Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung
Seminar

Thies Rubarth

Sicherheitskonzepte in SOA auf Basis sicherer Webservices

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Thies Rubarth

Thema der Ausarbeitung
Seminar

Sicherheitskonzepte in SOA auf Basis sicherer Webservices

Stichworte

SOA Webservice PKI Sicherheit Kerberos

Kurzzusammenfassung

In dieser Ausarbeitung wird gezeigt, warum es in modernen Softwarearchitekturen, die auf Service Oriented Architecture (SOA) basieren, neuer Sicherheitskonzepte bedarf. Dieses Ausarbeitung gibt einen Überblick über geeignete Sicherheitsarchitekturen und zeigt einen Ansatz, wie diese mit Webservices umgesetzt werden können.

Thies Rubarth

Title of the paper

Security concepts in SOA based on secure web services

Keywords

SOA web service PKI security cerberos

Abstract

This paper is about the necessity of new security concepts in modern software architectures. An overview of applicable security architectures is given and an approach of how it can be implemented with web services is shown.

Inhaltsverzeichnis

1 Einleitung

Moderne Softwarearchitekturen zeichnen sich durch eine lose Koppelung aus. Die lose Koppelung ermöglicht es Bestandteile von Software effizient mit anderen Anwendungen zu verbinden, was zu einer hohen Wiederverwendbarkeit von Software führt. Insbesondere wird die Wiederverwendung über Unternehmensgrenzen hinweg ermöglicht. Dies führt zu neuen Möglichkeiten beim Business Process Re-Engineering, fordert aber auch ein Umdenken, wenn es um das Thema Sicherheit[1] geht. Durch das Öffnen der Unternehmensgrenzen steigt die Zahl der möglichen Angreifer und es kann nicht mehr davon ausgegangen werden, dass man jeden Kommunikationspartner kennt. Daher müssen stärkere Sicherheitsmaßnahmen ergriffen werden und es müssen neue Vertrauensmodelle etabliert werden.

1.1 Aufbau

Im ersten Kapitel wird ein kurzer Überblick über die Entwicklung der Softwarearchitekturen hin zur Service Oriented Architecture (SOA) gegeben und es wird erläutert warum bei SOA neue Sicherheitskonzepte benötigt werden. In den beiden folgenden Kapiteln werden allgemeine Sicherheitsanforderungen vorgestellt und es wird ein Überblick über Sicherheitskonzepte gegeben. In dem darauf folgenden Kapitel werden sichere Webservices vorgestellt. Im letzten Kapitel wird ein Ausblick gegeben, wie eine Sicherheitsarchitektur für eine „globale SOA“ aussehen kann und wie dies im Rahmen einer Masterarbeit untersucht werden kann.

2 Softwarearchitekturen

Die ersten Anwendungen für Unternehmen waren in der Regel in sich geschlossene Anwendungen, die zunächst dazu dienten die Daten, die bisher in Papierform gespeichert wurden, in elektronischer Form leichter zugänglich zu machen. Als begonnen wurde mit Anwendungen auch Geschäftsprozesse zu unterstützen, stellten sich die „Standalone-Anwendungen“ als unzureichend heraus, da es schwierig war die Anwendungen miteinander zu koppeln.

Die Qualität der Softwareunterstützung der Geschäftsprozesse ist heute entscheidend für den Erfolg des Unternehmens. Heutige Anwendungen müssen deshalb schnell an neue Anforderungen angepasst werden können. Dabei sollen alte Anwendungen möglichst in die neuen Systeme integriert werden um den Entwicklungsaufwand zu reduzieren. Dabei fordern die heterogenen IT-Landschafen in den Unternehmen einfache Schnittstellen um den Integrationsaufwand niedrig zu halten.

Im Zuge der wachsenden IT-Unterstützung von Geschäftsprozessen mussten Softwarearchitekturen entwickelt werden, die eine leichte Koppelung von Anwendungen ermögli-

[1] Im Englischen wird zwischen Safety (Zuverlässigkeit) und Security (Schutz) unterschieden. In dieser Ausarbeitung wird der Begriff Sicherheit mit der Bedeutung: „Schutz“ benutzt.

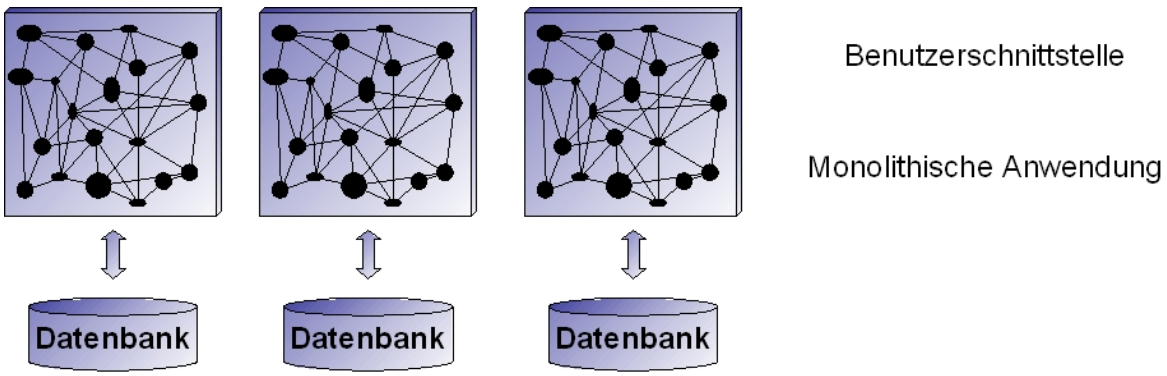

Abbildung 1: Zwei-Schichten-Anwendung

chen. In diesem Kapitel soll ein kurzer Überblick über die Entwicklung der Softwarearchitekturen hin zur Service Oriented Architecture (SOA) gegeben werden.

