TeleTrusT Deutschland e.V.

Der IT-Sicherheitsverband.

SOA Security in der Praxis

Entwurfsmuster für eine sichere Umsetzung

TeleTrusT-Arbeitsgruppe "SOA Security"

Impressum

Herausgeber:

TeleTrusT Deutschland e.V.
Chausseestraße 17
10115 Berlin
Tel.: +49 30 400 54 306
Fax: +49 30 400 54 311
E-Mail: info@ TeleTrusT.de
http://www.TeleTrusT.de

Herstellung:

DATEV eG, Nürnberg

1. Auflage

© 2010 TeleTrusT

Autoren:

Dr. Bruno Quint



Michael Menzel

Ivonne Thomas



Dr. Thomas Störtkuhl



Eva Saar



Simone Achtert-Wanner (TÜVIT), Markus Bartsch (TÜVIT), Hans-Joachim Bierschenk (TeleTrusT), Holger Junker (BSI), Andreas Meine (DRV), Andre Miede (TU Darmstadt), Martin Oczko (Sophos), Andreas Philipp (Sophos), Martin Räpple (SAP), Prof. Dr. Helmut Reimer (TeleTrusT), Omid Amir-hamzeh Tafreschi (Fraunhofer-Institut SIT)

(TeleTrusT-Arbeitsgruppe "SOA Security")

Inhalt

Vorwort 6

1. Einleitung 7
1.1 Motivation 7
1.2 Anforderungen an SOA Security Lösungen 8
1.3 Sicherheitsanforderungen in Service-Orientierten Architekturen 9
Anforderungen an Authentifizierung 9
Anforderungen an Autorisierung 9
Anforderungen an Integrität 9
Anforderungen an Vertraulichkeit und Datenschutz 9
Anforderungen an Verbindlichkeit (Nichtabstreitbarkeit) 10
Weitere Sicherheitsanforderungen 10
1.4 Grundlage der Betrachtungen in diesem Dokument 10
1.5 Aufbau des Dokuments 12

2. Wie wird dieser Entwurfsmusterkatalog verwendet? 14
2.1 Ansatz 14
2.2 Architekturbausteine 15
2.3 Anwendungsfälle 17
2.4 Auswahl der Entwurfsmuster 18
2.5 Entwurfsmuster 18
2.6 Schutzbedarf 19
2.7 Auswahl der Technologien und Methoden 19
2.8 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur 19

3. Architekturbetrachtungen für Service-Orientierte Architekturen 20
3.1 Nachrichtenaustausch 20
3.2 Vertrauensbeziehungen und Integration von Security Providern 21

4. Anwendungsfälle für Service-Orientierte Architekturen 23
4.1 Anwendungsfälle hinsichtlich der Authentifizierung 23
4.2 Anwendungsfälle hinsichtlich der Datensicherheit (Integrität/Vertraulichkeit) 23
4.3 Anwendungsfälle hinsichtlich der Nichtabstreitbarkeit 23
4.4 Anwendungsfälle hinsichtlich der Autorisierung 24

5. Entwurfsmuster für Authentizität 25
5.1 Auswahl der Entwurfsmuster 25
5.2 Entwurfsmuster 26
5.3 Auswahl der Technologien und Methoden 30
5.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur 30

6. Entwurfsmuster für Integrität 31
6.1 Auswahl der Entwurfsmuster 31
6.2 Entwurfsmuster 32
6.3 Auswahl der Technologien und Methoden 35
6.4. Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur 35

7. Entwurfsmuster für Vertraulichkeit 36
7.1 Auswahl der Entwurfsmuster 36
7.2 Entwurfsmuster 36
7.3 Auswahl der Technologien und Methoden 39
7.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur 39

8. Entwurfsmuster für Nichtabstreitbarkeit 41
8.1 Auswahl der Entwurfsmuster 41
8.2 Entwurfsmuster 44
8.3 Auswahl der Technologien und Methoden 48
8.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur 49

9. Entwurfsmuster für Autorisierung 50
9.1 Auswahl der Entwurfsmuster 50
9.2 Entwurfsmuster 50
9.3 Auswahl der Technologien und Methoden 56

10. Beispiel zur Vorgehensweise 57
10.1 Auswahl Architektur 58
10.2 Auswahl der Anwendungsfälle 59
10.3 Entwurfsmuster 59
10.4 Schutzbedarf der verarbeiteten Daten 60
10.5 Auswahl der Technologien und Methoden 60
10.6 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur 61

11. Anhang 62

Vorwort

TeleTrusT Deutschland verfolgt im Besonderen das Anliegen, die Vertrauenswürdigkeit von Informations- und Kommunikationstechnik in einer offenen Systemumgebung zu fördern.

Die vorliegende Publikation entstand aus der Motivation heraus, die unterschiedlichen Ansätze, Technologien und Methoden im Bereich der Service-Orientierten Architekturen in einem Pattern- oder, zu deutsch, Entwurfsmusterkatalog zusammenzufassen, der es Entwicklern und Sicherheitsverantwortlichen erleichtert, Alternativen und Möglichkeiten für die Umsetzung von Sicherheit in allgegenwärtigen SOA Sicherheitsszenarien zu finden.

Geordnet nach den bekannten Sicherheitszielen Authentizität, Integrität, Vertraulichkeit, Nichtabstreitbarkeit und Autorisierung werden verschiedene Entwurfsmuster vorgestellt und mit Anwendungsfällen, Architekturen und verwendbaren Technologien beschrieben.

Dieses Dokument wurde von der TeleTrusT-Arbeitsgruppe "SOA Security" im Rahmen regelmäßiger Arbeitsgruppentreffen erarbeitet.

Einen besonderen Dank möchten wir all denjenigen aussprechen, die mit uns gemeinsam in zahlreichen, wenn manchmal vielleicht auch endlos erscheinenden, doch am Ende sehr fruchtbaren Diskussionen dieses Dokument erst zu dem machten, was es ist: Ein Resümee aus unterschiedlichsten fachlichen Meinungen und Ansichten in Form eines Kataloges von Entwurfsmustern zur Umsetzung von Sicherheit in Service-Orientierten Architekturen.

Wir wünschen eine interessante und nützliche Lektüre.

Die Autoren

1. Einleitung

1.1 Motivation

Die Absicherung von Service-Orientierten Architekturen geht einher mit einer hohen Komplexität bedingt durch die große Anzahl an möglichen Ansätzen, Technologien und Standards. Dieses Dokument verfolgt das Ziel eine Empfehlung hinsichtlich der Nutzung von SOA-Sicherheitstechnologien zu ermöglichen basierend auf einer Vorgabe von Sicherheitsanforderungen und einer Bewertung verfügbarer Sicherheitsmechanismen.

Service-Orientierte Architekturen (SOA) werden vermehrt in organisationsübergreifenden Projekten umgesetzt. Die Idee, Funktionalität als lose gekoppelte, unabhängige Services über standardisierte Schnittstellen anzubieten, ermöglicht Organisationen Services nach ihren Bedürfnissen zu orchestrieren und im Bedarfsfall gegeneinander auszutauschen. Organisationen können damit flexibler auf sich ändernde Geschäftsprozesse reagieren und sich den Marktbedingungen anpassen. Volle Flexibilität und damit verbundene Kostenersparnisse können jedoch nur erreicht werden, wenn Services bestimmten Anforderungen genügen. Die Literatur beschreibt eine Vielzahl an Kriterien an Services, von denen die verbreiteten sind:

- Standardisierte Service Schnittstellen – um die Interoperabilität sicherzustellen.
- Lose Kopplung – um Abhängigkeiten zu minimieren und die Modellierbarkeit verschiedener Services zu einem Geschäftsprozess zu vereinfachen.
- Funktionsabstraktion – um eine hohe Wiederverwendbarkeit zu erreichen.
- Serviceautonomie – um ein (weitgehend) autarkes Agieren von Services zu erlauben
- Vorratsdatenfreiheit des Service (Statelessness) – um Services austauschbar zu machen. Persistente Daten werden nur gespeichert, wenn dies explizit ein Dienstleistungsbestandteil des Service ist.
- Auffindbarkeit des Services – um über definierte Repositories eine Orchestrierbarkeit der Services zu einem Gesamtprozess zu verbinden.

Grundsätzlich ist SOA ein Architekturkonzept. Es schreibt keine konkrete technische Realisierung oder bestimmte Methoden vor. Allerdings haben sich Web Services (basierend auf SOAP/ XML) zur Realisierung durchgesetzt. Ein Web Service ist eine auf XML (eXtensible Markup Language) und SOAP (Simple Object Access Protocol) basierende Anwendung, die definierte Methoden über eine standardisierte Schnittstelle anbietet und über eine URI (Uniform Resource Identifier) eindeutig identifizierbar macht. Als Kommunikationsprotokoll dient SOAP (W3C), mit dem Daten zwischen Systemen ausgetauscht werden können.

SOAP verwendet XML zur Datenrepräsentation sowie (in den meisten Fällen) TCP/IP für den Transport. Zu einer SOA Infrastruktur gehört weiterhin ein Service Repository, welches meist über die technischen Spezifikationen UDDI (Universal Description, Discovery and Integration) und WSDL (Web Services Description Language) realisiert wird.

1.2 Anforderungen an SOA Security Lösungen

Im Grunde gelten für SOA die gleichen Sicherheitsanforderungen wie für herkömmliche monolithische Systeme. Zusätzlich existieren durch die Verteilung von Systemen und Prozessen SOA-spezifische Sicherheitsrisiken, die entsprechend zu berücksichtigen sind.

