protokoll Teilen zur Wahrung der Integrität, die Verschlüsselung von SOAP-Teilen zur Gewährleistung von Vertraulichkeit und die den Transport bzw. die Einbettung von Sicherheitstoken in das SOAP Protokoll zur Authentifizierung. Für diese Arbeit ist die Tokenübergabe entscheidend, wie in [8] beschrieben.

Einordnung ISO/OSI Sicherheitsprotokolle Für die Kommunikationssicherheit in Webservice Architekturen existieren bereits verschiedene Mechanismen in den tiefer liegenden OSI Modell Schichten. WSS definiert Umsetzungsmöglichkeiten innerhalb der Anwendungsschicht und kann daher speziell an die Bedürfnisse der realisierten Applikation angepasst werden. Die weiteren Mechanismen können transparent genutzt werden und erhöhen gegebenenfalls das Sicherheitslevel zusätzlich. In der Tabelle 3 werden die Protokollebenen dargestellt.

Tabelle 3: ISO/OSI Einordnung der Sicherheitsprotokolle

Schicht	Sicherheitsprotokoll	
6, 7 Anwendung	WSS, SMIME, HTTPS	Benutzerauthentifizierung, Autorisierung
4, 5 Transport u. Session	TLS	Verschlüsselung, Systemauthentifizierung
2, 3 Netzwerk	VPN	Kapselung der Routinginformation
1 Physikalisch	geschlossenes Netz	für Webservice selten anwendbar

SOAP Signatur Die Integrität der SOAP Nachrichten wird mit eingebetteten XML-Signaturen nach [12] gewährleistet. Veränderungen der SOAP-Daten können beim Empfänger bei der Signaturvalidierung erkannt werden. In der Spezifikation werden Mehrfachsignaturen und verschiedene erweiterbare Signaturformate vorgesehen.

Der Inhalt der SOAP-Nachricht wird mit einem Schlüssel bestätigt, dies passiert z.B. über das weit verbreitete Secure-Hash-Verfahren mit anschließender RSA-Verschlüsselung des Hashwerts. Der Empfänger hat anhand dieser Information bereits die Möglichkeit, die mathematische Inhaltskontrolle durchzuführen. Ob der Schlüssel wiederum zum Absender passt, kann in Kombination mit einem angegebenen Sicherheitstoken geprüft werden. Beim RSA-Verfahren könnte dies ein X.509 Zertifikat sein. Durch eine Zertifikatsvalidierung kann die Authentizität des Absenders geprüft werden.

Das Beispiel für eine SOAP Nachrichtensignatur aus [7, Seite 44]:

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..."
              xmlns:wsu="..." xmlns:ds="...">
  <S11:Header>
    <wsse:Security>
      <wsse:BinarySecurityToken ValueType="...#X509v3"
            EncodingType="...#Base64Binary" wsu:Id="X509Token">
        MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod
          Algorithm="http://.../2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#myBody">
            <ds:Transforms>
              <ds:Transform
              Algorithm="http://.../2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
            Algorithm="http://.../2000/09/xmldsig#sha1"/>
            <ds:DigestValue>EULddytSo1...</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          BL8jdfToEb11/vXcMZNNjPOV...
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#X509Token"/>
```

```
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
      </wsse:Security>
    </S11:Header>
    <S11:Body wsu:Id="myBody">
      <tru:StockSymbol xmlns:tru="...">
        QQQ
      </tru:StockSymbol>
    </S11:Body>
</S11:Envelope>
```

SOAP Verschlüsselung Die Vertraulichkeit der SOAP Nachrichten kann mit einer XML Verschlüsselung nach [2] gewährleistet werden. Auch hier sind Mehrfachverschlüsselungen verschiedener Beteiligter und die Erweiterung durch neue Verfahren vorgesehen. Es können beliebige Teile der Nachrichten Nutzdaten und/oder Kopfdaten mit einem symmetrischen Schlüssel verschlüsselt werden. Dieser Schlüssel kann z.B. als Passwort Sender und Empfänger bereits bekannt sein und mit einem UserNameToken dem richtigen Empfänger zugeordnet oder aber auch in der Nachricht verschlüsselt mit übermittelt werden. Letztere Variante wird auch als hybrides Verfahren bezeichnet und kann mit einem Public/Private-Schlüsselpaar in einem X.509 Token kombiniert werden zur einfachen Adressierung an den korrekten Empfänger der Nachricht.

Das Beispiel aus [7, Seite 46f] sieht so aus:

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
              xmlns:ds="..." xmlns:xenc="...">
    <S11:Header>
        <wsse:Security>
            <xenc:EncryptedKey>
                ...
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference>
                        <ds:X509IssuerSerial>
                            <ds:X509IssuerName>
                                DC=ACMECorp, DC=com
                            </ds:X509IssuerName>
                            <ds:X509SerialNumber>
                                12345678
                            </ds:X509SerialNumber>
                        </ds:X509IssuerSerial>
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
                ...
            </xenc:EncryptedKey>
        ...
```

```
        </wsse:Security>
    </S11:Header>
    <S11:Body>
        <xenc:EncryptedData Id="bodyID">
            <xenc:CipherData>
              <xenc:CipherValue>...</xenc:CipherValue>
            </xenc:CipherData>
        </xenc:EncryptedData>
    </S11:Body>
</S11:Envelope>
```

Authentifizierung mit Sicherheitstoken In das `<wsse:Security>` Element können verschiedene Sicherheitstoken eingefügt werden mit dessen Hilfe Signaturen und Schlüssel einer Identität zugeordnet werden können. Die einfachste Form ist ein UserNameToken.

```
<wsse:UsernameToken wsu:Id="...">
    <wsse:Username>anybodys name</wsse:Username>
</wsse:UsernameToken>
```

Für kompliziertere Token gibt es einen generischen Binärtyp, in dem dann der Token-Typ sowie das Kodierungsverfahren spezifiziert wird. Die beteiligten Webservicekomponenten müssen diese dann alle unterstützen. Momentan definiert ist hier zum Beispiel der Typ X509v3, der ein X.509 Zertifikat in Base64 kodierter Form enthalten kann.

```
<wsse:BinarySecurityToken wsu:Id=...
                          EncodingType='#Base64Binary'
                          ValueType='X509v3'/>
```

2.3.3. Anforderungen an eine Webservice-Sicherheitsarchitektur

Für die Umsetzung von Softwareprojekten ist es meist erforderlich, eine komplette Webserviceinfrastruktur aus verschiedenen Komponenten aufzubauen. Die Sicherheitsanforderungen gehen dabei über die Protokoll- und Nachrichtensicherheit weit hinaus. In [5] werden Anforderungen an eine Webservice-Sicherheitsarchitektur dargestellt.

Kurz zusammengefasst sind das die folgenden 11 Ziele:

Zugriffskontrolle die einzelnen Ausführungsrechte entsprechend des Zugriffsmatrixmodels,

Verteilte Architektur um die Beschränkung auf kontrollierbare Sicherheitsdomänen zu ermöglichen,

Flexible Identitätspreisgabe um nur die notwendigen Rückschlüssen auf den Nutzer zu gestatten,

Automatische Policy Ermittlung damit ein Client notwendige Rechte erfragen kann,

Automatische Accounts sollen für bestimmte Geschäftsprozesse möglich sein,

SingleSignOn an orchestrierten Webservices zur komfortablen Nutzung,

Feinspezifikation von Rechten bis hinunter auf erlaubte Wertebereiche für Einzelmethoden,

Erweiterbarkeit als typische Anforderung an jede Softwarearchitektur,

Integrierbarkeit als typische Anforderung an jede Softwarearchitektur,

Usability als typische Anforderung an jede Softwarearchitektur,

Performance als typische Anforderung an jede Softwarearchitektur.