(Vgl. Woods (2003))

2.1 Monolithische Anwendungen

Die ersten Anwendungen waren sogenannte monolithische Anwendungen, bei denen die komplette Anwendung aus einem einzigen riesigen Programm besteht. Es gibt keine Trennung zwischen der Benutzerschnittstelle, der Anwendungslogik und den Daten. Die Daten werden in Dateien gespeichert.

Durch die enge Koppelung zwischen den einzelnen Funktionen einer monolithischen Anwendung gestaltet sich die Wartung einer solchen Anwendung sehr schwierig. Durch die fehlende Abstraktion werden die Anwendungen schnell unübersichtlich und Änderungen an einer Stelle können sich leicht durch die gesamte Anwendung fortführen. Die Verbindung von zwei monolithischen Anwendungen ist nur sehr umständlich realisierbar. Um zwei monolithische Anwendungen miteinander zu verbinden, müssen die Dateien, die von den einzelnen Anwendungen genutzt werden, ausgetauscht werden. Dabei sind die Programm eng an das Dateiformat und den Satzaufbau der Dateien gebunden. Mit dieser Koppelung ist es nicht möglich Anwendungsogik wiederzuverwenden, sondern lediglich Daten können gemeinsam genutzt werden.

2.2 Zwei-Schichten-Anwendungen

Mit Einführung von Datenbanken wurden die Datenschicht und die Anwendungsschicht voneinander getrennt. Diese Trennung ermöglicht es verschiedenen Anwendungen auf die gleichen Daten zuzugreifen, ohne dass jede Anwendung sämtliche Detailinformationen über die Daten benötigt (siehe Abbildung 1).

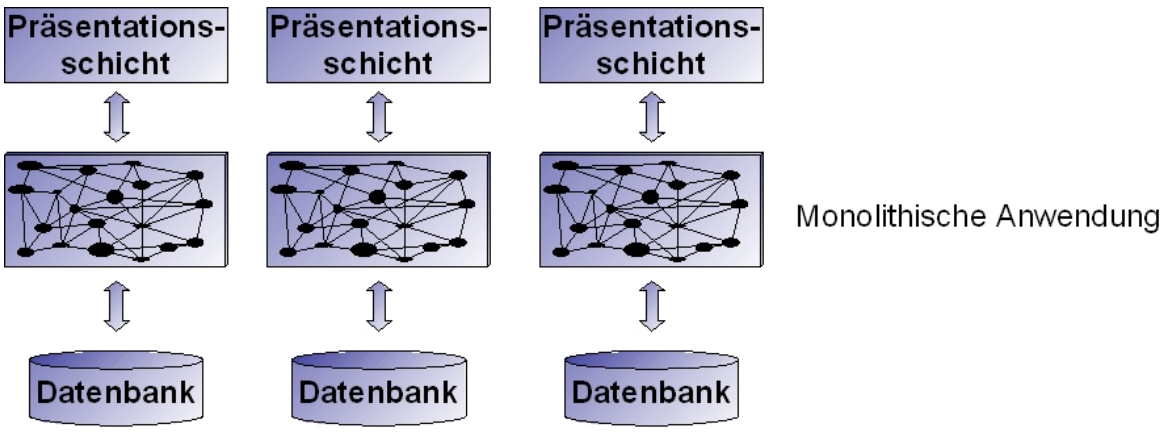

Abbildung 2: Drei-Schicht-Anwendung

Dennoch sind die Anwendungen weiterhin eng miteinander gekoppelt, da jede Anwendung beim schreibenden Zugriff auf die Daten deren Konsistenz sicherstellen muss. Es können also weiterhin nur Daten gemeinsam genutzt werden, wenn auch auf eine einfachere Art und Weise. Anwendungslogik kann nicht wiederverwendet werden und muss weiterhin redundant implementiert werden.

2.3 Drei-Schichten-Anwendungen

Im nächsten Schritt wurde auch die Benutzerschnittstelle von dem monolithischen Kern getrennt. Dies ermöglichte zunächst, dass Bentzerschnittstelle, Anwendungsogik und Datenbank jeweils auf unterschiedlichen Rechnern laufen können. Die Benutzerschnittstelle ist aber weiterhin eng mit dem monolithischen Kern der Anwendung verbunden (siehe Abbildung 2).

Trotz der Trennung der Benutzerschnittstelle vom monolithischen Kern sind Drei-Schichten-Anwendungen eng gekoppelt, da die Benutzerschnittstelle die genaue Schnittstelle der Anwendung kennen muss. In Unternehmen mit Drei-Schichten-Architektur stehen die Anwendungen in der Regel wie Datensilos nebeneinander. Durch erstellen von eigenen APIs ist es jedoch möglich die Anwendungen miteinander auf Ebene der Anwendungslogik zu verbinden. Diese Verbindung bedeuted aber weiterhin eine enge Koppelung, da die Anwendungen die APIs der anderen Anwendungen kennen müssen.

2.4 Service Oriented Architecture

Um die enge Koppelung von verschiedenen Anwendungen zu vermeiden, muss der monolithische Kern aufgespalten werden. Ein Ansatz dafür ist die Service Oriented Architecture (SOA). Bei SOA besteht die Anwendungslogik nicht mehr aus einem eng gekoppelten Kern,