Wie in monolithischen Systemen müssen auch in SOA-Infrastrukturen Identitäten und Berechtigungen verwaltet, Benutzer authentisiert, Zugriffsberechtigungen überprüft, Zugriffe und Zugriffsversuche aufgezeichnet und Daten mittels kryptografischer Methoden signiert (Integrität) oder verschlüsselt (Vertraulichkeit) werden. Gerade in verteilten Umgebungen werden zudem systemweite Sicherheitsrichtlinien (Policies) benötigt, deren Einhaltung (Compliance) es permanent zu überwachen gilt.
Die Bedrohungspotentiale lassen sich grob in zwei Kategorien einteilen. Zum einen wird eine SOA Infrastruktur basierend auf einem Netzwerk/IT-System allgemeinen Gefahren wie Malware, Buffer Overflows, Backdoors, Netzangriffen (Scanning, Spoofing, etc.) sowie Denial of Service (DoS) und vielen mehr ausgesetzt.

Zum anderen wirft der Grundgedanke von SOA weitere Anforderungen und Bedrohungspotentiale auf, die durch die Integration verteilter und teilweise heterogener Systeme insbesondere zu beachten sind. So müssen Replay-Attacken, Service- Kompromittierung sowie unberechtigte Service-Nutzung durch adäquate Mechanismen verhindert werden. Die Realisierung eines Geschäftsprozesses über eine Vielzahl lose gekoppelter Services erfordert zudem mehr Authentifizierungs- und Autorisierungsvorgänge, eine jederzeit gewährleistete Vertraulichkeit sowie eine höhere Integrität als in einem monolithischen System. Ein Prozess-Design über Unternehmensgrenzen hinweg und zwischen unternehmensfremden Teilnehmern verstärkt diese Sicherheitsanforderungen an Services und Lösungen und setzt Federationkonzepte voraus. Technisch gesehen muss dabei die Interoperabilität sichergestellt werden. Zu nennen sind an dieser Stelle beispielsweise die Umwandlung von Identitäten und deren Formate, der Einsatz von Industriestandards und die Gewährleistung der Wiederverwendbarkeit von Security Services. Herkömmliche Sicherheitsanforderungen müssen um SOA-spezifische Ausprägungen erweitert werden.

Gerade in einer dezentralen und komplexen Architektur muss gewährleistet werden, dass das Gesamtsystem beherrschbar bleibt. Bei der Umsetzung eines Sicherheitskonzeptes für eine Service-Orientierte Architektur ist eine zentrale Administration daher ein wichtiger Bestandteil. Es ist unabdingbar, dass ein Sicherheitskonzept flexibel auf sich ändernde Geschäftsprozesse angepasst werden kann. Dedizierte Administrationskonzepte und Mandantenfähigkeit sind analog zu herkömmlichen Systemen zu erfüllen.

Eine weitere Forderung in diesem Kontext ist die Integrationsfähigkeit der Sicherheitslösung in bestehende Sicherheitslösungen wie bspw. PKI, Meta Directories, Identity Management u.v.m. Eine Integration schafft neben Kostenersparnissen die Grundlage für einen reibungslosen Prozessablauf ohne Medienbrüche sowie die Sicherstellung konsistenter Informationen in allen beteiligten (Sicherheits-) Systemen.

1.3 Sicherheitsanforderungen in Service-Orientierten Architekturen

Anforderungen an Authentifizierung

In SOA Umgebungen trifft man – aufgrund der losen Kopplung – auf eine Vielzahl von Authentifizierungspunkten. Ein kritischer Geschäftsprozess erfordert die Authentizität sämtlicher Service-Aufrufe, beispielsweise innerhalb einer Supply Chain. In einer losen Kopplung sind zudem Fälle zu realisieren, in denen sich Service Consumer gegenüber bislang unbekannten Service Providern authentisieren müssen. Diese Federationkonzepte technisch abzubilden ist ein nicht zu vernachlässigender Punkt. Zudem muss eine Interoperabilität von Authentifizierungsangaben und –formaten gewährleistet sein. Hinzu kommt die Authentifizierung an Legacy Systemen, um diese in die SOA integrieren zu können.

Anforderungen an Autorisierung

Die Administration muss in der Lage sein, ein zentrales Management für Rechte und Rollen von verschiedenen Systemen bereitzustellen. Zudem wird ein Service-Autorisierungssystem benötigt, welches den Zugriff von Service Consumern auf den Web Service regelt und durchsetzt. Das Authentifizierungssystem muss Regeln und Rollen für sämtliche Services an den verschiedenen Systemen und Plattformen umsetzen.

Anforderungen an Integrität

In Service-Orientierten Architekturen werden Transaktionen und entfernte Funktionsaufrufe (RPC, Remote Procedure Calls) verstärkt über Unternehmensgrenzen hinweg realisiert. Daher kann man nicht mehr, wie dies häufig geschieht, ein als sicher betrachtetes LAN als Basis für die Kommunikation voraussetzen.

Eine Realisierung von SOA durch Web Services auf Basis offener Standards ermöglicht eine Kommunikation auf Nachrichtenebene (Message-Based). Daraus resultiert eine einfache Manipulierbarkeit für Angreifer. Eine Sicherung und Prüfbarkeit der Integrität auf Nachrichtenebene muss daher gewährleistet sein. Bei der Verwendung einer SOA muss bei der Integrität zudem beachtet werden, dass in vielen Fällen die Integrität auch auf definierte Teile der Nachricht gewährleistet werden muss. Dies ist erforderlich, da bei einem realen Workflow mehrere Knoten bzw. Teilnehmer durchlaufen werden, die jeweils prozessgesteuerte Transformation des Inhalts der Nachricht durchführen. Beispielsweise sind Kreditkarteninformationen nur definierten Teilnehmern zugänglich. Die Integritätsprüfung muss dies beachten und darf nur die relevanten Teile der Nachricht einzeln berücksichtigen. Die Integrität der Daten muss durch Technologien und Standards aus dem Web Service Bereich[1] abgesichert werden.

Anforderungen an Vertraulichkeit und Datenschutz

Die Kommunikation ist auf Nachrichtenebene zu schützen. Die Nachricht selbst muss den Sicherheitskontext über alle Stationen beibehalten. Die Sicherung einzelner Elemente der Nachricht ist in einem realen Szenario unabdingbar. Als Beispiel sei eine Bestellung genannt, in der neben Artikelstammdaten weitere Angaben zum Kreditrahmen des Kunden zu finden sind. Unterschiedliche Emp-

[1] Siehe Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)

fänger sollen dabei Zugang zu den für sie bestimmten Informationen erhalten. Auch aus Datenschutzgründen ist eine Sicherung von kundenbezogenen Daten notwendig.

Anforderungen an Verbindlichkeit (Nichtabstreitbarkeit)

Verbindlichkeit ist eine weitere Bedingung. Gerade bei der Involvierung verschiedener und teilweise unternehmensfremder Teilnehmer ist die Nichtabstreitbarkeit von getätigten Aktionen (Bestellungen, Aufträgen etc.) eine wesentliche Bedingung. Rechtsgültigkeit wird durch die Konformität der im Signaturgesetz verankerten Rahmenbedingungen zur Nutzung digitaler Signaturen erreicht. So ist beispielsweise die Signatur mit qualifizierten Zertifikaten eine Grundvoraussetzung zur gesetzeskonformen Sicherung der Verbindlichkeit.

Weitere Sicherheitsanforderungen

In einem verteilten System ist die Verfügbarkeit eine weitere Forderung. Ein Geschäftsprozessverlauf, der viele lose gekoppelte Services nutzt, ist dementsprechend von der Verfügbarkeit aller Services auf dem kritischen Pfad abhängig. Neben dem Einsatz bewährter Konzepte (Load Balancing, Application Server Management, etc.) muss in jedem Fall ein Single-Point-of-Failure vermieden werden. Dies gilt in gleicher Weise für alle Security Komponenten. Ein Geschäftsprozess, der durch eine SOA abgebildet wurde, könnte nicht mehr ausgeführt werden, wenn die Security Komponente (z. B. Authentifizierung) nicht mehr verfügbar ist. So sind zentrale Systeme, die Security Infrastrukturen online zur Verfügung stellen, anfällig gegen einfache DoS-Attacken. Die Architektur der Security Systeme ist daher von zentraler Bedeutung.

Die Umsetzung von Audits und Revisionssicherheit in einer verteilten Umgebung ist um ein Vielfaches komplexer. Lösungen müssen in der Lage sein, transaktionsgenau Informationen an einer zentralen Stelle und in einem einheitlichen Format zur Verfügung zu stellen. Ebenso ist die Granularität und Vertraulichkeit zu beachten. Werden kryptografische Methoden eingesetzt, muss das dahinterliegende Key Management Revisionsansprüchen genügen, auch müssen Vertrauensbeziehungen nachvollziehbar bleiben und der Zugriff auf geschützte Daten über die gesamte Lebenszeit der Information gewährleistet sein. Ein zentrales Logging und Reporting muss selbstverständlich ebenso erfüllt werden.

1.4 Grundlage der Betrachtungen in diesem Dokument

Im Folgenden werden die Grundlagen für die Betrachtungen in diesem Dokument kurz beschrieben.

Datentypen

Bei der Betrachtung von Sicherheitsanforderungen sind zwei Typen von Daten zu betrachten:

1. "Daten in Ruhe", die gespeichert sind, also zum Beispiel in Datenbanken, Directories, File Servern oder in sonstigen Repositories abgespeicherte Daten.
2. "Daten in Bewegung", die über Kommunikationskanäle übermittelt werden.

In dieser Studie werden ausschließlich "Daten in Bewegung" betrachtet, da

- im Bereich der Service Orientierten Architekturen die Kommunikation im Vergleich zu herkömmlichen Anwendungen nicht vornehmlich über ununterbrochen abgesicherte Kommunikationskanäle realisiert werden und daher neue Fragestellungen zu betrachten sind.
- bei gespeicherten Daten nach wie vor die bereits umgesetzten und bekannten Sicherheitsmaßnahmen (z.B. Einschränkung des Benutzerkreises, personenbezogene Authentifizierung, Integritätschecker) für die Schutzziele gelten.