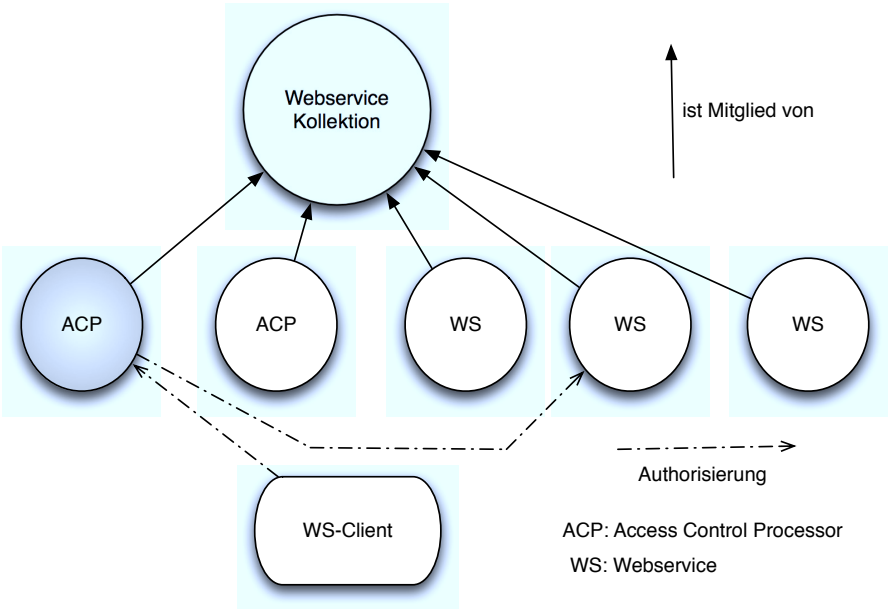
Aus diesen Anforderungen leitet Kraft eine Architektur ab, deren wesentlichen Komponenten die Access Control Processors (ACP) und der sogenannte Gatekeeper sind. ACP sind dabei selbst wiederum Webservices die Autorisierungsentscheidungen treffen. Es ist möglich jedem Webservices mehrere dieser ACP zuzuordnen, so dass je nach Komplexität und Performanceanforderung auch Einzelentscheidungen zu bestimmten Methoden oder Wertebereichen verteilt werden können. Am Ende trifft dann der Gatekeeper von dem es jeweils nur eine zugeordnete Instanz gibt, die finale Entscheidung und stellt entsprechend Tickets aus. Der Gatekeeper ist eine Art Sicherheitsproxy durch den die Anfragen getunnelt werden müssen, um ihre Autorisierung zu prüfen. Die Clients haben hier also keinen direkten Kontakt zum eigentlichen Webservice sondern sprechen immer direkt mit dem Gatekeeper. Das hat natürlich entsprechende Konsequenzen, beim Aufbau der Architektur. Die Gatekeeper müssen in der Lage sein TLS-Sessions zu terminieren und müssen mindestens die gleiche Performance und Verfügbarkeit, wie der eigentliche Service bieten, um die ursprüngliche Architektur nicht zu verschlechtern. Eine erhöhte Latenzzeit lässt sich durch die zusätzlichen Prüfungen natürlich nicht vermeiden.

2.4. Zusammenführung der Komponenten

Ziel dieser Arbeit ist die Kombination der dargestellten Komponenten zur Realisierung eines autorisierten Webservicezugriffs, sowie die Umsetzung einer anonymen Nutzung eines Dienstes. Die X.509 Attributzertifikate werden mit ihrer grundlegenden Eigenschaft Rechte zu speichern und überprüfbar an einen X.509 Zertifikatsinhaber zu binden als Ticket benutzt. Eine Attributauthorität (AA) legt die Rechtedefinition im Ticket entsprechend der für die Anwendungsdomäne notwendigen Zugriffsbeschränkungen an und kann über die Gültigkeitsdauer des Zertifikats auch die Rechte für einen eingeschränkten Zeitraum vergeben. Ausserdem können die Rechte zurückgezogen werden, in dem das Zertifikat widerrufen (revoked) wird.

Die Architektur von [5] wird vereinfacht und die Autorisierungsprüfung nicht von einen separaten ACP durchgeführt, sondern direkt im Zielwebservice über einen Security Handler als ein Protokollschritt vorgeschaltet. Nicht autorisierte Zugriffe können dann dort sofort abgebrochen werden und die Attributzertifikate können der Anwendungslogik hochgereicht werden, um eine Weiterverwendung in nachgeschalteten Subwebservices zu ermöglichen. Eine Übersicht der Architektur bietet das nachfolgende Anwendungsbeispiel.

Abbildung 1: Webservice-Sicherheitsarchitektur von Reiner Kraft [5, Seite 43]



3. Anwendungsbeispiel: Direktdemokratie

3.1. Ziel

3.1.1. Story

Die Stadt Berlin möchte ihren Bewohnern ermöglichen auf die regionale Politik direkt Einfluss zu nehmen und gleichzeitig einen Anreiz schaffen in der Stadt möglichst viele Steuern zu zahlen. Dafür wird im Jahr 2012 ein Punktesystem eingeführt. Die Bürgerpunkte (BPs) und die Steuerpunkte (SPs). Jeder in Berlin wahlberechtigte Bürger erhält für die Dauer einer Wahlperiode (4 Jahre) Bürgerpunkte und für seine eingezahlten Steuern entsprechend ihrer Höhe Steuerpunkte. Die Idee der zukünftigen Landesregierung ist, dass politische Entscheidungen, die keine direkte Investition von Steuergeldern nach sich ziehen (z.B. ethische Grundgesetze) mittels Volksabstimmung mit BPs entschieden werden und sämtliche Entscheidungen zu Investitionen mit entsprechend erworbenen SPs. In der 1. Stufe wird es nur eine Art von SP geben, später ist jedoch geplant, diese nach Steuerarten zu unterscheiden und z.B. Investitionen in die Verkehrsinfrastruktur nur durch entsprechend vom Verkehr initiierte Steuerpunkte (KFZ-, Mineralölsteuer, Maut) zu entscheiden. Die Punkte werden als Gewicht auf die eigene Stimme angerechnet und können während ihrer Gültigkeitsdauer beliebig oft, aber pro Abstimmung nur einmal, eingesetzt werden.

Erfreulicherweise kann das Projekt auf die erfolgreich eingeführte einheitliche Steuernummer und die dazu gehörige elektronische Steuerkarte aufbauen. Die Steuerkarte enthält für jeden in Deutschland gemeldeten Einwohner ein eindeutig zuordbares elektronisches Zertifikat, in dem

die Steuerbehörde die Identität des Individuums fehlerfrei bestätigt.

Zur Einführung einer effizienten Direktdemokratie sollen folgende Kriterien erfüllt werden:

- vollständig elektronische Abwicklung der Abstimmungsverfahren unter Einsatz des bestehenden Systems zur Authentifikation mittels Steuerkarte,
- Zugang zu Abstimmungen sowohl virtuell über Webportale als auch persönlich an Terminals in Wahllokalen,
- frei steuerbarer Verfall der Gültigkeit der Punkte durch die ausgebende Behörde,
- sofortiger Rückzug des Stimmrechts (z.B. bei Wegzug aus Berlin).