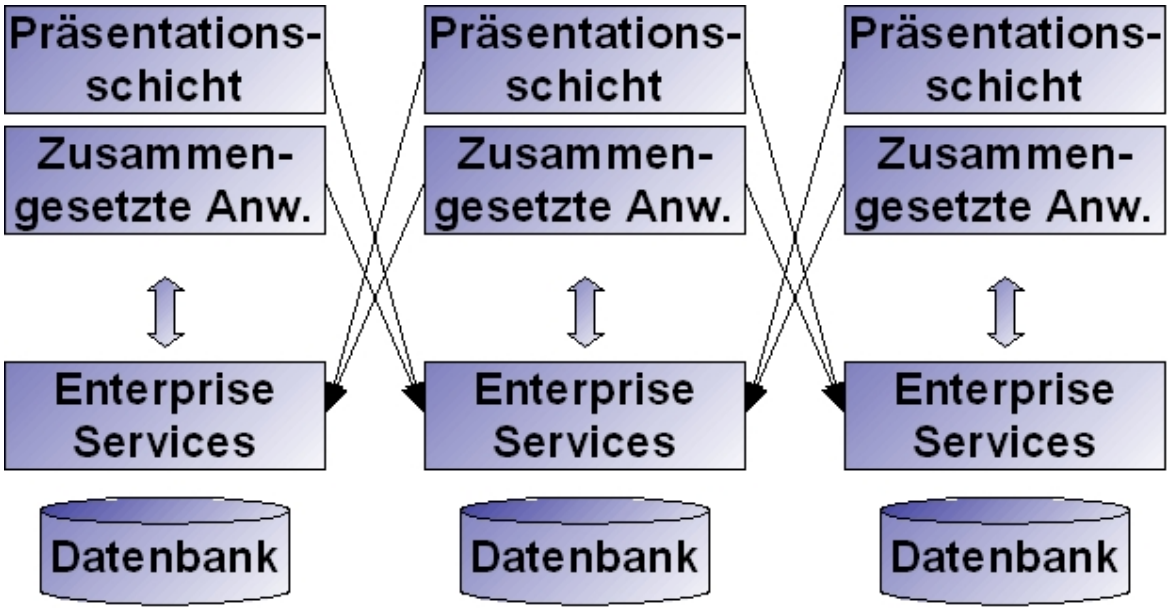

Abbildung 3: Service Oriented Architecture

sondern aus vielen lose gekoppelten Einzeldiensten, die über offen zugängliche Schnittstellen miteinander verbunden werden (siehe Abbildung 3).

Für die Implementierung von SOA haben sich Webservices als die derzeit am besten geeignetste Möglichkeit herausgestellt. Ein Webservice ist eine Anwendung, die über das Internet aufgerufen werden kann. Die Schnittstelle eines Webservices wird maschinenlesbar mit der Web Service Definition Language (WSDL) beschrieben. Für die Kommunikation werden einfache Internetprotokolle (zumeist HTTP) verwendet. Die augetauschten Nachrichten sind in der Regel XML-kodierte Nachrichten. Dazu wird das Simple Object Access Protocol (SOAP) verwendet.

Durch die maschinenlesbare Beschreibung der Schnittstelle kann ein Webservice mit wenig Aufwand in eine Anwendung integriert werden. Die verwendeten Protokolle werden von allen modernen Programmiersprachen unterstützt, so dass kein großer Aufwand getrieben werden muss, um die Kommunikation zwischen verschiedenen Systemen zu ermöglichen. Bei alten Anwendungen besteht die Möglichkeit Adapter für diese Protokolle zu schreiben (Vgl. Woods (2003), Erl (2005)).

Durch die leichte Integrierbarkeit von Webservices ist es möglich Prozesse schnell und effizient umzustrukturieren. Dabei können auch Webservices von anderen Unternehmen in eigene Anwendungen integriert werden. In Abbildung 4 ist ein Beispiel für solch ein Business Process Re-Engineering (BPR) abgebildet. In diesem Beispiel wird der Prozess P dadurch umstrukturiert, dass die Schritte A und B von einem Geschäftspartner 1 übernommen

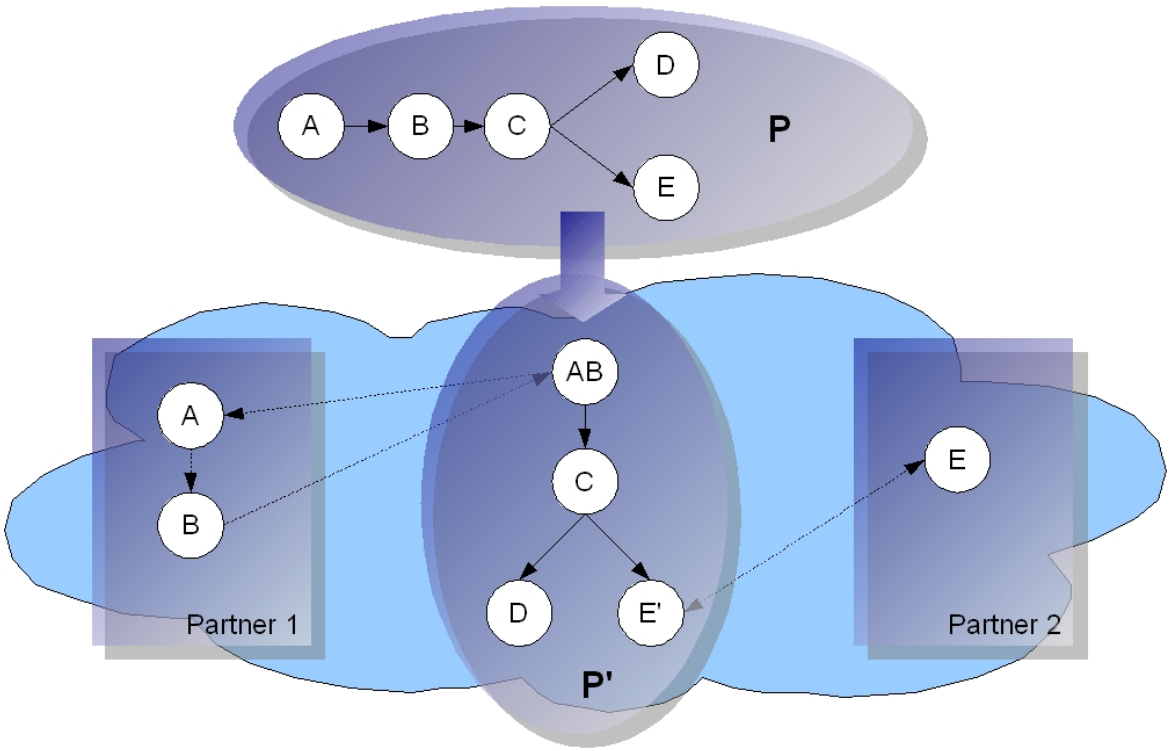

Abbildung 4: Business Process Re-Engineering

und der Schritt E von einem Geschäftspartner 2 übernommen wird (Vgl. Weerawarana u. a. (2005)).