Die Nachrichten zwischen Service Consumer und Service Provider sind über alle Zwischenstationen hinweg zu schützen. Die Nachrichten können dabei einen reinen Datenaustausch darstellen (document-style messaging), aber auch dazu führen, dass der Service Provider Funktionen ausführt oder Transaktionen auslöst (RPC-style messaging). Bei entsprechendem Schutzbedarf muss sowohl der Request des Service Consumer als auch die Response des Service Provider vor Manipulationen geschützt werden. Des Weiteren muss die Integrität bei der Veröffentlichung eines Web Services in einem Service Repository und bei der Suche nach einem Web Service in einem Service Repository gegen Manipulationen abgesichert werden.

Dezentrale Umsetzung von Sicherheit

Service-orientierte Systeme erfordern – aufgrund ihrer dezentralen und verteilten Natur – eine ebenso dezentrale Um- und Durchsetzung von Sicherheit. Die Grundlage für eine solche Umsetzung ist die Bereitstellung von Sicherheitsservices („Security as a Service“), die die benötigten Informationen und Funktionalitäten zentral bereitstellen.

Die Umsetzung der Sicherheit erfolgt bei jedem involvierten Service individuell, basierend auf den in Sicherheitspolicies deklarativ spezifizierten Sicherheitsanforderungen. In Policies werden die notwendigen Sicherheitsrichtlinien formal beschrieben, die von den Services in einer Service-orientierten Architektur genutzt werden, um kontextabhängig die entsprechenden Sicherheitsmechanismen anzuwenden. Die zugrundeliegenden Konzepte sind im Kapitel „Policy Management“ des SOA-Security-Kompendiums[2] umfassend beschrieben. Im Rahmen des Policy Managements können folgende Phasen unterschieden werden:

1. Policy Definition
2. Verteilung von Policies (Policy Deployment)
3. Aushandeln von Sicherheitspolicies – Sicheres Late Service Binding
4. Durchsetzung von Policies (Policy Enforcement)
5. Policy Monitoring

Wie im Kompendium beschrieben, kommen in den o.g. unterschiedlichen Phasen bestimmte Basiskomponenten zum Einsatz, die im Zusammenspiel für die Umsetzung der Policies verantwortlich sind. Abbildung 1 zeigt die relevanten Basiskomponenten und deren Abhängigkeiten untereinander.

[2] SOA-Security-Kompendium, Sicherheit in Service Orientierten Architekturen, Version 2.0, Bundesamt für Sicherheit in der Informationstechnik (BSI)

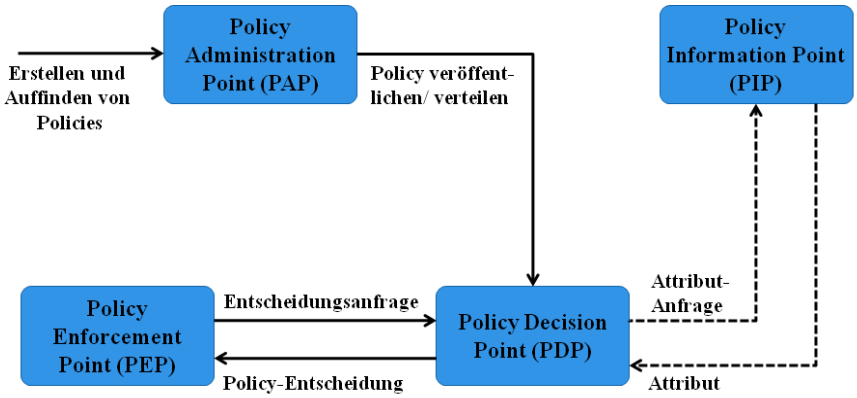


Abbildung 1: Basiskomponenten für das Policy Management

Folgende Komponenten können unterschieden werden (Auszug aus dem Kompendium):

- Im Policy Administration Point (PAP) (u.a. auch als Policy Management Point bekannt) werden Policies erstellt, so dass abstrakte Beschreibungen von geschäftlichen oder rechtlichen Anforderungen vorliegen.
- Der Policy Decision Point (PDP) besitzt die Aufgabe auf Grundlage von Policies Entscheidungen zu treffen (beispielsweise Zugriffsentscheidungen im Rahmen der Autorisierung). Die Policies erhält der PDP vom Policy Adminstration Point, der für die Verteilung der Policies verantwortlich ist.
- Werden zu einem Zeitpunkt für eine Entscheidung bestimmte Informationen benötigt, wird der Policy Information Point (PIP) (z.B. LDAP-Verzeichnis) von dem Policy Decision Point kontaktiert.
- Der Policy Enforcement Point (PEP) ist für die Durchsetzung von Policies verantwortlich. D.h. der PEP stellt technisch z.B. sicher, dass, je nach Entscheidung des Policy Decision Points, ein Zugriff auf eine Unternehmensressource (Web Service, Datenbank, Anwendung...) erfolgen kann oder der Zugriff verweigert wird.

Die in diesem Dokument betrachtenden Architekturen und Entwurfsmuster abstrahieren von diesen umsetzungstechnischen Details. Generell wird nur von Sicherheitsservices (Security Services) gesprochen, die von Services oder Dienstnutzern eingebunden werden. Dieses Modell setzt implizit voraus, dass jeder Akteur durch einen Policy Enforcement Point geschützt ist und – sofern dargestellt – spezifische Sicherheitsservices nutzt. Diese Sicherheitsservices können bei der Autorisierung beispielsweise einen Policy Decision Point darstellen.

1.5 Aufbau des Dokuments

Das Dokument beschreibt zunächst in Kapitel 2 eine Vorgehensweise, durch die der Leser für seinen Anwendungsfall die für ihn passenden Sicherheitsmechanismen identifizieren kann. Danach werden in Kapitel 3 die Architekturelemente eingeführt und kurz beschrieben, mit denen in diesem Dokument die Architekturen aufgebaut werden. Danach schließt sich in Kapitel 4 die Beschreibung der Anwendungsfälle an, die verwendet werden. In den Kapiteln 5 bis 9 werden dann ausgehend von den Architekturen und Anwendungsfällen die Entwurfsmuster für die Schutzziele Authentizität, Integrität, Vertraulichkeit,

Nichtabstreitbarkeit und Autorisierung entwickelt. Zudem werden Technologien und Methoden ausgewiesen, mit denen man die Entwurfsmuster realisieren kann. Das Dokument wendet die entwickelten Entwurfsmuster in Kapitel 10 auf ein Beispiel an und führt die in Kapitel 2 definierte Vorgehensweise beispielhaft vor. Abgeschlossen wird das Dokument mit einem Anhang, in dem die wesentlichen Standards für den Bereich SOA Security aufgelistet und kurz beschrieben werden.

2. Wie wird dieser Entwurfsmusterkatalog verwendet?

2.1 Ansatz

Ausgangspunkt für die Empfehlung von Sicherheitsmaßnahmen bildet die Definition der zu betrachtenden serviceorientierten Szenarios. Dies umfasst zum einen die Anwendungsfälle hinsichtlich Sicherheit sowie die zugrundeliegende Service-basierte Architektur, die diese Prozesse umsetzt. Diese Architektur wird charakterisiert durch die vorhandenen Services, Sicherheitsservices, Service Consumer, Service Provider und deren Beziehungen. Die Auswahl der Architektur und des Anwendungsfalls dienen als Grundlage für die Auswahl der Entwurfsmuster für SOA-Sicherheit, wie in Abbildung 1 gezeigt.

Für die verschiedenen Kombinationen von Anwendungsfällen und Architekturen werden spezifische Entwurfsmuster empfohlen. Die Entwurfsmuster für SOA-Sicherheit ermöglichen die Abbildung der Sicherheitsziele auf konkrete Technologien. Dies resultiert in einem Katalog von möglichen Maßnahmen für die Services und Service Consumer in einer SOA. Dadurch ergibt sich eine Empfehlung für mögliche Sicherheitsmaßnahmen im spezifizierten SOA-Szenario.

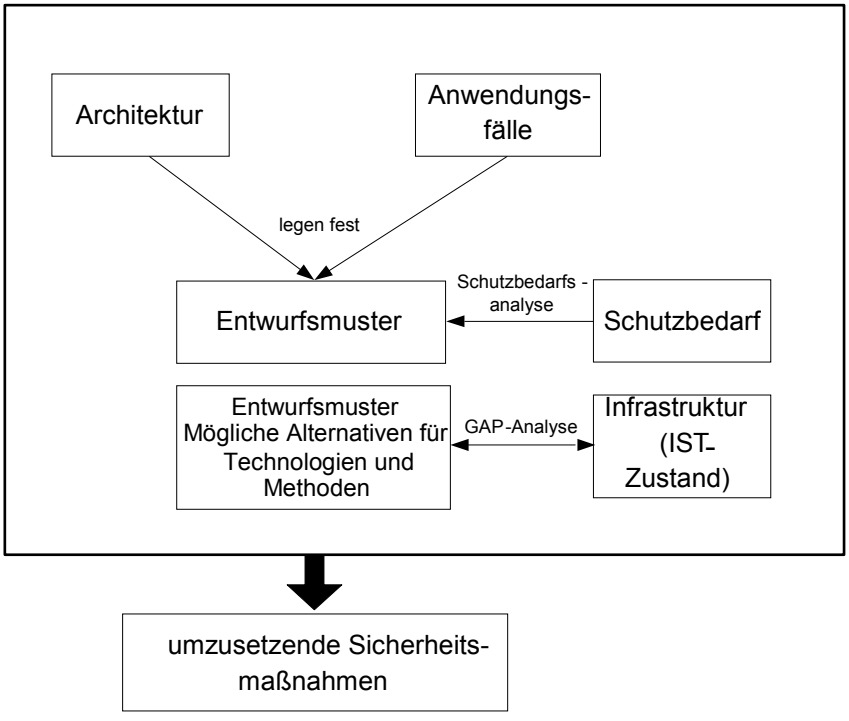
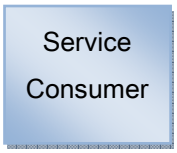