Nach der technischen Umsetzung dieser Punkte unter Berücksichtigung aller Datenschutzaspekte wird die Stadt Berlin ein Portal aufbauen, in dem alle politischen Entscheidungen, Planungen öffentlicher Etats, soziale Stadtprojekte usw. übersichtlich und verständlich dargestellt werden. Auf dem Portal ist ein direkter Meinungsaustausch in Foren, Blogs und Chats möglich. Über ein Bewertungssystem ist es vor anstehenden Entscheidungen möglich, die Aufmerksamkeit darauf zu erhöhen. Die eigentlichen Abstimmungen finden einmal im Monat an dem sogenannten Volkswahlsonntag statt. Von 0 bis 20 Uhr kann hier abgestimmt werden. Die Ergebnisse werden dann ab 20 Uhr veröffentlicht und fliessen direkt in die Gesetze zur Umsetzung ein.

3.1.2. Technik

Die Beispiele sind entstanden in der Programmiersprache Java[2] Version 1.6 (build 1.6.0-dp-b88-34) unter Nutzung Quellcodeprojekttools maven[3]. Als Teil des Paketes PMI wird die Erstellung und Prüfung von Attributzertifikate gezeigt. Das Paket WS als Wahl-Webservice enthält einen übersichtlich gehaltenen Prototypen aus Webservice und dazugehörigem Client. Besonderer Fokus liegt auf der Integration der Attributprüfung in das SOAP Protokoll bei Aufruf der Webservicemethoden. Die Attributzertifikate werden vom Client bei jedem Aufruf mitgeliefert („Pushverfahren“).

3.1.3. Einschränkung

Auch wenn hier als exemplarische Anwendung ein demokratisches Abstimmungsverfahren dargestellt wird, entspricht der Vorschlag keiner deutschen Wahlordnung. Verschiedene Aspekte wie z.B. die organisatorische Umsetzung, die Anonymität der Stimmabgabe oder die Manipulationssicherheit werden hier nicht betrachtet. Zur Sicherstellung der Einmalabgabe von Stimmen müsste weitere Geschäftslogik im Webservice hinzugefügt werden, die transaktionssicher bereits abgegebene Stimmen inkl. Zeitstempel wahlautomatenübergreifend registriert und bei jeder Stimmenabgabe abgleicht. Alternativ könnten auch Zertifikate eingeführt werden, die pro Abstimmung nur einmal gültig sind. Hierbei müsste sofort während der Nutzung der Stimme diese über einen Zertifikatswiderruf ungültig gemacht werden. Die Echtzeitverteilung dieser Widerrufsdaten stellte, dann eine weitere Herausforderung dar.

[2] `http://java.sun.com/`
[3] `http://maven.apache.org/`

3.2. Architektur

Wie zu erwarten, werden die SP's als Attributzertifikate und die Hauptapplikation zur Durchführung von Abstimmungen als Webservice umgesetzt. Die existierende Steuerkarten PKI wird um eine PMI erweitert in der zwei AAs berechtigt werden, Attributzertifikate auszustellen. Einmal das Einwohnermeldeamt (EMA) und die Oberfinanzdirektion Berlin (OFD). Zur Vereinfachung wird das EMA die vorhandene Infrastruktur der Steuerbehörde mitnutzen und über eine gemeinsame PMI Schnittstelle arbeiten, so dass die beiden lediglich über ihre Zertifikate unterschieden werden.

3.2.1. Beteiligte

Wähler: Personen, die autorisiert sind eine Stimme abzugeben

Ämter: Stellen, die Autorisierungen ausstellen und widerrufen

Treuhänder: Personen oder Institutionen, die berechtigt sind für andere Stimmen abzugeben

Wahlamt: Institution, die Abstimmungen erstellt und startet, die Stimmen zählt und Ergebnisse veröffentlicht

3.2.2. Anwendungsfälle

Aus den Anforderungen in 3.1 werden die nachfolgenden Anwendungsfälle abgeleitet. Diese stellen lediglich eine Auswahl dar, um die grundlegenden Prinzipien zu verdeutlichen. Für die tatsächliche Etablierung der Direktdemokratie gibt es sicher wesentlich mehr und auch komplexere Abläufe zu meistern. Der Fokus liegt auf den Autorisierungsvorgängen. Abbildung 2 stellt die Beziehungen der Beteiligten innerhalb der Anwendung dar.

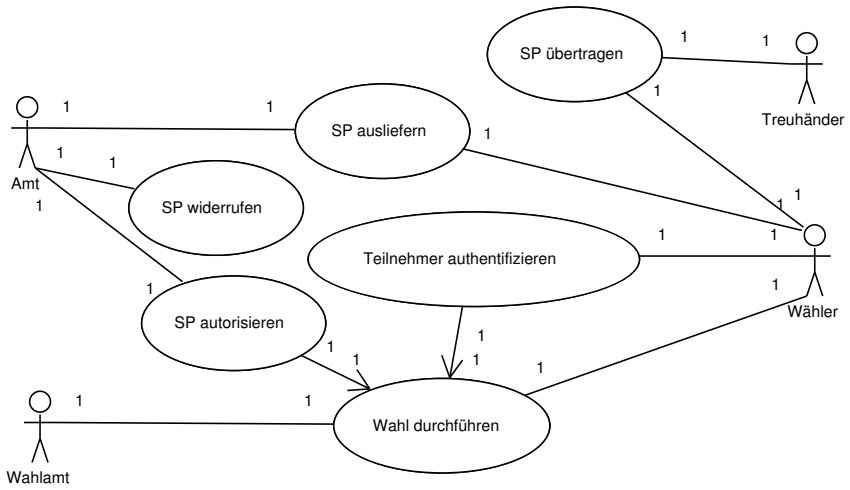


Abbildung 2: Betrachtete Anwendungsfälle in der Direktdemokratie

Anwendungsfall 1: Zuteilung der SPs an den Wähler Nach Eingang der Jahressteuererklärung wird an den Wähler ein Ticket ausgehändigt auf dem die Steuerpunkte für das laufende Jahr verzeichnet sind. Der Wähler wird hierbei von der OFD als Herausgeber über ein gültiges Attributzertifikat autorisiert an Wahlen teilzunehmen.

Anwendungsfall 2: Direkter Einsatz der SPs bei einer Onlineabstimmung zum Wiederaufbau des Berliner Stadtschlosses Auf dem Demokratieportal ist eine Abstimmung zum Wiederaufbau des Berliner Stadtschlosses gestartet. Der Wähler setzt seine SP ein, um dafür zu stimmen. Dabei wird der Wahlclient die „votePro“ Methode des Wahlautomat Webservice aufrufen. Wenn das dabei übermittelte AC gültig ist, wird die Stimme gezählt und entsprechend der Anzahl angefragter SP gewichtet. An der Befragung dürfen nur Personen, die das 18. Lebensjahr vollendet haben, teilnehmen.

Anwendungsfall 3: Ein mittlerweile nach Stuttgart verzogener Wähler wird bei der Abstimmung über die Einführung der Umweltzone in Berlin abgelehnt Für die Einrichtung der Umweltzone in Berlin können Wähler abstimmen. Ein vor kurzem umgezogener ehemaliger Bürger versucht unberechtigter Weise dagegen zu stimmen. Seine Stimme wird abgelehnt. Der Wahlclient ruft die „voteContra“ Methode des Wahlautomat Webservice auf. Die Validierung des übermittelten ACs ergibt, dass dieses widerrufen wurde. Die Stimme des Wählers wird verworfen.