3 Sicherheitsanforderungen

Bei SOA wird das Internet als Infrastruktur genutzt. Das Internet ist eine sehr offene Infrastruktur, so dass potentielle Angreifer problemlos Daten, die über das Internet versendet werden, abfangen und manipulieren können, wenn keine Maßnahmen ergriffen werden, um dies zu verhindern.

Bei monolithischen Anwendungen sind die Personen, die Zugriff zu dem System haben, persönlich bekannt. Man kann diesen Personen z.B. persönlich ein Passwort überreichen, mit dem sie sich am System authentifizieren können. Bei Unternehmensübergreifenden Prozessen ist es unter Umständen nicht möglich jeden Kommunikationspartner persönlich zu kennen. Daher müssen neue Vertrauensmodelle etabliert werden, die es ermöglichen neuen Partnern Zugriff zu einem System zu gewähren, ohne diesen persönlich zu kennen.

3.1 Schutzbedarf

Wenn ein Softwaresystem nicht sicher ist, können potentielle Angreifer Daten aus dem System stehlen oder Daten im System verändern. Dies kann unterschiedliche Auswirkungen haben, die im folgenden aufgezählt werden:

- Verstoß gegen Gesetze oder Verträge
- Beeinträchtigung des informationellen Selbstbestimmungsrechts
- Beeinträchtigung der persönlichen Unversehrtheit (z.B. in Krankenhäusern)
- Beeinträchtigung der Aufgabenerfüllung
- Negative Auswirkungen (Image des Unternehmens)
- Finanzielle Auswirkungen

Um die Notwendigkeit einer Sicherheitsarchitektur und deren Stärke festzustellen muss eine Schutzbedarfsanalyse durchgeführt werden, in der die möglichen Auswirkungen bei erfolgreichen Angriffen und deren Schwere bestimmt werden. Daraus ergibt sich welche Gegenmaßnahmen ergriffen werden müssen, bzw. vernachlässigt werden können (Vgl. Eckert (2005)).

3.2 Schutzziele

Damit eine Anwendung sicher ist, müssen folgende Ziele erreicht sein:

Vertraulichkeit Daten können nur von berechtigten Personen gesehen werden.

Integrität Daten können nicht unbemerkt und insbesondere nicht von unberechtigten Personen geändert werden.

Authentifizierung Die Identität eines Benutzers kann zweifelsfrei festgestellt werden.

Autorisierung Für jeden Benutzer kann festgestellt werden, welche Aktionen er ausführen darf.

4 Sicherheitskonzepte

Um beim Austausch von Daten die Vertraulichkeit und die Integrität sicherzustellen, werden die Daten verschlüsselt. Beim Verschlüsseln wird zwischen den symmetrischen und den asymmetrischen Verschlüsselungsverfahren unterschieden. Bei symmetrischen Verfahren wird zum ver- und entschlüsseln derselbe geheime Schlüssel verwendet. Bei asymmetrischen Verfahren werden unterschiedliche Schlüssel verwendet. Bei asymmetrischen Verfahren hat man dadurch den Vorteil, dass man den geheimen Schlüssel nicht austauschen muss, sondern den zum Verschlüsseln verwendeten Schlüssel öffentlich mitteilen kann.

Um den Absender einer Nachricht sicherzustellen gibt es die Möglichkeit Nachrichten zu signieren. Dabei wird z.B. die Nachricht (oder ein Hashwert davon) mit dem geheimen Schlüssel des Absenders verschlüsselt. Hat der Empfänger der Nachricht den öffentlichen Schlüssel des Absenders kann er die Signatur entschlüsseln und kann somit sicherstellen, dass die Nachricht tatsächlich von dem Absender kommt, da nur dieser seinen geheimen Schlüssel kennt.

Um bei dieser Art der Signierung die tatsächliche Identität des Absenders sicherzustellen, muss garantiert sein, dass der öffentliche Schlüssel auch tatsächlich zu der Person gehört. Im folgenden werden zwei Konzepte vorgestellt, die „globales Vertrauen“ schaffen sollen.

4.1 Public Key Infrastructure

Bei einer Public Key Infrastructure (PKI) belegen Zertifikate[2] die Gültigkeit eines öffentlichen Schlüssels. Der Schlüssel wird damit an eine natürliche oder juristische Person gebunden. Die Zertifikate werden von einem Trust Center ausgestellt. Das Trust Center signiert das ausgestellte Zertifikat mit seinem eigenen Zertifikat. Um die Gültigkeit eines Zertifikates zu prüfen muss also die Gültigkeit des Zertifikats des Trust Centers sichergestellt werden.

Damit nicht jeder die Zertifikate aller Trust Center kennen muss, sind die Trust Center in einer Hierarchie angeordnet (siehe Abbildung 5). Ausgangspunkt der Hierarchie ist die Internet Policy Registration Authority (IPRA). Die IPRA zertifiziert die Policy Certification Authorities (PCA), die jede eine eigene Strategie zum Vergeben von Zertifikaten definieren können. Die PCAs wiederum zertifizieren die Certification Auhorities (CA), die entweder noch weiter untergliedert sind oder direkt die Benutzer zertifizieren.

Um ein Zertifikat zu validieren, das aus einem anderen Vertrauensbereich der PKI stammt, muss also ein Pfad durch die Hierarchie gefunden werden, der sicherstellt, dass das Zertifikat gültig ist. Dabei können auch Cross-Zertifikate verfolgt werden, mit denen eine CA die Vertrauenswürdigkeit einer anderen CA zertifiziert (Vgl. Eckert (2005)).