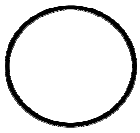
Abbildung 2: Elemente des Modells des Best Practise Ansatzes für SOA

Durch die Elemente Architektur und Anwendungsfall werden die Entwurfsmuster bestimmt, die für den betrachteten Fall aus Sicht der Informationssicherheit relevant sind. Die Entwurfsmuster geben einen Satz von alternativen Technologien an, mit denen sie realisiert werden können und die insbesondere für einen unterschiedlichen Schutzbedarf geeignet sind. Daher ist es notwendig eine Schutzbedarfsanalyse für das betrachtete Szenario durchzuführen, d.h. den Schutzbedarf der verarbeiteten Daten zu identifizieren. Anschließend können alle Technologien des Entwurfsmusters, die dem geforderten Schutzbedarf nicht genügen, ausgeschlossen werden. Im Allgemeinen kann aber an diesem Punkt noch zwischen Alternativen hinsichtlich möglicher Technologien gewählt werden. Die möglichen Alternativen können jedoch priorisiert werden, wenn man die bereits implementierte Infrastruktur berücksichtigt. Der IST-Zustand der bestehenden (oder geplanten) Infrastruktur muss also mit den durch die Entwurfsmuster und den Schutzbedarf geforderten Technologien (SOLL-Zustand) verglichen werden. Als Ergebnis erhält man die noch umzusetzenden Sicherheitsmaßnahmen. Die Sicherheitsmaßnahmen sind zu favorisieren, die mit der bestehenden Infrastruktur bzgl. Kosten und Aufwand günstig realisiert werden können.

2.2 Architekturbausteine

Das Element Architektur arbeitet mit den folgenden Architektur-Bausteinen:

Service Consumer	Nutzer eines Service Der Service Consumer ist eine Applikation, Komponente oder Software Modul die (oder das) einen anderen Service (→ Service Provider) nutzen möchte. Der Service Consumer findet seinen nutzbaren Service in einem Service Repository und sendet den Request an den Service Provider. Im Allgemeinen ist der Service Consumer keine Person sondern eine Applikation oder ein Service. Ein Benutzer startet einen Service Request in der Regel über eine Web-Applikation, die den Service dann aufruft.	
Service Provider	Anbieter eines Service Der Service Provider bietet einen Service an, der über ein Netzwerk zugreifbar ist, und von einem Service Consumer genutzt werden kann. Der Service Provider veröffentlicht seinen angebotenen Service über ein Service Repository.	
Security Provider	Anbieter von Security Verwaltet und steuert die Security Credentials einer Entität (Person, Service, usw.). Security Credentials umfassen alle Sicherheits-	

	Attribute von Entitäten. So können zum Beispiel durch einen Identity Provider digitale Identitäten von der Erstellung bis zur Bereitstellung dieser Identitäten verwaltet werden. Identity Provider sind i.A. Directory Services oder Active Directory. Auch ein Secure Token Service (STS) wird hier als ein Security Provider angesehen. Folgende Credentials werden oft verwendet: Username/Passwort Secure Tokens (One Time Keys) Zertifikate (X.509, Kerberos) Infocards SAML	
Zwischenstation	Eine Zwischenstation vermittelt eine Kommunikation oder ein Credential und liegt nicht im Administrationsbereich des Service Consumers oder Service Providers. Eine Zwischenstation kann ein technischer Service und/oder ein organisatorischer Prozess (mit der Einbeziehung von Personen) sein.	
Vertrauensbeziehung	Service Provider und Service Consumer können Vertrauensbeziehungen aufbauen Wenn ein Service Provider einem Service Consumer als Trusted System bekannt ist, ist z.B. keine Authentifizierung notwendig. In dieser Arbeit gehen wir von beidseitigen Vertrauensbeziehungen aus. Jeder Service ist damit ein Trusted System zum anderen.	
Domain	Eine Domain ist ein zusammenhängender Teilbereich des hierarchischen Domain Name System (DNS). In dieser Arbeit gehen wir von einem durchgängigen Sicherheitskonzept innerhalb einer Domain aus. Ein Security Provider soll nur innerhalb einer Domain wirken.	
Kommunikation	Informationsaustausch zwischen Sender und Empfänger über ein Netzwerk Der Pfeil für die Kommunikation zeigt die Kommunikationsrichtung an.	

Eine Architektur wird durch die Architektur-Bausteine und die Kommunikations- und Vertrauensbeziehungen zwischen diesen beschrieben.

Ein Bild einer Architektur wird in Abbildung 3 gezeigt:

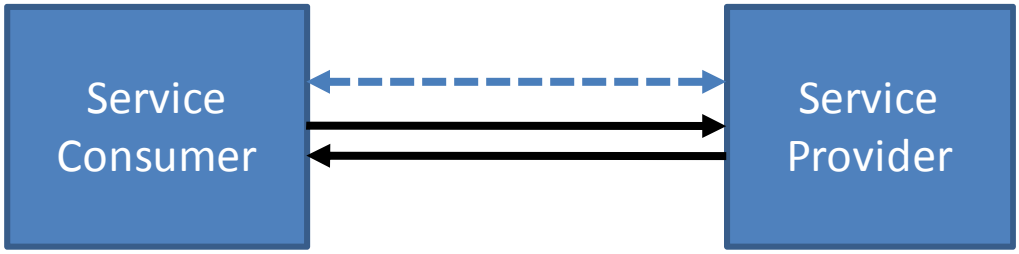


Abbildung 3: Beispiel für eine Architektur

In Abbildung 3 wird eine Kommunikation zwischen Service Consumer und Service Provider mit Request und Response gezeigt. Weiter besteht eine beidseitige Vertrauensbeziehung zwischen Service Provider und Service Consumer.

2.3 Anwendungsfälle

Die Anwendungsfälle geben auf sehr abstraktem Niveau die Zielstellung und Rahmenbedingungen vor. Beispielsweise wird beschrieben, welche Sicherheitsziele zu erfüllen sind und welche Informationen schützenswert sind. Die im hier beschriebenen Best Practise Model verwendeten Anwendungsfälle werden für die verschiedenen Schutzziele Vertraulichkeit, Integrität, Authentifizierung, Autorisierung und Nichtabstreitbarkeit in Abschnitt 4 aufgelistet.

Authentifizierung - Wie sind die Benutzer einem Service bekannt? / Wer soll einen Service benutzen dürfen?	
Anmelden eines Service Consumers bei einem Service Provider (Authentifizierung bei einem Web Service)	Auth_U2

Tabelle 1: Beispiel eines Anwendungsfalls hinsichtlich der Authentifizierung

Tabelle 1 listet beispielhaft einen Anwendungsfall auf. Ein Service Consumer möchte einen Service nutzen. Dazu ist ein Login bei einem Service Provider erforderlich.

2.4 Auswahl der Entwurfsmuster

Für die verschiedenen Architekturen und Anwendungsfälle werden für die genannten Schutzziele die möglichen Entwurfsmuster in einer Kreuztabelle angegeben. Somit kann der Leser für seine Situation zunächst die Architektur und den Anwendungsfall identifizieren und dann in der Kreuztabelle die möglichen Entwurfsmuster ablesen.

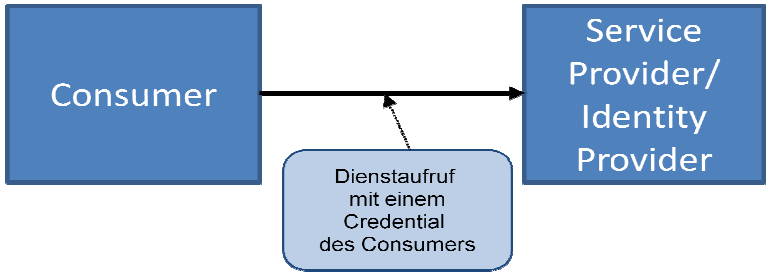
2.5 Entwurfsmuster

Ein Entwurfsmuster ist durch folgende Attribute beschrieben:

- Beschreibung
- Beispielszenarien
- Beispiele für Technologien

Ein Beispiel für ein Entwurfsmuster für das Sicherheitsziel Authentifizierung ist in

Abbildung 4 gezeigt:

<table>
<tr><td>Muster: Direkte Authentifizierung</td><td>P1</td></tr>
<tr><td>Beschreibung</td><td>

Der Service Consumer verwendet ein Credential mit einer Aussage über sich selbst (z.B. ich bin Benutzer X) und einer Authentifizierungsinformation. Der Service Provider überprüft das Credential selbst, um den Benutzer zu authentisieren.</td></tr>
<tr><td>Beispielszenarien</td><td>Authentifizierung eines Nutzers bei einer Webseite oder einem Service.</td></tr>
<tr><td>Beispiele für Technologien</td><td><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Methoden</th></tr>
<tr><td>Authentifizierung des Benutzers:</td><td>Benutzername/Passwort,…</td></tr>
</table></td></tr>
</table>

Abbildung 4: Beispiel für ein Entwurfsmuster für das Schutzziel Authentifizierung

Die Beschreibung ergänzt das Architekturbild in Bezug auf die Informationssicherheit und liefert eine Lösung für die durch den Anwendungsfall beschriebene Problemstellung. Es wird angegeben, welcher Sicherheitsaspekt von Bedeutung ist und wie dieser im Rahmen der Architektur umzusetzen ist. Im Beispiel wird ein selbstausgestelltes Credential durch den Service Consumer an den Service Provider zum Zwecke der Authentifizierung gesendet. Ein selbstausgestelltes Credential unterscheidet sich von einem Credential, welches durch eine dritte Partei ausgestellt und signiert wurde und resultiert somit in unterschiedlichen Technologieempfehlungen.