3.2.3. Komponenten

Das Direktdemokratiesystem besteht grob unterteilt aus drei Komponenten der Privilege Management Infrastructur (PMI), dem Wahlautomaten und dem Wahlklient wie in Abbildung 3 dargestellt.

PMI In der PMI werden die Attributzertifikate (AC) verwaltet. Im Wesentlichen sind dafür drei Komponenten notwendig. Die Registration Authority dient der Feststellung der Identität des Wählers sowie der Rechteverwaltung. Hier wird geprüft, welche Attribute ausgestellt werden. Diese werden in einem Zertifikat beglaubigt und an die ID (Steuerkarte) des Wählers geknüpft. Die ACs können offline oder per remote Schnittstelle übergeben werden. Als Protokoll kann hier das standardisierte XKMS oder ein speziell dafür abgestimmter Webservice dienen. Die Zertifikatsinformation und Widerrufslisten (ACRLs) werden in einem sicheren Repository abgelegt.

Der AA Manager dient als Verwaltungsmöglichkeit der ausgestellten Zertifikate. Hier können bestehende ACs zurückgezogen werden. Die Validierung von ACs übernimmt die AC Validation Engine. Diese Funktionalität kann wiederum per XKMS oder Webservice remote zur Verfügung gestellt oder aber auch über eine lokale offline Schnittstelle durchgeführt werden.

Wahlautomat Der Wahlautomat ist das Herzstück einer Online-Direktdemokratie. Hier werden Abstimmungen verwaltet und durchgeführt. Wenn dem Wahlautomat über einen regelmäßigen Abgleich mit dem Repository der PMI alle Daten für eine Validierung zur Verfügung stehen,

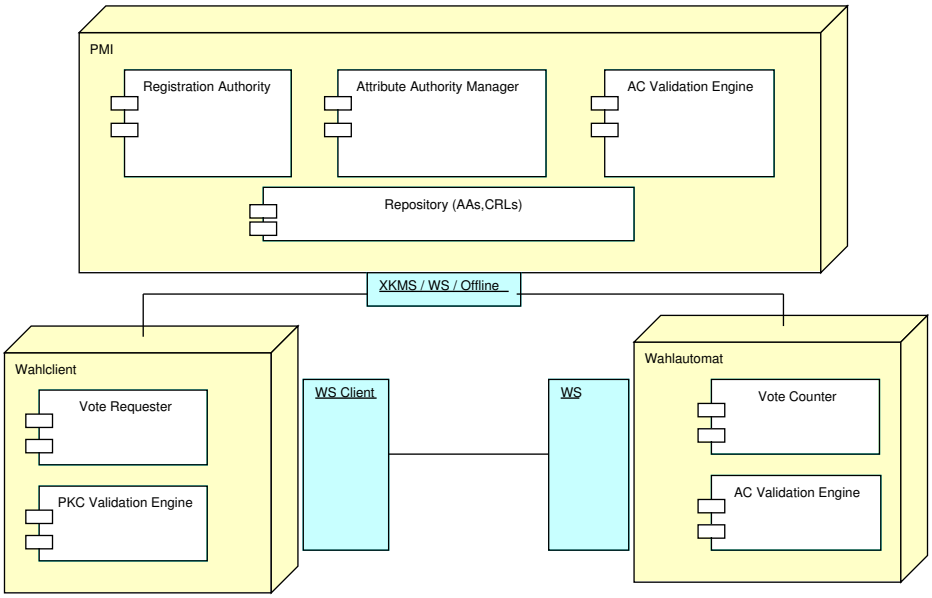


Abbildung 3: Komponenten im Direktdemokratiesystem

kann diese lokal und offline mit Hilfe der AC Validation Engine durchgeführt werden. Die Validierung kann aber auch in Echtzeit zur PMI weitergereicht werden. Der Stimmenzähler nimmt über den Wahlwebservice gültige Stimmen entgegen und wertet diese aus.

Wahlclient Der Wahlclient kann eine Software auf dem Computer des Wählers oder aber auch ein spezielles Wahlterminal sein. Über klassische Verfahren wird bereits hier die Identität des Wählers mit der PKC Validation Engine überprüft. Wenn diese einwandfrei festgestellt ist, führt der Vote Requester die Stimmenabgabe am Wahlautomat durch. Das AC mit den SPs und BPs wird dabei mit übertragen.

3.2.4. Schnittstellen

Wie mit den Komponenten bereits dargestellt, gibt es Schnittstellen zur PMI und zwischen Wahlclient und Wahlautomat. Für diese Arbeit werden die remote Schnittstellen zur PMI nicht weiter verfeinert. Zur Vereinfachung wird hier mit einer lokalen Datenübergabe auf Dateibasis gearbeitet. Hierbei ist auch eine Mehrfachabstimmung zur gleichen Wahl mit unterschiedlichen Stimmen möglich. Interessant und Thema dieser Arbeit ist der Webservice des Wahlautomaten an dem sich der Wahlclient mit einem AC für eine bestimmte Wahl autorisiert.

Wie in 2.3.2 dargestellt wird das AC als Binärstring in dem Webservice-Aufruf verpackt. Die Schnittstelle hat zwei Methoden: `pro()` und `contra()`. Beide Methoden bekommen als Parameter die ID der laufenden Abstimmung sowie die Anzahl der Punkte mit der die Stimme gewichtet werden soll. Diese Anzahl muss kleiner oder gleich der Punkteanzahl im AC sein, ansonsten wird die Stimme nicht gewertet.

3.3. Privilege Management Infrastructur (PMI)

Die Basisfunktionen einer PMI wurden in der Programmiersprache Java 1.6 und mit wesentlicher Beteiligung der Sicherheitsbibliothek Bouncycastle[4] erstellt. In den nachfolgenden Unterkapiteln werden die einzelnen Schritte erläutert. Der Quellcode, die Bibliotheken, die Testprogramme und die kurze Anwendungsdokumentation befinden sich auf der beiliegenden CD.

3.3.1. Vorbereitung

Die Kernaufgaben einer PMI sind die Herausgabe, Verwahrung und Überprüfung von Attributzertifikaten. Zur beispielhaften Umsetzung der drei beschriebenen Anwendungsfälle in dieser Arbeit wird ein AA Zertifikat, ein gültiges AC und ein zurückgezogenes AC erstellt. Die Verwaltung und Verwahrung wird mit einfachen Bordmitteln (java keystore, Dateisystem) und nur lokal umgesetzt. Natürlich macht es Sinn diese auch innerhalb einer SOA verteilt zur Verfügung zu stellen, was z.B. mit WS-Trust oder XKMS Architekturen möglich ist. Die Prüfeinheit ist eine Javaklasse, die als Bibliothek im Webservice angesprochen wird. Neben der Gültigkeitsprüfung kann diese auch genutzt werden, um die Attribute aus dem AC auszulesen.