[2] Der heute im Internet verwendete Standard sind X509-Zertifikate

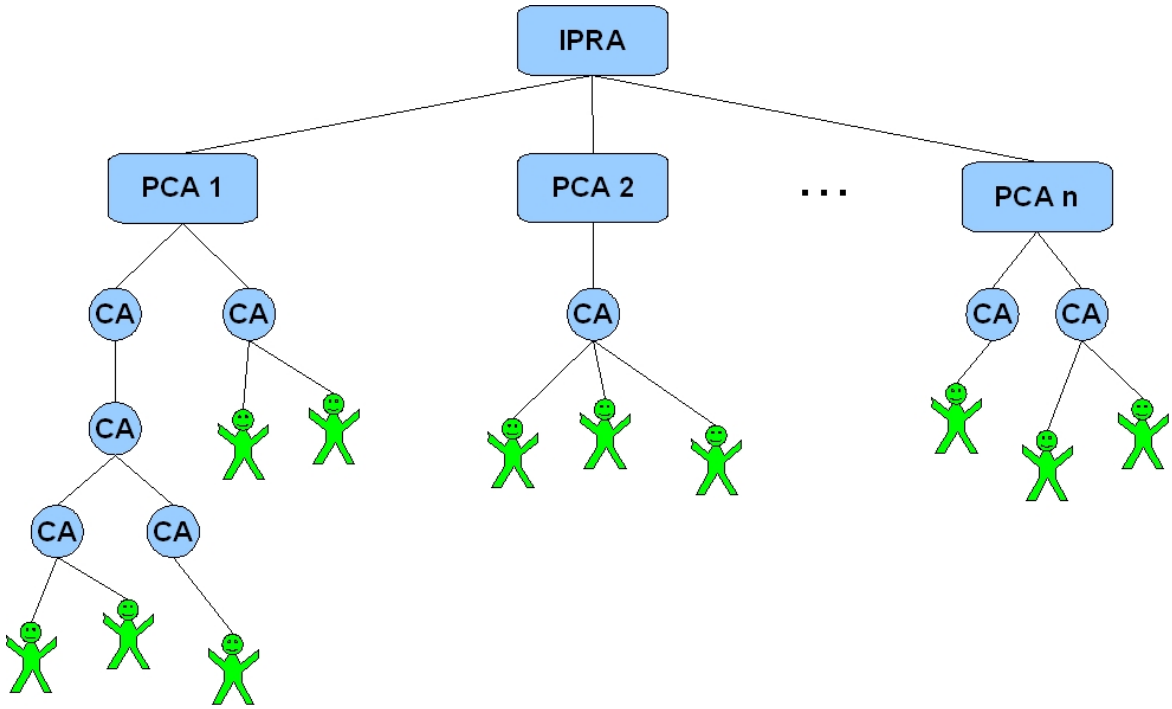

Abbildung 5: Public Key Infrastructure

4.2 Kerberos

Kerberos ist ein Konzept, das ursprünglich ohne asymmetrische Verschlüsselungsverfahren auskommt. Inzwischen gibt es aber auch Ansätze, die Kerberos mit asymmetrischen Verschlüsselungsverfahren kombinieren.

Die Basis in einem Kerberos-System stellt das Key Distribution Center (KDC) dar (siehe Abbildung 6). Das KDC besteht aus einem Authentication Server (AS) und dem Ticket Granting Service (TGS). Meldet sich ein Benutzer am System an, muss er sich zunächst mit seinem Schlüssel, der in der Regel aus dem Passwort der Benutzers generiert wird, beim AS authentifizieren (Schritt 1 und 2 in der Abbildung 6). Der AS kennt den Schlüssel des Benutzers und stellt ihm das Initialticket aus (Schritt 3), das den Benutzer berechtigt Tickets beim TGS anzufordern. Möchte der Benutzer z.B. die Dienste eines Servers nutzen, muss er beim TGS ein Ticket dafür anfordern (Schritt 4). Hat der Benutzer ein gültiges Initialticket mitgeschickt, stellt der TGS das Ticket aus, das den Benutzer zum Zugriff auf den Server berechtigt (Schritt 5). Mit diesem Tickte kann der Benutzer dann auf den Server zugreifen (Schritt 6) (Vgl. Eckert (2005)).