Die Beispielszenarien beschreiben kurz (in wenigen Sätzen), wie zugehörige Anwendungsszenarien in der Praxis üblicherweise ausgestaltet sind.

Die Beispiele unter "Mögliche Technologien/Methoden" benennen die Technologien und Methoden, mit denen der in der Beschreibung geforderte Sicherheitsaspekt für das Entwurfsmuster realisiert werden kann.

2.6 Schutzbedarf

Die Definition der Schutzbedarfsklassen lehnt sich an die Definition der Schutzbedarfskategorien des BSI-Standard 100-2: IT-Grundschutz-Vorgehensweise an. Folgende Schutzbedarfsklassen werden also verwendet:

Schutzbedarfsklasse	Kurzbeschreibung
Normal	Die Schadensauswirkungen sind begrenzt und überschaubar.
Hoch	Die Schadensauswirkungen können beträchtlich sein.
Sehr hoch	Die Schadensauswirkungen können ein existentiell bedrohliches, katastrophales Ausmaß erreichen.

Tabelle 2: Quelle: BSI-Standard 100-2: IT-Grundschutz-Vorgehensweise

Der Schutzbedarf ist mindestens für die Schutzziele Vertraulichkeit und Integrität zu ermitteln.

2.7 Auswahl der Technologien und Methoden

Nach der Bestimmung der Architektur, des Anwendungsfalls, des Entwurfsmusters und des Schutzbedarfs werden die Technologien und Methoden, die gemäß Entwurfsmuster möglich sind ausgewählt und gemäß Schutzbedarf priorisiert.

2.8 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur

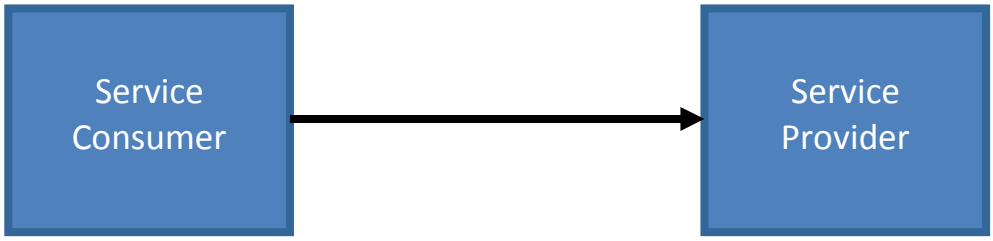
Hier wird die IST-Situation (oder die geplante) der vorhandenen Technologien und Methoden der bestehenden Infrastruktur beschrieben. Die in Abschnitt 0 ausgewählten Technologien und Methoden können gemäß der IST-Situation weiter priorisiert werden. Ein Beispiel hierzu: Wird z.B. eine 2-Faktor-Authentifizierung verlangt und besteht bereits eine Infrastruktur z.B. mit RSA Token, so kann diese natürlich verwendet werden. Ist hingegen keine Infrastruktur für eine 2-Faktor-Authentifizierung vorhanden, so kann man zwischen den Alternativen wie z.B. RSA Token oder Zertifikat wählen und muss diese Infrastruktur dann natürlich erst implementieren und betreiben.

3. *Architekturbetrachtungen für Service-Orientierte Architekturen*

Die Architekturen werden durch Beziehungen zwischen den in Abschnitt 0 aufgeführten Architekturbausteinen aufgebaut. Dabei werden Architekturen hinsichtlich des Nachrichtenaustausches und der Vertrauensbeziehung unterschieden. Die sich ergebenden Architekturen können miteinander kombiniert werden z.B. die Architektur A2 mit der Architektur A5. Dann ergibt sich eine Kommunikation mit Request und Response zwischen Service Consumer und Service Provider mit beidseitiger Vertrauensbeziehung.

3.1 Nachrichtenaustausch

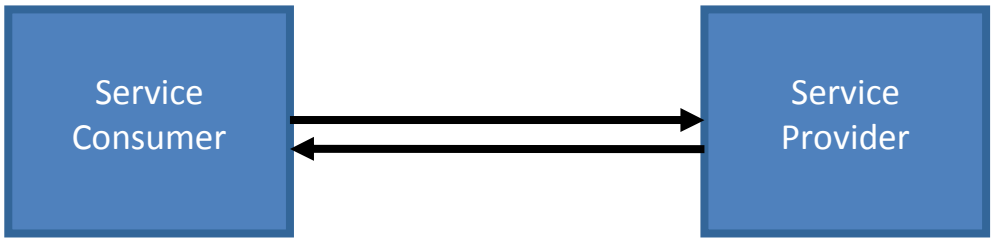
Direkter Service Request ohne Response	A1



Charakteristik:

Diese Architektur liegt vor, wenn der Service Consumer einen Request an den Service Provider sendet.

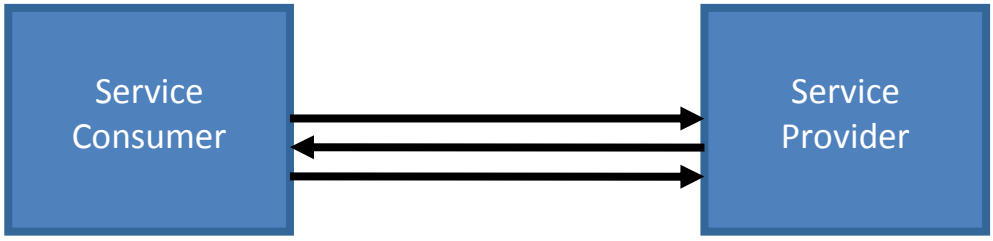
Direkter Service Request mit Response	A2



Charakteristik:

Diese Architektur liegt vor, wenn der Service Consumer einen Request an den Service Provider sendet. Der Service Provider antwortet mit einer Response.

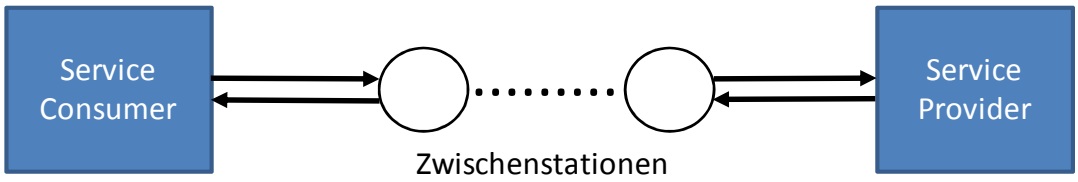
Direkter Service Request mit Response und Bestätigung (zuverlässiger Nachrichtenaustausch)	A3



Charakteristik:

Diese Architektur liegt vor, wenn der Service Consumer einen Request an den Service Provider sendet. Der Service Provider antwortet mit einer Response. Der Service Consumer bestätigt die Response.

Indirekter Service Request mit Response über Zwischenstationen	A4

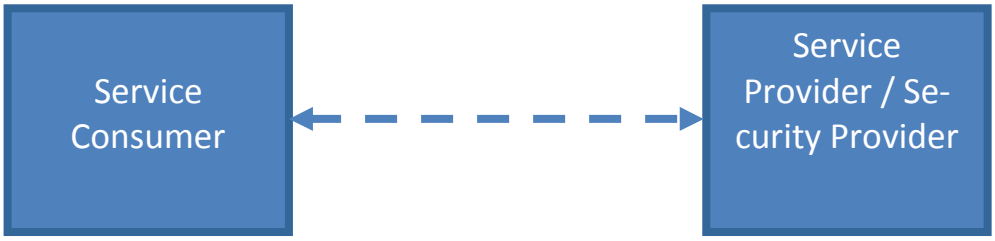


Charakteristik:

Diese Architektur liegt vor, wenn der Service Consumer mit einem Service Provider über mehrere Zwischenstationen kommuniziert. Für die direkte Kommunikation mit/zwischen den Zwischenstationen können die Entwurfsmuster für die direkte Kommunikation A1, A2 oder A3 genutzt werden.

3.2 Vertrauensbeziehungen und Integration von Security Providern

Direktes Vertrauensverhältnis	A5



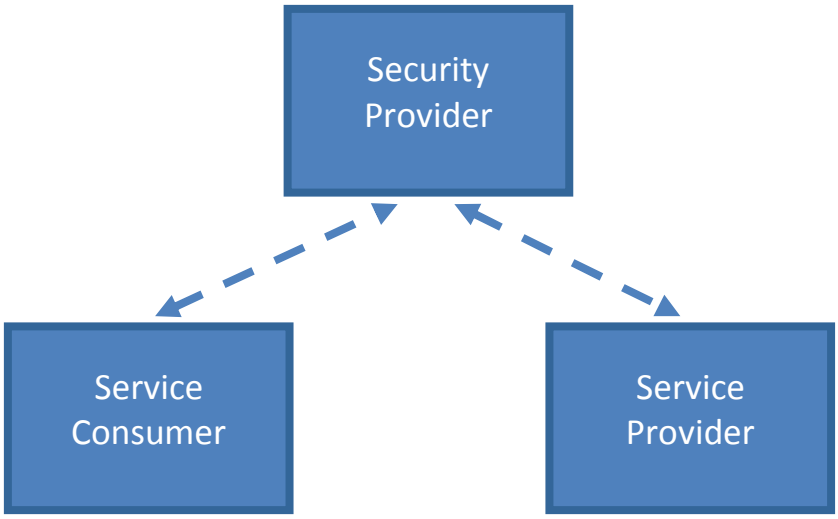
Charakteristik:

Es besteht eine beidseitige Vertrauensbeziehung zwischen einem Service Consumer und einem Service Provider aufgrund

1. eines gemeinsamen Key Managements (z.B. preshared keys, Aushandlung eines symmetrischen Schlüssels, bei asymmetrischem Verfahren ein gemeinsames Root Zertifikat).
2. einer Service-seitigen Verwaltung von Credentials. Beispielsweise können Benutzer einer Anwendung/ eines Services durch diese/diesen eigenständig verwaltet werden, z.B. in der Datenbank der Anwendung. Somit ist jeder Service-Provider auch Security-Provider. Dies setzt eine initiale Registrierung des Benutzers voraus, so dass ein beidseitiges Vertrauensverhältnis entsteht.