3.3.2. Erstellung AA

Die Erstellung einer Attribute Authority gleicht dem Verfahren zur Generierung einer CA in einer PKI. Im Beispiel `AAissuer.java` wird ein neues Schlüsselpaar und daraus ein selbstsigniertes Wurzelzertifikat erzeugt. Diese kann zur Signatur des AC im nächsten Schritt verwendet werden. Das Zertifikat sieht folgendermaßen aus:

```
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number: 10 (0xa)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=DE, O=Humboldt University,
            OU=Systems Architecture Group, OU=Sample OFD
    Validity
        Not Before: Jan 20 23:56:01 2009 GMT
        Not After : Jul 20 23:56:01 2010 GMT
    Subject: C=DE, O=Humboldt University,
             OU=Systems Architecture Group, OU=Sample OFD
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
            00:f8:0d:a0:23:32:a6:16:60:92:ac:52:92:48:7a:
            3c:49:09:3e:7e:72:10:d3:c0:72:32:78:0d:7c:90:
            d9:3d:8f:42:df:2f:fb:43:63:7c:15:77:93:29:3e:
```

[4] http://www.bouncycastle.org

```
            95:74:97:5c:81:82:f1:e4:f1:23:db:eb:73:4c:9f:
            18:ca:17:3d:8e:04:98:98:b6:e7:c7:b0:5b:d4:0b:
            ef:1e:20:59:ff:ad:c2:10:e9:1d:68:3c:56:e1:4b:
            ea:cf:d7:61:59:48:c4:74:a7:0d:ca:f9:20:bd:82:
            08:2e:ae:55:1e:68:db:c8:6e:fc:5a:4b:d0:d5:3c:
            ce:4e:f8:66:c2:44:7d:ad:15
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:TRUE, pathlen:0
        X509v3 Key Usage: critical
            Certificate Sign, CRL Sign
Signature Algorithm: sha256WithRSAEncryption
    34:7d:61:a4:9c:44:1a:b4:ad:d3:3f:92:4c:48:9a:08:39:34:
    d3:eb:ec:a7:05:4f:17:e5:30:a5:c2:97:41:33:97:3c:37:6b:
    02:4e:89:71:6f:f5:93:0c:19:1b:d1:5d:34:62:85:52:b8:b8:
    9d:92:33:72:1a:bf:8c:39:33:01:cf:fe:11:38:00:fc:6e:bc:
    8b:1b:22:8b:45:2f:82:27:3d:47:ab:00:43:ce:00:c0:ac:50:
    ce:d7:4d:bd:82:29:ac:1d:15:41:84:53:18:87:c4:5f:b7:0b:
    d9:92:26:45:d3:bd:83:ea:91:24:90:55:45:93:6c:2f:40:79:
    f7:78
```

3.3.3. Erstellung AC

In der Klasse `ACissuer.java` wird ein Attributzertifikat erzeugt und der Anwendungsfall 1 umgesetzt. Als Voraussetzung wird ein Identitätszertifikat benötigt, dass in der Klasse `PKCissuer.java` erzeugt wird. Für dieses Beispiel wird die gleiche CA verwandt wie für das AC und mit dieser ein Identitätszertifikat für die Bürgerin *Identa* ausgestellt. Dieses PKC ist mit dem ausgestellten AC verknüpft. Hier zunächst das PKC Zertifikat:

```
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=DE, O=Humboldt University,
            OU=Systems Architecture Group, OU=Sample OFD
    Validity
        Not Before: Jan 27 23:13:28 2009 GMT
        Not After : Jul 27 23:13:28 2010 GMT
    Subject: CN=Identa, C=DE, O=Humboldt University,
             OU=Systems Architecture Group, OU=Sample OFD
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
```

```
        RSA Public Key: (2048 bit)
            Modulus (2048 bit):
                00:9e:96:fd:c1:88:3b:23:df:99:9c:1f:17:c9:d5:
                ae:e7:6b:ef:e0:f1:f7:c4:0d:91:1a:06:9d:b5:0a:
                11:20:c7:1a:ed:7a:74:1f:70:99:00:29:58:55:aa:
                b5:f7:f7:a7:79:c7:2b:15:c3:37:cc:2a:3d:59:64:
                5a:54:27:56:32:f1:f8:1b:06:a8:00:81:34:90:ef:
                a7:1e:c0:ca:1f:76:43:03:85:d4:67:d0:49:b5:32:
                71:44:a2:99:27:4e:3a:1e:80:57:a1:65:9f:a5:19:
                3b:d6:64:e1:d4:d3:0c:71:fd:1d:6f:45:f1:e5:87:
                7b:27:57:c0:4a:fd:77:b0:ea:28:33:fd:e0:8a:dd:
                ae:8d:5a:d4:21:da:a6:2e:39:ef:aa:6f:0c:13:f2:
                46:a3:bc:9b:72:b1:60:0e:00:0a:1c:51:7a:1b:8c:
                9d:c5:38:41:c2:8d:9b:46:f8:38:d9:02:81:0d:e8:
                7e:df:65:4b:62:8c:17:63:2f:99:ca:40:ae:ee:09:
                39:46:e4:22:97:bd:99:5e:4f:7a:61:eb:3c:a1:8a:
                68:0d:9a:58:07:b4:00:ca:bb:a5:21:ec:42:53:3f:
                d8:ad:2d:c6:76:9b:34:f9:ef:02:e7:cc:b9:a3:45:
                c6:4f:18:7f:b1:f7:12:43:ef:34:c9:c5:bb:4b:d3:
                af:e5
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Authority Key Identifier:
            keyid:38:4D:4D:62:62:77:12:93:A7:59:8E:9B:4D:
                  C4:A0:2E:EA:4F:0E:42
            DirName:/C=DE/O=Humboldt University/OU=Systems
                    Architecture Group/OU=Sample OFD
            serial:0A
        X509v3 Subject Key Identifier:
            FC:67:2C:E5:7C:8C:8A:4E:4C:D2:77:24:A4:9E:CB:
            7A:5B:21:B9:22
        X509v3 Basic Constraints: critical
            CA:FALSE
        X509v3 Key Usage: critical
            Digital Signature, Non Repudiation, Data Encipherment
Signature Algorithm: sha256WithRSAEncryption
    17:d5:e5:24:d2:2d:08:41:c4:cf:1c:24:fb:65:b0:46:e0:a3:
    3f:f6:2f:d9:17:b2:c1:8e:89:6a:a8:1f:ca:76:42:75:1f:8d:
    6a:45:43:69:62:74:87:ca:a9:14:93:53:cd:6f:ed:88:54:8d:
    3d:77:74:8a:a6:ef:29:b6:42:8d:fc:a5:7a:44:6a:ee:db:b0:
    57:66:0f:17:22:d8:a7:d2:25:11:c5:93:dd:c0:cd:ac:b7:54:
    ed:bb:e8:a8:b1:93:a4:64:87:2d:b0:b3:0d:3a:81:74:b0:5a:
    b8:fb:94:31:3b:47:49:cf:dc:37:ca:c5:28:e0:31:2d:bc:52:
    fa:60
```