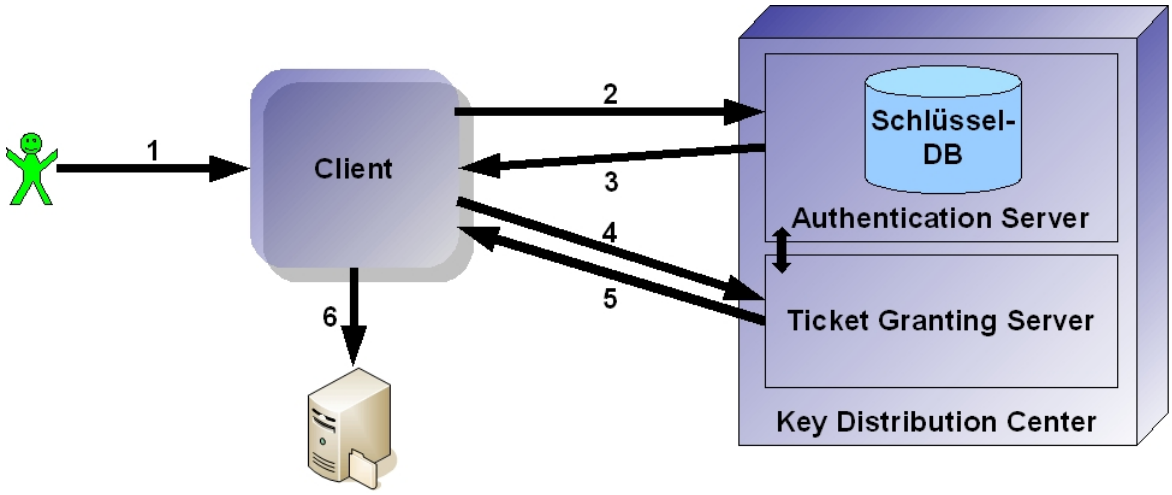

Abbildung 6: Funktionsweise von Kerberos

5 Sichere Webservices

Um die im vorherigen Kapitel vorgestellten Sicherheitskonzepte in SOA Umsetzen zu können, muss es möglich sein die Nachrichten, die von Webservices ausgetauscht werden, sicher zu übertragen. In diesem Kapitel werden zwei Ansätze vorgestellt, die eine sichere Übertragung von SOAP-Nachrichten ermöglichen.

5.1 Secure Socket Layer

Secure Socket Layer (SSL) wird bei zahlreichen Internetanwendungen eingesetzt und ist die Basis für HTTPS. SSL ermöglicht die Authentifizierung des Clients und des Servers und eine verschlüsselte Datenübertragung. Dabei können je nach Fähigkeiten des Servers und des Clients unterschiedliche Protokolle verwendet werden, die in der Handshake-Phase zwischen Client und Server verabredet werden.

SSL ist damit in der Lage eine sichere Datenübertragung zwischen zwei direkt miteinander kommunizierenden Partnern auf der Transportebene zu garantieren. Wenn es aber erforderlich ist einen Sicherheitskontext herzustellen, der über mehrere Partner hinweg besteht, stößt SSL an seine Grenzen (siehe Abbildung 7, Vgl. Weerawarana u. a. (2005)).

5.2 WS-* Spezifikationen

Webservices sind Programme, die über das Internet aufgerufen werden können. Wie die Übertragung der Nachrichten, die dazu ausgetauscht werden müssen, abläuft ist nicht festgelegt. Die Programme, die Webservices nutzen, müssen daher nichts darüber wissen und

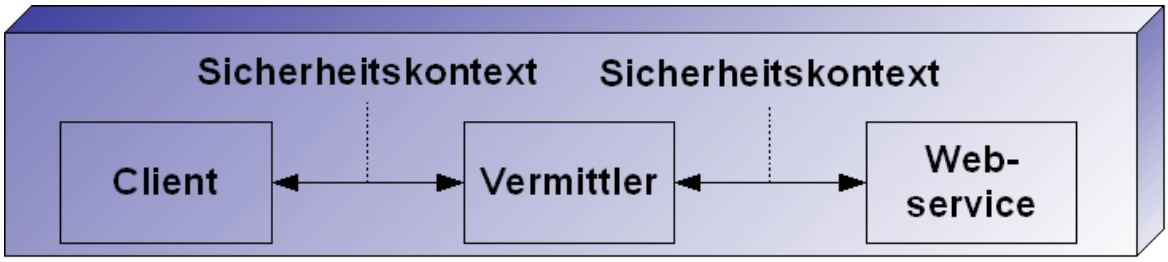

Abbildung 7: Sicherheitskontext von SSL

können das Übertragen der Daten einer Middleware überlassen. Um verschiedene Aspekte von Quality of Service dabei zu berücksichtigen, werden zahlreiche Spezifikationen für Webservices entwickelt, die sich unter anderem auch mit Sicherheit auseinander setzen. In diesem Kapitel wird ein kurzer Überblick über die Spezifikationen für sichere Webservices gegeben (Vgl. Weerawarana u. a. (2005)).

5.2.1 WS-Policy, WS-SecurityPolicy

WS-Policy ermöglicht das Beschreiben von Anforderungen eines Webservices. Dabei können die Anforderungen verschachtelt und auf beliebige Weise miteinander kombiniert werden. WS-SecurityPolicy spezifiziert besondere Anforderungen, die im Zusammenhang mit WS-Security, WS-Trust und WS-SecureConversation benötigt werden (Vgl. Bajaj u. a. (2004), Della-Libera u. a. (2005)).

5.2.2 WS-Security

WS-Security spezifiziert wie Security Tokens (z.B. Benutzerinformation, verschlüsselte Daten oder Signaturen) in Webservice-Nachrichten eingebunden werden können. Für verschlüsselte Daten und Signaturen können XML-Encryption und XML-Signature verwendet werden. Dadurch, dass die Sicherheitsmechanismen in der Nachricht selbst verankert sind und nicht wie bei SSL außen herumgelegt werden, ist es möglich einen Sicherheitskontext aufzubauen, der sich über mehrere Partner erstreckt (siehe Abbildung 8, Vgl. Antony u. a. (2005)).

5.2.3 WS-Trust

WS-Trust spezifiziert Tokens, über die Vertrauen erzeugt werden kann. Dabei wird nicht festgelegt, welche Tokens benötigt werden, damit eine Nachricht vertrauenswürdig ist, sondern es wird ein Protokoll zum Anfordern, Erstellen und Austauschen von Tokens definiert. Für eine PKI oder ein Kerberos-System können z.B. X509- oder Kerberos-Tokens erstellt werden (Vgl. Anderson u. a. (2005b)).