Als Kommunikationsmuster können die Architekturen A1, A2, A3 oder A4 zum Einsatz kommen.

Indirektes Vertrauensverhältnis mittels Security Provider	A6



Charakteristik:

Diese Architektur liegt vor, wenn die Credentials und/oder Attribute (z.B. Rechte) eines Dienstnutzers einer Anwendung/ eines Services durch eine dritte eigenständige Partei verwaltet werden, z.B. in einem unternehmensweiten LDAP oder bei einem föderiertem Service Provider. In diesem Fall stehen die verwalteten Attribute mehreren Services / Anwendungen zur Verfügung.

4. Anwendungsfälle für Service-Orientierte Architekturen

Dieses Kapitel beschreibt gängige Anwendungsfälle für die verschiedenen Sicherheitsziele, die eine spätere Auswahl der Sicherheitspattern ermöglichen.

4.1 Anwendungsfälle hinsichtlich der Authentifizierung

Authentication - Wie sind die Benutzer einem Service bekannt? / Wer soll einen Service benutzen dürfen?	
Registrieren eines Service Consumer bei einem Service Provider (Beispiele: Bürger/Unternehmen registriert sich bei einer Behörde)	Auth_U1
Anmelden eines Consumers bei einem Provider (Authentifizierung bei einem Web Service)	Auth_U2
Anmelden eines Consumers bei einer Domäne (man möchte verschiedene Services in der gleichen Domäne mit einer Anmeldung benutzen)	Auth_U3
Anmelden eines Consumers bei einem Provider mit SSO bei anderen Providern	Auth_U4

4.2 Anwendungsfälle hinsichtlich der Datensicherheit (Integrität/Vertraulichkeit)

Vertraulichkeit/Integrität	
Ein Service Consumer sendet einen Request an einen Service Provider. Der Service Provider antwortet mit einer Response. Mit dem Response werden Daten übermittelt (document-style messaging).	DS_U1
Ein Service Consumer sendet einen Request an einen Service Provider. Der Service Provider löst eine Transaktion aus und antwortet mit einer Response. Mit der Response werden Statusmeldungen, Fehlermeldungen etc. übermittelt (RPC-style messaging).	DS_U2
Ein Service Provider stellt einen Web Service im Repository ein (Übertragung eines WSDL Files).	DS_U3
Ein Service Consumer ruft Web Service Beschreibungen bei einem Repository ab.	DS_U4

4.3 Anwendungsfälle hinsichtlich der Nichtabstreitbarkeit

Nichtabstreitbarkeit	
Nichtabstreitbarkeit eines Aufrufes durch den Consumer	NA_U1
Nichtabstreitbarkeit eines Aufrufes durch den Provider	NA_U2
Nichtabstreitbarkeit des Inhaltes eines Aufrufes eines Services durch den Consumer	NA_U3
Nichtabstreitbarkeit des Inhaltes eines Aufrufes eines Services durch den Provider	NA_U4

Nichtabstreitbarkeit einer Response durch den Consumer	NA_U5
Nichtabstreitbarkeit einer Response durch den Provider	NA_U6
Nichtabstreitbarkeit des Inhaltes einer Response eines Services durch den Consumer	NA_U7
Nichtabstreitbarkeit des Inhaltes einer Response eines Services durch den Provider	NA_U8

Diese Anwendungsfälle können auch aus einem anderen Blickwinkel betrachtet werden:

Wer soll nicht abstreiten können?	Was soll er nicht abstreiten können?	
Service Consumer	Aufruf	NA_U1
Service Provider	Aufruf	NA_U2
Service Consumer	Aufruf mit Inhalt	NA_U3
Service Provider	Aufruf mit Inhalt	NA_U4
Service Consumer	Response	NA_U5
Service Provider	Response	NA_U6
Service Consumer	Response mit Inhalt	NA_U7
Service Provider	Response mit Inhalt	NA_U8

4.4 Anwendungsfälle hinsichtlich der Autorisierung

Autorisation – Welche Rechte haben die Benutzer eines Service? / Wer darf was in einen Service benutzen?	
Ein Service Consumer ruft einen Service Provider auf. Er erhält nach Prüfung durch den Service Provider das Recht bestimmte Services zu nutzen. (Beispiele: Bürger/Unternehmen-Behörde)	I_U1
Ein Service Consumer ruft einen Service Provider auf. Er übermittelt beim Aufruf zusätzliche Credentials (z.B. bei einem Vier-Augen-Prinzip die Credentials des Vorgesetzen). Er erhält nach Prüfung durch den Provider das Recht bestimmte Services zu nutzen. (Beispiele: Bürger/Unternehmen-Behörde)	I_U2
Ein Service Consumer ruft einen Service Provider auf. Der Provider überprüft und erhält zusätzliche Credentials (z.B. bei einem Vier-Augen-Prinzip die credentials des Vorgesetzen). Er erhält nach Prüfung durch den Provider das Recht bestimmte Services zu nutzen. (Beispiele: Bürger/Unternehmen-Behörde)	I_U3

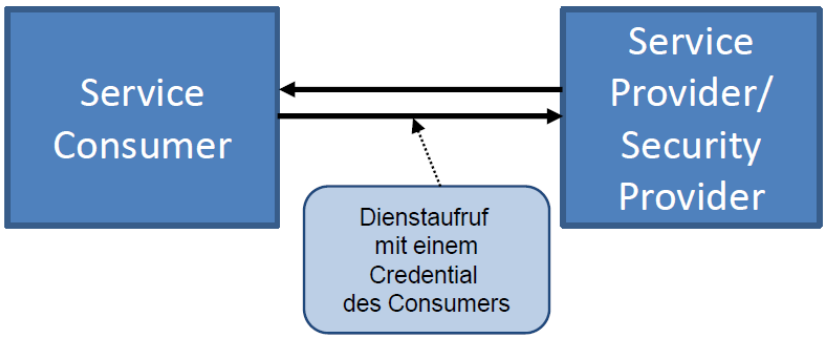
5. *Entwurfsmuster für Authentizität*

Grundlegende Entwurfsmuster zur Umsetzung der Authentifizierung werden in diesem Kapitel beschrieben. Die im Folgenden dargestellte Tabelle ermöglicht eine Zuordnung zwischen den Architekturen und Anwendungsfällen zu möglichen Entwurfsmustern, die zur Umsetzung einer konkreten Problemstellung herangezogen werden können.

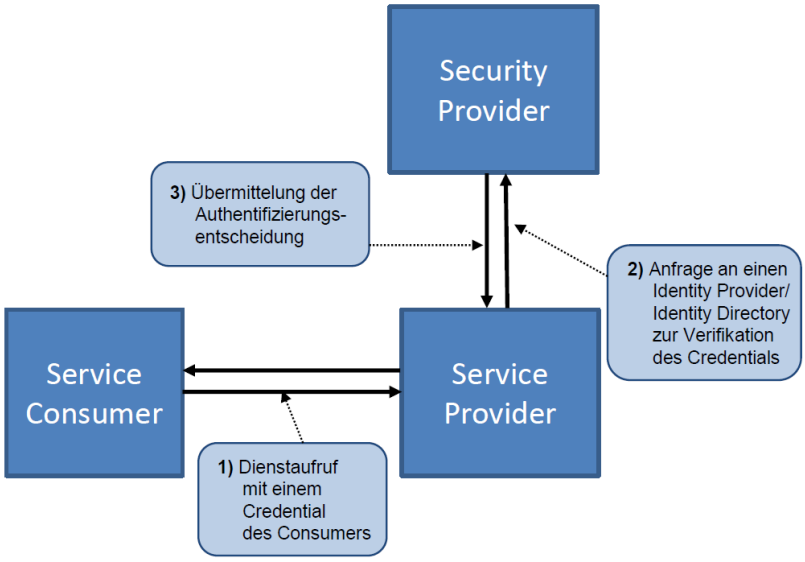
5.1 Auswahl der Entwurfsmuster

	(A5) Direktes Vertrauens-verhältnis	(A6) Indirektes Vertrauens-verhältnis
(Auth_U1) Registrierung	➔ (P1) Direkte Authentifizierung	–
(Auth_U2) einfache Anmeldung	➔ (P1) Direkte Authentifizierung	➔ (P2) Delegierte Service Provider-zentrische Authentifizierung ➔ (P3) Delegierte Service Consumer-zentrische Authentifizierung ➔ (P4) Delegierte Service Consumer-zentrische Authentifizierung mit Rückkopplung
(Auth_U3) domänen-interne Anmeldung	–	➔ (P3) Delegierte Service Consumer-zentrische Authentifizierung ➔ (P4) Delegierte Service Consumer-zentrische Authentifizierung mit Rückkopplung
(Auth_U4) domänen-übergreifende Anmeldung	–	➔ (P5) Föderative Authentifizierung