Die Bürgerin *Identa* bekommt ein AC mit 10 eingetragenen Steuerpunkten. Ausserdem wird noch das Attribut P18 gesetzt, dass das überschreiten dieser Altersgrenze bestätigt. Das Zertifikat sieht folgendermassen aus (Die Darstellung unterscheidet sich jetzt etwas zu den obigen openssl Ausgaben, da openssl aktuell keine Attributzertifikate unterstützt.)

```
Certificate:
Data:
        Version: 1
    SerialNumber: 1001
        Issuer: OU=Sample OFD, OU=Systems Architecture Group,
                O=Humboldt University, C=DE
      Start Date: Mon Jan 28 02:32:28 CET 2009
      Final Date: Mon Dec 28 03:32:28 CEST 2009
Holder (issuer): OU=Sample OFD, OU=Systems Architecture Group,
                O=Humboldt University, C=DE
Holder (serial): 1
       Signature: bc929a9c0cb4080cd00c9cc5b76a1fa7db54bdfe
                c329de0fa141cb43477529654c38e2b6d2514462
                9c9a73ecd619fee7ed7567fcf7ad49959bcc113a
                c1d29321b6f100b2d8fae9ba1093dd940140d913
                4ea7cc8560806cc7ad7849a9fc2f1d80248b1caa
                cfaaaab91cbe357dcb68604b83d107ede5a93fec
                6e611c3e74ef4c09
     Attributes:
                P18 = true
                 SP = 10
```

3.3.4. Verifier

Die Klasse ACverifier stellt eine Methode zur Überprüfung des Attributzertifikats zur Verfügung. Bei einer solchen Validierung wird der Gültigkeitszeitraum, die Signatur der ausstellenden AA, der Zertifikatspfad bis zum Wurzel AA Zertifikat sowie die Revocation geprüft. Das Beispiel enthält zur Vereinfachung keine keystore oder LDAP Logik zur Suche der dazugehörigen Zertifikate. Diese werden aus bekannten statischen Pfaden ausgelesen. Normalerweise wäre es erforderlich anhand des Issuer-DN das jeweils signierende Zertifikat in einem öffentlichen Verzeichnis nachzuschlagen. Eine weitere Vereinfachung betrifft die Überprüfung des Revocationstatus. Zwei Verfahren sind hierfür möglich: OCSP und ACRLs. Beide sind aus der PKI bekannt und nicht im Fokus dieser Arbeit.

Die Verifizierungsmethode kann nun im Security Handler des Webservice genutzt werden zur Prüfung der angehängten „Tickets“. Als kleine zusätzliche Hilfsmethode dient `getAttribute( X509AttributeCertificate acCert, String attOID )`, um einzelne Attribute abzufragen.

3.4. Webservice

3.4.1. Java SDK integrierter Webservicestack: JAX-WS

Seit der Version 1.5 ist im Java SDK ein kompletter Webservicestack enthalten, der die Entwicklung von Webservice- Serverkomponenten und -clientkomponenten ermöglicht ohne auf externe Bibliotheken zurückgreifen zu müssen. Langfristig wird dadurch die weit verbreitete AXIS Bibliothek von der Apache Foundation vermutlich als „Bestpractise“ abgelöst. Diese Arbeit ist deshalb auch auf die SUN eigenen Standardkomponenten ausgelegt.

Es gibt zwei mögliche Entwicklungsverfahren zur Implementierung von Webservices: ausgehend von einem WSDL XML-Dokument, das den Service beschreibt, werden sowohl Server als auch Client generiert oder, basierend auf einer Implementierung der Serviceklassen, wird ein WSDL xml generiert und daraus dann die Clientseite. Beide Verfahren sind mit JAX-WS möglich und durch entsprechende Werkzeuge unterstützt, aber besonders der zweite Weg ist durch die Nutzung von Annotationen zu einer angenehmen, übersichtlichen und quellcodezentrierten Variante geworden. Da für den hier erstellten VoteService keine komplizierten eigenen Datentypen und keine speziellen Serialisierungen von Datenstrukturen notwendig sind, ist der quellcodebasierte Weg in der Umsetzung wesentlich einfacher zu handhaben. Die fortschreitenden Spezifizierungen im SOAP Protokoll Umfeld bleiben dem Anwender (Entwickler) ebenso verborgen, wie zusätzliche Komponenten und Namensräume (z.B. aus der WS-Addressing Spezifikation).

Die Beispielimplementierung zur Direktdemokratie orientiert sich an der Dokumentation über Webservicesicherheit[5] und den dort aufgeführten Beispielen. Das folgende kurze Beispiel zeigt die Nutzung der Annotationen zur vollständigen Beschreibung eines Webservice mit einer Methode.

```
@WebService(targetNamespace="http://server.ws.termpaper/")
public class VoteService {
  @Resource WebServiceContext context;

  @WebMethod
  public String votePro(@WebParam(name="voteId") String voteId,
                @WebParam(name="taxPoints") String taxPoints) {
    try {
      vote(authToken, voteId, "pro", taxPoints);
    } catch (Exception e) {
      return "failed: " + e.getMessage();
    }
    return "ok";
  }
}
```

Nach dem Start des Servers kann unter der Standard URL[6], das dazugehörige WSDL bezogen werden. Aus diesem WSDL Dokument lässt sich dann der Client entsprechend ableiten. Alle

[5] https://xwss.dev.java.net/Securing_JAVASE6_WebServices.html
[6] http://localhost:9999/SecureWS/VoteService?wsdl

benötigten Webservicekomponentenklassen werden durch die in maven integrierten Werkzeuge automatisch generiert.

```
public class VoteClient {
  @WebServiceRef(wsdlLocation=
    "http://localhost:9999/SecureWS/VoteService?wsdl")
  private static VoteServiceService service;

  public static void main(String[] args) throws Exception {
    service = new VoteServiceService();
    VoteService port = service.getVoteServicePort();
    result = port.votePro("voteId", "5"); // 'ok'
    result2 = port.getVoteResult("voteId");
    // 'VoteResult{voteId='voteId',proVotes=1,proTaxPoints=5}'
  }
}
```

3.4.2. Interoperabilität und Sicherheit auf Basis der JAX-WS API: das Metro Projekt

Auf der Suche nach einer Java Implementierung der Webservice Sicherheitsbibliotheken trifft man schnell auf das Metroprojekt[7] der Glassfish Gemeinde. Metro implementiert einen kompletten Webservice-Stack mit vielen begleitenden Komponenten und unterstützt die Webservice Interoperabilitätstechnologie (WSIT), eine Implementierung zur Gewährleistung der Zusammenarbeit von Java- und .NET-Implementierungen. Da darüber hinaus auch der Webservice Security Standard implementiert ist, stellt metro aktuell die beste Möglichkeit zur Umsetzung der Anwendung dar. Für die Nutzung sind nur wenige zusätzliche Abhängigkeiten notwendig.

```
<dependency>
        <groupId>com.sun.xml.ws</groupId>
        <artifactId>webservices</artifactId>
        <version>2.0-SNAPSHOT</version>
</dependency>
<dependency>
        <groupId>com.sun.xml.ws</groupId>
        <artifactId>webservices-rt</artifactId>
        <version>2.0-SNAPSHOT</version>
</dependency>
<dependency>
        <groupId>com.sun.tools.ws</groupId>
        <artifactId>webservices-tools</artifactId>
        <version>2.0-SNAPSHOT</version>
</dependency>
```

[7] https://metro.dev.java.net/

Die Javaimplementierung erfolgt wie bereits in 3.4.1 dargestellt mittels Annotationen. Für WSIT wird jeweils auf der Server- und Clientseite eine spezielle Konfiguration angelegt, in der der Webservice detaillierter beschrieben und mittels WS-Policy die Sicherheitsanforderungen inklusive der Übergabe und Validierung der Attributzertifikate definiert werden. Um eine Kollision mit den WSIT X509- Validierungsmethoden zu vermeiden, wird das AC im `UserNameToken` abgelegt. Eine Schicht tiefer bei der Erstellung der SOAP Nachricht, wird der Typ automatisch richtig erkannt und wie gewünscht als X509BinaryToken deklariert und transportiert. Das `UserNameToken` wird in der WSIT Richtlinie als `SignedSupportingToken` definiert und damit automatisch signiert. Jegliche Manipulation am Inhalt des SOAP Pakets führt dadurch bereits auf Protokollebene zu einem Fehler und dem Abbruch der Konversation.