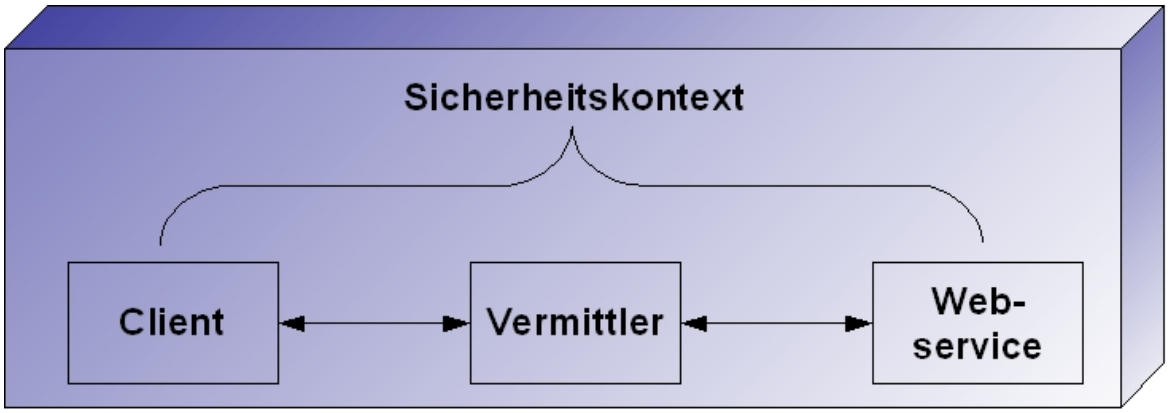

Abbildung 8: Übergreifender Sicherheitskontext

5.2.4 WS-SecureConversation

WS-SecureConversation ermöglicht die Erstellung eines Sicherheitskontextes, der über mehrere Webservice-Aufrufe hinweg bestehen kann. Dabei wird eine geheime Information ausgetauscht, die dann z.B. zum Ableiten von Schlüsseln dienen kann, mit denen Nachrichten verschlüsselt werden (Vgl. Anderson u. a. (2005a)).

5.3 WS-Federation

WS-Federation legt fest welche Tokens ausgetauscht werden müssen um Vertrauen zu erzeugen. Dabei werden insbesondere verschiedene Modelle erstellt, wie der Wechsel von einem Vertrauensbereich (Realm) in einen anderen vollzogen werden kann (Vgl. Bajaj u. a. (2003)).

5.3.1 Weitere Spezifikationen

Zu den genannten Spezifikationen gibt es teilweise noch Unterspezifikationen, die sich mit Detailproblemen beschäftigen. Darüber hinaus werden noch weitere Spezifikationen entwickelt. Davon sind angekündigt: WS-Authorization und WS-Privacy.

6 Ausblick

SOA bietet die Möglichkeit Anwendungen zu implementieren, die auf global verteilten Systemen arbeiten. Wie in dieser Ausarbeitung gezeigt wird, müssen dafür neue Sicherheitskonzepte etabliert werden, um Vertrauen herstellen zu können. Mit Hilfe der WS-* Spezifika-

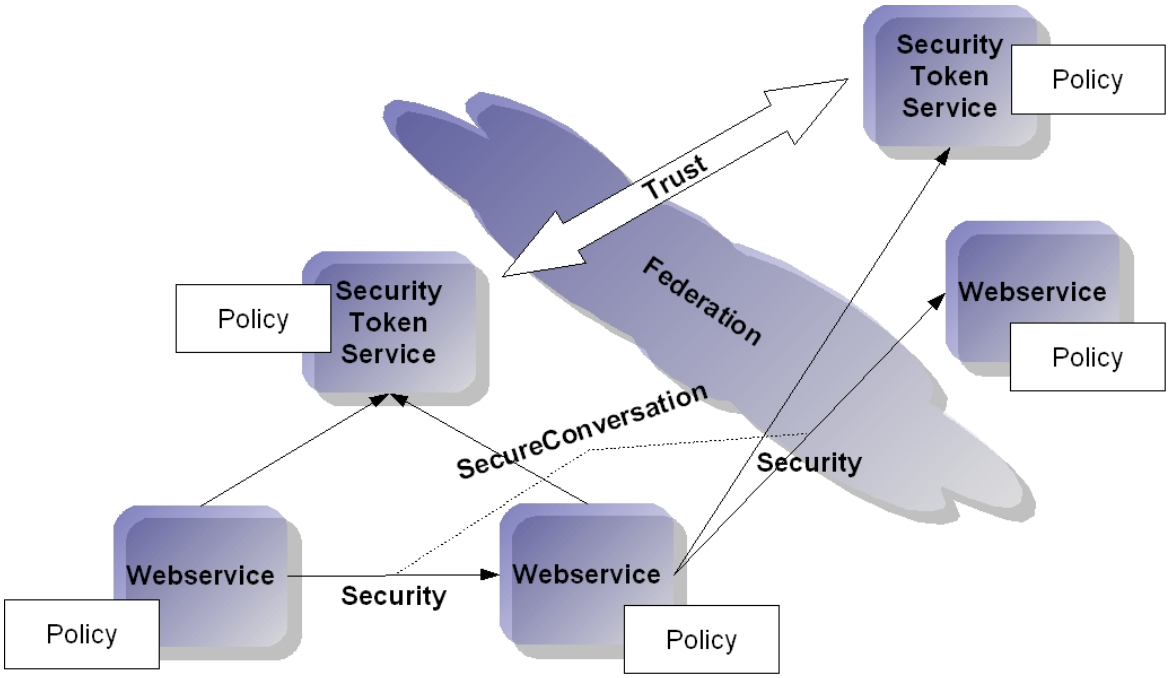

Abbildung 9: WS-* Spezifkationen im Überblick

tionen scheint es möglich zu sein, neue Sicherheitskonzepte in SOA zu realisieren. Wie die einzelnen Spezifikationen dabei zusammen spielen ist in Abbildung 9 zu sehen.

Im Rahmen einer Masterabeit kann die Sicherheitslandschaft, wie sie in der Abbildung zu sehen ist konkretisiert werden. Dabei muss geprüft werden, welches Sicherheitskonzept (PKI, Kerberos oder andere) für eine globale Sicherheitsarchitektur geeignet ist. Es muss ein Modell für ein Rahmenwerk entworfen werden, das zeigt wie Anwendungen in eine solche Sicherheitsarchitektur eingebettet werden können, so dass kein oder nur sehr wenig Implementierungsaufwand notwendig ist. Der größte Teil sollte durch das Rahmenwerk übernommen werden und nur konfiguriert werden müssen.