5.2 Entwurfsmuster

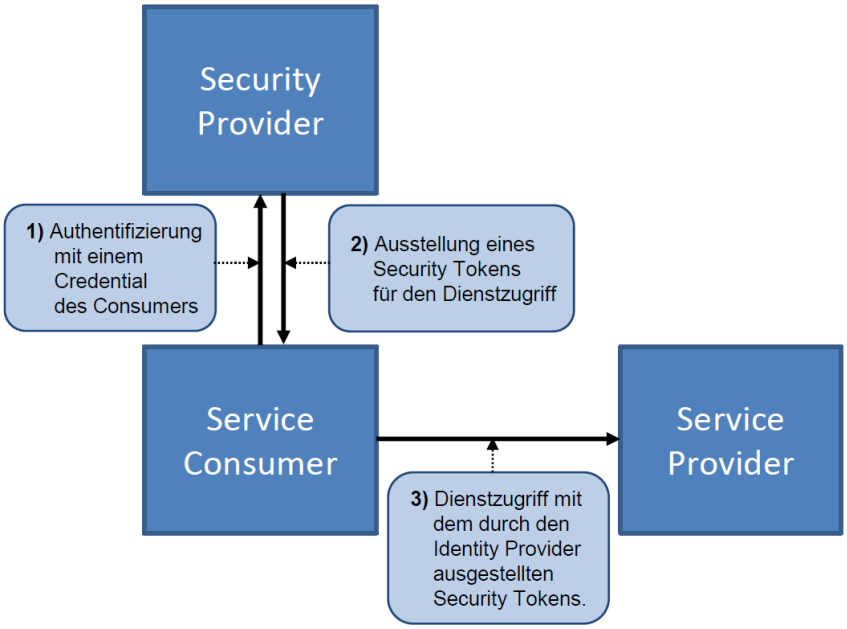
<table>
<tr><th colspan="2">Muster: Direkte Authentifizierung</th><th>P1</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer verwendet ein Credential mit einer Aussage über sich selbst (z.B. ich bin Benutzer X) und einer Authentifizierungsinformation. Der Service Provider überprüft verifiziert das Credential selbst, um den Benutzer zu authentisieren.</td></tr>
<tr><td>Beispielszenarien</td><td colspan="2">Authentifizierung eines Nutzers bei einer Webseite oder einem Service.</td></tr>
<tr><td>Beispiele für Technologien</td><td colspan="2"><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Authentifizierung des Benutzers:</td><td>Benutzername/Passwort,…</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="2">Muster: Delegierte Provider-zentrische Authentifizierung</th><th>P2</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer verwendet ein Credential mit einer Aussage über sich selbst (z.B. ich bin Benutzer X) und einer Authentifizierungsinformation. Der Service Provider kontaktiert einen Security Provider (bspw. Identity Provider/ Identity Directory), um das Credential verifizieren zu lassen.</td></tr>
</table>

<table>
<tr><td>Beispielszenarien</td><td>Authentifizierung eines Nutzers mittels eines unternehmensinternen LDAP.</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Authentifizierung des Benutzers:</td><td>Benutzername/Passwort,…</td></tr>
<tr><td>Umsetzung des Identity Providers/Identity Directory</td><td>LDAP, WS-Trust,…</td></tr>
</table>
</td></tr>
</table>

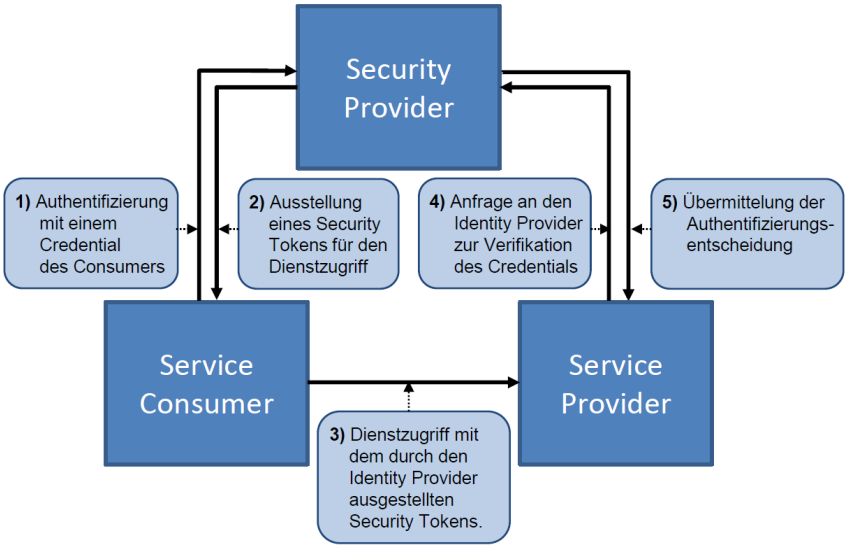
<table>
<tr><th>Muster: Delegierte Consumer-zentrische Authentifizierung</th><th>P3</th></tr>
<tr><td>Beschreibung</td><td>


Zur Authentifizierung beim Service wird ein Credential genutzt, in dem eine dritte (vom Service vertraute) Instanz Aussagen über den Benutzer macht. Alle Services und Nutzer vertrauen dieser Instanz.
Soll der Benutzer beim Service authentifiziert werden, sind folgende Schritte notwendig:
1. Der Service Consumer muss sich durch den Security Provider (Identity Provider) authentifizieren lassen (Verwendung des Musters direkte Authentifizierung).
2. Der Security Provider muss Aussagen über die Identität (Claims) in ein Credential einfügen, mit dem sich der Service Consumer beim Service Provider authentifizieren kann.
3. Der Service Provider muss zum einen die Authentizität des Security Providers prüfen, der das Credential ausgestellt hat und zum anderen die in dem Credential enthaltenen Informationen.
</td></tr>
</table>

<table>
<tr><td>Beispielszenarien</td><td>

Zentrale Benutzerverwaltung im Unternehmen
Bereitstellung zentraler Daten, wie unternehmensweites Adressbuch, etc.
Unternehmensweites SSO für Unternehmensanwendungen
SSO für offene Umgebungen wie das Internet mit mehreren potenziellen Identitätsprovidern (Beispiel InformationCard)

</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Authentifizierung des Benutzers:</td><td>Benutzername/Passwort,…</td></tr>
<tr><td>Bereitstellung der Authentifizierungsentscheidung:</td><td>SAML, Kerberos, …</td></tr>
<tr><td>Abrufen der Authentifizierungstokens:</td><td>WS-Trust, SAML, Kerberos, …</td></tr>
</table>
</td></tr>
</table>

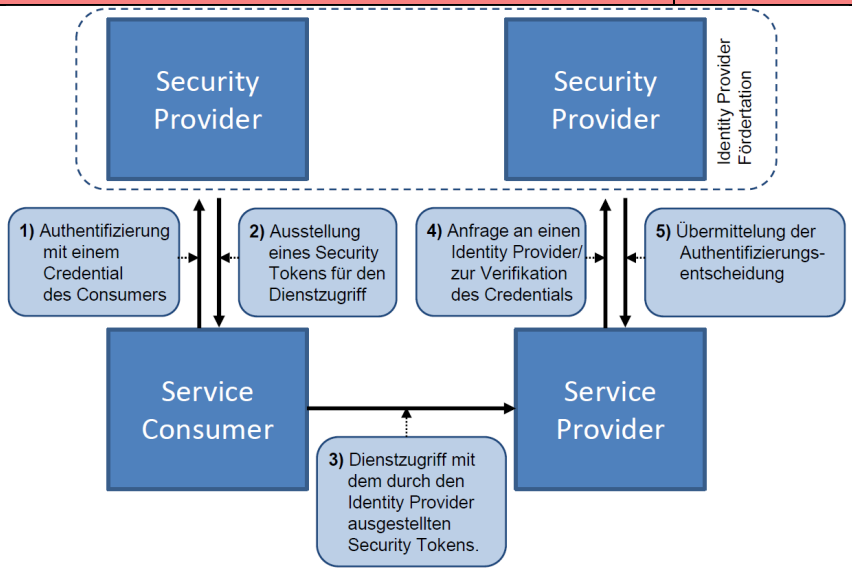
<table>
<tr><th>Muster: Delegierte Consumer-zentrische Authentifizierung mit Rückkopplung</th><th>P4</th></tr>
<tr><td>Beschreibung</td><td>


Dieses Entwurfsmuster erweitert das Entwurfsmuster ‚Delegierte Consumer-zentrische Authentifizierung', um einen zusätzlichen Verifikationsschritt (Schritt 4+5).

Beim Aufruf eines Service wählt der Benutzer von welchem Identitätsprovider die Informationen, die der Service benötigt, kommen sollen. Hierbei kann es mehrere Security Provider geben und der Service entscheidet welchen Security Providern er vertraut.

Soll der Benutzer beim Service authentifiziert werden, sind folgende Schritte notwendig:
1. Der Service Consumer muss sich durch den Security Provider authentifizieren lassen (Verwendung des Musters direkte Au-
</td></tr>
</table>

	thentifizierung). 2. Der Security Provider muss Aussagen über die Identität (Claims) in ein Credential einfügen, mit dem sich der Service Consumer beim Service Provider authentifizieren kann. 3. Der Service Provider verbindet sich mit dem Security Provider, der das Token ausgestellt hat, um das Credential zu verifizieren.
Beispielszenarien	Zentrale Benutzerverwaltung im Unternehmen (Beispiel RSA Token) als auch offene Umgebungen wie das Internet mit mehreren potenziellen Identitätsprovidern (Beispiel OpenID)
Beispiele für Technologien	<table><tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr><tr><td>Authentifizierung des Benutzers:</td><td>Benutzername/Passwort,…</td></tr><tr><td>Bereitstellung der Authentifizierungsentscheidung:</td><td>OpenID, RSAToken, …</td></tr><tr><td>Abrufen der Authentifizierungstokens:</td><td>OpenId, RSAToken, …</td></tr></table>