Im Folgenden wird die entscheidende Integration der AC Übergabe in den Standardprogrammfluß skizziert. Auf der Clientseite wird ein spezieller CallbackHandler konfiguriert, in dessen Implementierung das AC gelesen und dann als Username übergeben wird.

```
wsit-termpaper.ws.client.VoteService.xml:
<wsp:Policy wsu:Id="VoteServicePortBindingPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
    "..."
    <sc:CallbackHandlerConfiguration wspp:visibility="private"
        useXWSSCallbacks="true">

      <sc:CallbackHandler name="usernameHandler"
      classname="termpaper.ws.client.UsernameCallbackHandler"/>

    </sc:CallbackHandlerConfiguration>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>"
```

Auf der Serverseite muss für die Gegenstelle ein eigener Validator konfiguriert werden. Dieser bekommt das Zertifikat des aktuellen Serviceaufrufs vom Protokollhandler übergeben und kann dieses dann validieren. Auf der Protokollebene wird zunächst die generelle Zertifikatsgültigkeit validiert. Dazu gehört die Gültigkeit der Ausstellerzertifikatskette, des Zeitraums und der Signatur sowie die Prüfung der Widerrufsliste. Die inhaltliche Auswertung der Attribute wird in diesem Beispiel komplett innerhalb der Geschäftslogik der Applikation durchgeführt.

```
wsit-termpaper.ws.server.VoteService.xml:
<wsp:Policy wsu:Id="VoteServicePortBindingPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
    <sc:ValidatorConfiguration wspp:visibility="private">

      <!-- use the attribute cert validator"
      <sc:Validator wspp:visibility="private"
      name="usernameValidator" classname="
```

```
        termpaper.ws.server.X509AttributeCertificateValidator"/>

      </sc:ValidatorConfiguration>
      "..."
      </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>"
```

Wenn ein Webserviceaufruf dann in der aufgerufenen Methode „ankommt“, ist bereits sichergestellt das ein gültiges AC geliefert wurde. Die Applikation kann dann je nach durchzuführender Aktion die Autorisierung anhand der Attribute vornehmen. Im Fall der Direktdemokratie werden z.B. die zur Abstimmung eingesetzten SPs verglichen mit den maximal verfügbaren SPs des Wählers, die im Attribut verzeichnet sind. Das sieht als Javacode dann so aus:

```
// check taxpoint attribute
taxPoints = Integer.parseInt(taxPointsStr);
String authorizedTaxpoints = CertTool.getAttribute(acCert,
                                    CertTool.OID_SP_ATTRIBUTE);
if (taxPoints > Integer.parseInt(authorizedTaxpoints)) {
  throw new VoteAuthorizationException("requested taxPoints "
    + taxPointsStr + " exceeds authorized " + authorizedTaxpoints);
}
```

Damit ist die Implementierung als prototypischer „Durchstich“ vollständig. Die Attributzertifikate werden als Ticket zum Transport von Autorisierungsinformationen erfolgreich genutzt.

3.5. Projektstruktur und Sourcecodemanagement (SCM)

Das Projekt nutzt die maven build und deployment Struktur mit Apache Maven 2.2.1[8] und Java version: 1.6.0_15. Die drei Subprojekte PMI, WS-Server und WS-Client sind jeweils als Projektmodell in einer `pom.xml` beschrieben. Alle benötigten Bibliotheken sind im `pom.xml` als Abhängigkeiten eingetragen und werden bei der Ausführung von `mvn` automatisch ins lokale maven-Paketdepot geladen. Für die erste Ausführung oder Kompilierung muss der Rechner also eine Internetverbindung haben, um die Abhängigkeiten zu laden.

Eine Besonderheit in der Struktur ist das META-INF Verzeichnis innerhalb der `resources`. Die `resources` werden für die Ausführung mit im CLASSPATH geladen. Durch das META-INF Verzeichnis findet WSIT seine Konfigurationsdateien und Zertifikatsspeicher auch ohne Webarchivpaketierung und Applikationsserverintegration und das Beispiel kann mit dem im JDK enthaltenen HTTP-Server getestet werden.

Als komplette Übersicht der Dateistruktur dient der Anhang A.

3.6. Ausführen und Testen der Beispielimplementierung

Zur Demonstration der Funktionalität der Beispielimplementierung ist diese auf die Ausführung als eigenständiger Javaprozess ausgelegt. Für die Übertragung in „echte“ Applikationen ist

[8] `http://maven.apache.org/`

ein Deployment innerhalb eines Applikationsservers wie zum Beispiel glassfish v3 ratsam. Ein einzelner Javadaemon ist administrativ schlechter wartbar und aus Sicht von Langzeitstabilität und Performance sicher suboptimal. Zum Testen der Webserviceinteraktion sind zwei Prozesse notwendig. Einmal die Serverseite und darauf zugreifend ein Client. Die jeweiligen Projektbeschreibungen `pom.xml` sind so erweitert, dass mit Hilfe von maven die Javaprozesse gestartet werden können. Für die oben beschriebenen Anwendungsfälle werden in den folgenden Abschnitten Teile des SOAP Protokolls dargestellt, die die Einbettung der Attributzertifikate und Validierungsergebnisse der Authentifizierungsinformationen zeigen.

Installation der PMI Komponente im lokalen maven repository:

```
cd trunk/PMI; mvn clean install
```

Start der Serverseite:

```
cd trunk/WS/server; mvn clean package exec:java
```

Start der Clientseite (nachdem der Server läuft):

```
cd trunk/WS/client; mvn clean compile exec:java
```

3.6.1. Anwendungsfall 2: Durchführung der vote Operationen mit gültigen Attributen

Im Anwendungsfall 2 wird mit gültigen Zertifikaten die erlaubte Operation `votePro()` aufgerufen. Das dabei übergebene Attributzertifikat wird statisch aus einer Datei ausgelesen. Zur Umsetzung einer „echten“ Anwendung wird das Zertifikat aus einem Speicher, z.B. einer Smartcard des Wählers stammen. Über den in Abschnitt 3.4.2 beschriebenen Callbackhandler wird das Zertifikat der Anfrage hinzugefügt. Der sich daraus ergebenen SOAP Teil sieht folgendermassen aus:

```
<S:Envelope>
<S:Header>
  <To>http://localhost:9999/SecureWS/VoteService</To>
  <Action>
    http://server.ws.termpaper/VoteService/voteProRequest
  </Action>
  <wsse:Security S:mustUnderstand="1">
    <wsu:Timestamp>
      <wsu:Created>2009-09-27T08:28:01Z</wsu:Created>
      <wsu:Expires>2009-09-27T08:33:01Z</wsu:Expires>
    </wsu:Timestamp>
    <wsse:BinarySecurityToken
        ValueType="http://docs.oasis-open.org/wss/2004/01/\
            oasis-200401-wss-x509-token-profile-1.0#X509v3"
        EncodingType=".../2004/01/oasis-200401-wss-soap-\
                    message-security-1.0#Base64Binary">
```

```
        MIIDDzCCAnigAwIBAgIBAjANBgkqhkiG9w0BAQQ
        ...
        6n9hC0abODh8cLUh7Q==
      </wsse:BinarySecurityToken>
      ...
      <xenc:EncryptedKey>...</xenc:EncryptedKey>
      <xenc:EncryptedData>...</xenc:EncryptedData>
      <ds:Signature>...</ds:Signature>
    </wsse:Security>
</S:Header>
```