Die Machbarkeit eines solchen Rahmenwerkes muss durch einen Prototypen oder ggf. mehrere Teilprototypen belegt werden. Darüber hinaus muss die Skalierbarkeit einer solchen Architektur untersucht werden.

6.1 Risiken

Bei dem Vorhaben ergeben sich im wesentlichen zwei Risiken. Die WS-* Spezifikation befinden sich zum größten Teil noch in der Entwicklung oder sind erst angekündigt. Zudem ist die Masse an Spezifikationen nur schwer zu überschauen. Dadurch besteht die Gefahr, dass zuviel Zeit beim Erarbeiten der Spezifikationen benötigt wird.

Das zweite Risiko ist die fehlende Praxiserfahrung mit globalen Sicherheitsarchitekturen.

Modelle für zwei miteinander kommunizierende Partner sind in der Praxis erprobt, aber das Zusammenschließen von beliebig vielen Kommunikationspartnern stellt neue Probleme, die evtl. erstmal vernachlässigt werden müssen. Insbesondere muss geprüft werden ob und wie die in den Unternehmen bestehenden Sicherheitsarchitketuren in eine globale Sicherheitsarchitektur integriert werden können.

Literatur

[Anderson u. a. 2005a] ANDERSON, Steve ; BOHREN, Jeff ; BOUBEZ, Toufic ; CHANLIAU, Marc ; DELLA-LIBERA, Giovannis ; DIXON, Brendan ; GARG, Praerit ; GUDGIN, Martin ; HADA, Satoshi ; HALLAM-BAKER, Phillip ; HONDO, Maryann ; KALER, Chris ; LOCKHART, Hal ; MARTHERUS, Robin ; MARUYAMA, Hiroshi ; NADALIN, Anthony ; NAGARATNAM, Nataraj ; NASH, Andrew ; PHILPOTT, Rob ; PLATT, Darren ; PRAFULLCHANDRA, Hemma ; SAHU, Maneesh ; SHEWCHUK, John ; SIMON, Dan ; SRINIVAS, Davanum ; WAINGOLD, Elliot ; WAITE, David ; WALTER, Doug ; ZOLFONOON, Riaz: Web Services Secure Conversation Language. (2005)

[Anderson u. a. 2005b] ANDERSON, Steve ; BOHREN, Jeff ; BOUBEZ, Toufic ; CHANLIAU, Marc ; DELLA-LIBERA, Giovannis ; DIXON, Brendan ; GARG, Praerit ; GUDGIN, Martin ; HALLAM-BAKER, Phillip ; HONDO, Maryann ; KALER, Chris ; LOCKHART, Hal ; MARTHERUS, Robin ; MARUYAMA, Hiroshi ; NADALIN, Anthony ; NAGARATNAM, Nataraj ; NASH, Andrew ; PHILPOTT, Rob ; PLATT, Darren ; PRAFULLCHANDRA, Hemma ; SAHU, Maneesh ; SHEWCHUK, John ; SIMON, Dan ; SRINIVAS, Davanum ; WAINGOLD, Elliot ; WAITE, David ; WALTER, Doug ; ZOLFONOON, Riaz: Web Services Trust Language. (2005)

[Antony u. a. 2005] ANTONY, Nadalin ; KALER, Chris ; HALLAM-BAKER, Phillip ; MONZILLO, Ronald: Web Services Security: SOAP Message Security 1.1. (2005)

[Bajaj u. a. 2004] BAJAJ, Siddharth ; BOX, Don ; CHAPPELL, Dave ; CURBERA, Francisco ; DANIELS, Glen ; HALLAM-BAKER, Phillip ; HONDO, Maryann ; KALER, Chris ; LONGWORTHY, Dave ; MALHOTRA, Ashok ; NADALIN, Anthony ; NAGARATNAM, Nataraj ; NOTTINGHAM, Mark ; PRAFULLCHANDRA, Hemma ; RIEGEN, Claus von ; SCHLIMMER, Jeffrey ; SHARP, Chris ; SHEWCHUK, John: Web Service Policy. (2004)

[Bajaj u. a. 2003] BAJAJ, Siddharth ; DELLA-LIBERA, Giovannis ; DIXON, Brendan ; DUSHE, Mike ; HONDO, Maryann ; HUR, Matt ; KALER, Chris ; LOCKHART, Hal ; MARUYAMA, Hiroshi ; NADALIN, Anthony ; NAGARATNAM, Nataraj ; NASH, Andrew ; PRAFULLCHANDRA, Hemma ; SHEWCHUK, John: Web Services Federation Language. (2003)

[Della-Libera u. a. 2005] DELLA-LIBERA, Giovannis ; GUDGIN, Martin ; HALLAM-BAKER, Phillip ; HONDO, Maryann ; GRANQVIST, Hans ; KALER, Chris ; MARUYAMA, Hiroshi ; MCINTOSH, Micheal ; NADALIN, Anthony ; NAGARATNAM, Nataraj ; PHILPOTT, Rob ; PRAFULLCHANDRA, Hemma ; SHEWCHUK, John ; WALTER, Doug ; ZOLFONOON, Riaz: Web Services Security Policy Language. (2005)

[Eckert 2005] ECKERT, Prof. Dr. C.: *IT-Sicherheit (Studienausgabe)*. Oldenbourg, 2005

[Erl 2005] ERL, Thomas: *Service-Oriented Architecture*. Prentice Hall PTR, 2005

[Weerawarana u. a. 2005] WEERAWARANA, Sanjiva ; CURBERA, Francisco ; LEYMANN, Frank ; STOREY, Toney ; FERGUSON, Donald F.: *Web Services Plattform Architecture*. Prentice Hall PTR, 2005

[Woods 2003] WOODS, Dan: *Enterprise Service Architecture*. O'Reilly, 2003