Die auf diese Anfrage ablaufende serverseitige Validierung verläuft in allen Stufen erfolgreich und das Abstimmungsergebnis wird hinterlegt. Die Ergebnisabfrage liefert folgende Ausgabe:

```
call port.getVoteResult(stadtschloss) after result:
VoteResult{voteId='stadtschloss',
           proVotes=1, proTaxPoints=5,
           contraVotes=0, contraTaxPoints=0}
```

3.6.2. Anwendungsfall 3: Versuchte Durchführung einer vote Operation mit zurückgezogenem Attributzertifikat

Im Anwendungsfall 2 wird mit einem zurückgezogenem AC versucht an einer Abstimmung zur Berliner Umweltzone teilzunehmen. Die übermittelte SOAP Anfrage sieht von der Struktur her identisch aus zur Anfrage im vorhergehenden Abschnitt 3.6.1. Auf der Serverseite wird dann die Verifikationsfunktion durchlaufen, wobei bei der Prüfung der ACRL auffällt, dass das AC zurückgezogen wurde. Die Logmeldung dazu sieht folgendermassen aus:

```
termpaper.pmi.ACverifier - attribute certificate is invalid
java.security.cert.CertificateException: attribute certificate
revoked since Sun Sep 27 13:14:47 CEST 2009
```

Das AC wird also bereits während der Protokollabarbeitung abgelehnt und die Logik zur Stimmzählung nicht ausgeführt. Hier liegt eine erfolgreiche negative Autorisierung vor. Die anschließende Abfrage der aktuellen Abstimmungsergebnisse zeigt dann auch das gewünschte Resultat:

```
call port.getVoteResult(umweltzone) after result:
VoteResult{voteId='umweltzone',
           proVotes=0, proTaxPoints=0,
           contraVotes=0, contraTaxPoints=0}
```

4. Fazit

Die Nutzung von Attributzertifikaten zur Autorisierung lässt sich in das Webservice-Protokoll SOAP elegant integrieren. Hierbei können bereits vorhandene und standardisierte Protokollerweiterungen genutzt werden, die eine Interoperabilität auch in Zukunft sicher stellen. Die Wiederverwendung der Erkenntnisse aus mehreren Jahren PKI Praxis vereinfachen die Umsetzung des hier vorgeschlagenen Ansatzes zur Nutzung von Attributzertifikaten als Sicherheitstoken. Alle gängigen Prozesse, wie Ausstellung, Verifikation oder Widerrufen, lassen sich auf Attributzertifikate übertragen und es existieren dafür bereits Javaimplementierungen. Im Bereich Webservice sind die bestehenden Implementierungen im Metroprojekt bereits weit fortgeschritten, so dass mit der finalen Version 2.0 alle Vorraussetzungen vorliegen, um auch interoperabel und unter Nutzung der neuesten Webservice-Standards zu arbeiten und bei der Implementierung auf Basiskomponenten der SUN Java Gemeinde zurückzugreifen.

Auch wenn die technische Kombination der beiden Technologien erfolgreich gezeigt werden konnte, bleiben einige Herausforderungen bestehen. Die enge und pragmatische Verzahnung der Autorisierungsprüfung mit dem Kommunikationsprotokoll kann bei anderen Anforderungen und höherer Komplexität des Anwendungsbereiches auch zu einer architektonischen Schwäche werden. Die Modularisierung und Abgrenzung zwischen Protokollsicherheit und Anwendungssicherheit ist nicht eindeutig definiert. Zur Kapselung der Autorisierungsprüfung empfiehlt sich zur weiteren Betrachtung die Nutzung eines separaten Services, der dann sowohl zur Vorprüfung im Webserviceablauf als auch zur detaillierten Rechtebestimmung bei der Abarbeitung der Geschäftslogik dienen kann.

A. Verzeichnisstruktur der Beispielimplementierung

```
|____trunk
| |____PMI
| | |____pom.xml
| | |____src/main
| | | |____java
| | | | |____termpaper/pmi
| | | | | |____AAissuer.java
| | | | | |____ACissuer.java
| | | | | |____ACverifier.java
| | | | | |____CertTool.java
| | | | | |____PKCissuer.java
| | | |____resources
| | | | |____certs
| |____WS
| | |____pom.xml
| | |____client
| | | |____pom.xml
| | | |____src/main
| | | | |____java
| | | | | |____termpaper/ws
| | | | | | |____client
| | | | | | |____UsernameCallbackHandler.java
| | | | | | |____VoteClient.java
| | | | |____resources
| | | | | |____client-security-env.properties
| | | | | |____META-INF
| | | | | | |____client-*store.jks
| | | | | | |____wsit-client.xml
| | | | | | |____wsit-termpaper.ws.client.VoteService.xml
| | |____server
| | | |____pom.xml
| | | |____src/main
| | | | |____java
| | | | | |____termpaper/ws
| | | | | | |____server
| | | | | | |____VoteService.java
| | | | | | |____X509AttributeCertificateValidator.java
| | | | |____resources
| | | | | |____META-INF
| | | | | | |____server-*store.jks
| | | | | | |____wsit-termpaper.ws.server.VoteService.xml
```

Literatur

[1] CHADWICK, D. W., A. OTENKO und E. BALL: *Implementing Role Based Access Controls Using X.509 Attribute Certificates*. IEEE Internet Computing, 07(2):62–69, Mar/Apr 2003.

[2] EASTLAKE, D. und J. REAGLE: *XML Encryption Syntax and Processing*. W3C Recommendation, W3C, Dez. 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

[3] ECKERT, C.: *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. Oldenbourg Verlag, Muenchen Wien, Studienausg. Aufl., Jan 2005.

[4] FARRELL, S. und R. HOUSLEY: *An Internet Attribute Certificate Profile for Authorization*. TLS WG, draft-ietf-tls-ac509prof-00.txt, April 2002.

[5] KRAFT, R.: *Designing a distributed access control processor for network services on the Web*. In: *XMLSEC '02: Proceedings of the 2002 ACM workshop on XML security*, S. 36–52, New York, NY, USA, 2002. ACM Press.

[6] NYKAENEN, T.: *Attribute Certificates in X.509*. Techn. Ber., Helsinki University of Technology, 2000.

[7] OASIS: *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*, Feb 2006.

[8] OASIS: *Web Services Security X.509 Certificate Token Profile 1.1*, Feb 2006.

[9] RFC3280: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, April 2002.

[10] RFC4120: *The Kerberos Network Authentication Service (V5)*, July 2005.

[11] RITSKO, J. J. und A. BIRMAN: *Preface.*. IBM Systems Journal, 44(4):651–652, 2005.

[12] ROESSLER, T., D. SOLO, F. HIRSCH, D. EASTLAKE und J. REAGLE: *XML Signature Syntax and Processing (Second Edition)*. W3C Recommendation, W3C, Juni 2008. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/.

[13] SCHNEIER, B.: *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2004.

[14] SUHRBIER, L. und T. HILDMANN: *PKI based Access Control with Attribute Certificates for Data held on Smartcards*. Techn. Ber., Technical University of Berlin, May 2002.

Abbildungsverzeichnis

Tabellenverzeichnis