Muster Föderative Authentifizierung	P5
Beschreibung	 Bei der föderativen Authentifizierung existiert ein Vertrauenszirkel zwischen verschiedenen Security Providern (Identity Providern) aus unterschiedlichen Vertrauensdomains. Förderationsverträge regeln die Vertrauensbeziehung zwischen den Partnern. Alle Security Provider akzeptieren Sicherheitstoken von den Security Providern der Partner.
Einsatzszenarien	Dieses Entwurfsmuster findet Anwendung in B2B Szenarien, in denen zwei Unternehmen bereits jeweils eine eigene Identitätsmanagementinfrastruktur besitzen. Jedes Unternehmen stellt einen eigenen Identitätsprovider bereit. Beispiel: Mitarbeiter des einen Unternehmens möchten auf Ressourcen

<table>
<tr><td></td><td>im Partnerunternehmen zugreifen</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Authentifizierung des Nutzers:</td><td>Benutzername/Passwort,…</td></tr>
<tr><td>Bereitstellung der Authentifizierungsentscheidung:</td><td>SAML, …</td></tr>
<tr><td>Abrufen der Authentifizierungstokens:</td><td>WS-Trust, SAML, …</td></tr>
<tr><td>Etablierung der Föderation:</td><td>WS-Federation, SAML, …</td></tr>
</table>
</td></tr>
</table>

5.3 Auswahl der Technologien und Methoden

Für den Aspekt Authentifizierung des Nutzers gelten die in folgender Tabelle möglichen Technologien und Methoden:

Schutzbedarf	Technologien und Methoden	Erläuterungen
normal	Username- Password, PIN Challenge-Response-Verfahren (z.B. Captcha) Location-based (IP-Range, Geographie) SAML-Token (mit Aussagen über den Nutzer)	
hoch	2-Faktor-Authentifizierung	Challenge-Response-Verfahren (RSA Token …) Zertifikatsbasierte Authentifizierung (softwarebasiert)
Sehr hoch	2-Faktor-Authentifizierung nach gesetzlichen Anforderungen	Zertifikatsbasierte Authentifizierung (Hardware (SmartCard)) neuer Personalausweis: eID Karte

5.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur

Vorhandene Domain-weite Authentifizierungsmechanismen wie z.B. Active Directory, RSA-Tokens usw. sollten natürlich verwendet werden.

Viele Organisationen verwenden eine Microsoft Infrastruktur mit Active Directory und einer Microsoft CA. Ein normaler Schutzbedarf kann mit einer Anmeldung am Active Directory mit Username Passwort abgedeckt werden. Ein hoher Schutzbedarf kann mit einem zertifikatsbasiertem SSL (Microsoft CA) umgesetzt werden. Für sehr hohen Schutzbedarf ist ein zertifiziertes Trustcenter, das Smartcards mit qualifizierten Zertifikaten unterstützt, notwendig.

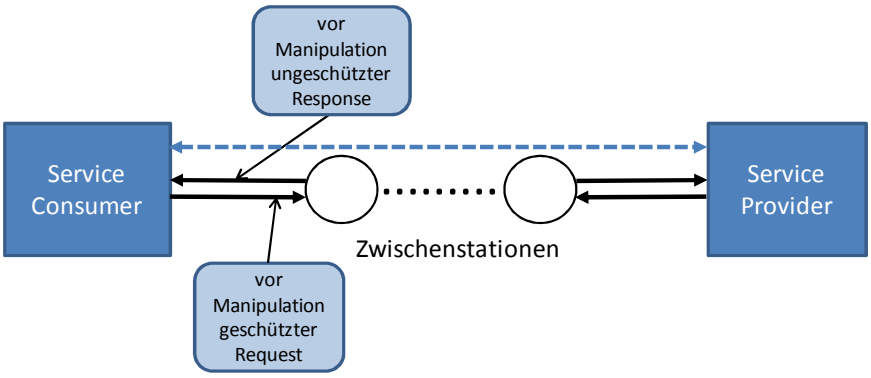
6. *Entwurfsmuster für Integrität*

Grundlegende Entwurfsmuster zur Umsetzung der Integrität werden in diesem Kapitel beschrieben. Die im Folgenden dargestellte Tabelle ermöglicht eine Zuordnung zwischen den Architekturen und Anwendungsfällen zu möglichen Entwurfsmustern, die zur Umsetzung einer konkreten Problemstellung herangezogen werden können.

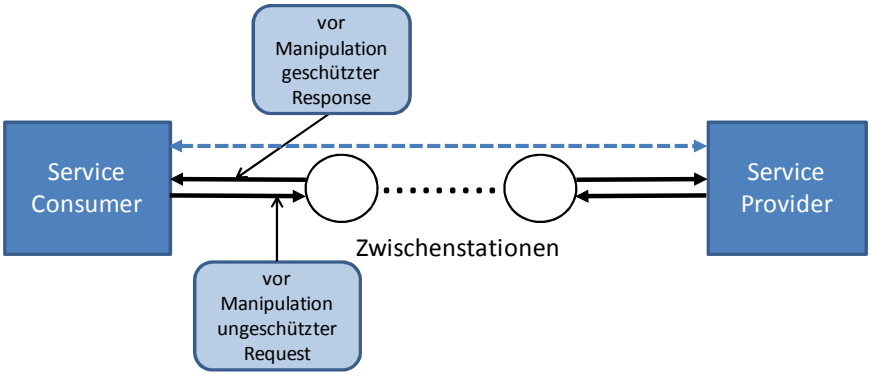
6.1 Auswahl der Entwurfsmuster

	(A2) mit (A5) Direkte Kommunikation beidseitige Vertrauensbeziehung	(A4) mit (A5) Indirekte Kommunikation beidseitige Vertrauensbeziehung
(DS_U1) Datenübermittlung	➜ (P4) Integrität des Kommunikationskanals gewährleistet	➜ (P1) Integrität des Request gewährleistet ➜ (P2) Integrität der Response gewährleistet ➜ (P3) Integrität des Requests und der Response gewährleistet
(DS_U2) Transaktion	➜ (P4) Integrität des Kommunikationskanals gewährleistet	➜ (P1) Integrität des Request gewährleistet ➜ (P2) Integrität der Response gewährleistet ➜ (P3) Integrität des Requests und der Response gewährleistet
(DS_U3) Web Service einstellen	➜ (P4) Integrität des Kommunikationskanals gewährleistet	➜ (P1) Integrität des Request gewährleistet ➜ (P3) Integrität des Requests und der Response gewährleistet
(DS_U4) Web Service abfragen	➜ (P4) Integrität des Kommunikationskanals gewährleistet	➜ (P2) Integrität der Response gewährleistet ➜ (P3) Integrität des Requests und der Response gewährleistet

6.2 Entwurfsmuster

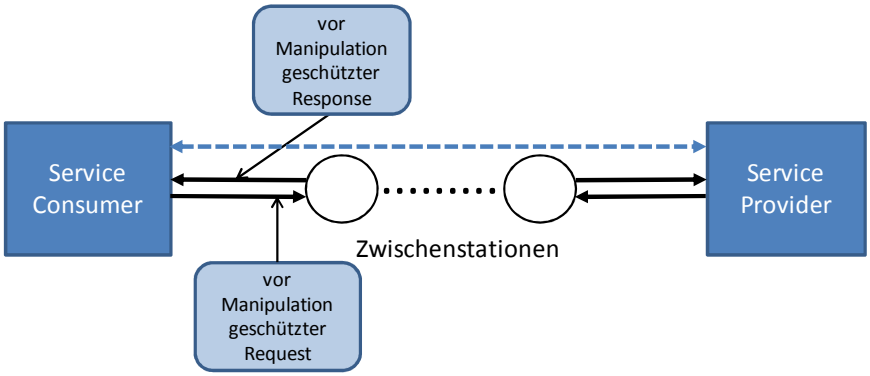
<table>
<tr><th colspan="2">Muster: Integrität des Request gewährleistet</th><th>P1</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer sendet einen Request, der vor einer Manipulation geschützt wird, d.h. der Service Provider kann feststellen, dass eine Manipulation erfolgte.</td></tr>
<tr><td>Beispielszenarien</td><td colspan="2">Ein Service Consumer bestellt eine Ware in einem Online-Shop und erhält eine Auftragsbestätigung. Durch den Request wird eine Transaktion ausgelöst (das Versenden der Ware).
Ein Zulieferer stellt Logistikdaten auf dem Logistik-System des Kunden ein und erhält eine Bestätigung des Eingangs der Daten. Die Kommunikation verläuft über das Internet.</td></tr>
<tr><td>Beispiele für Technologien</td><td colspan="2"><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Integrität des Requests</td><td>digital signierte Message (XML-Signature)</td></tr>
</table></td></tr>
</table>

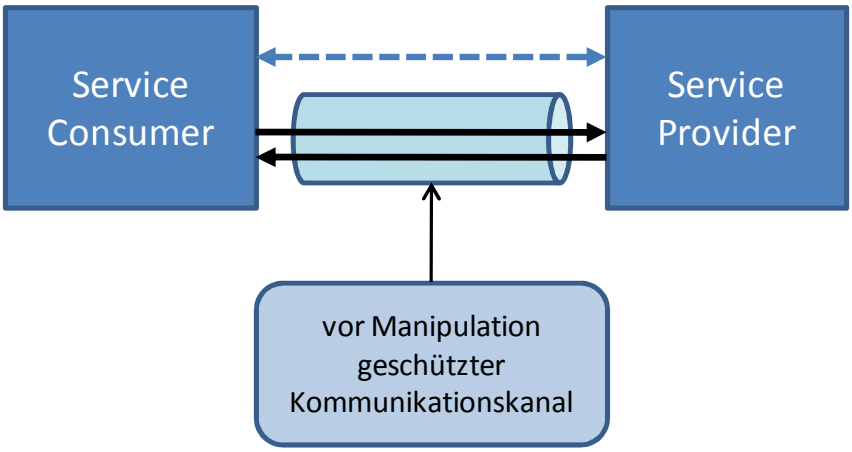
<table>
<tr><th colspan="2">Muster: Integrität der Response gewährleistet</th><th>P2</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer sendet einen Request, der nicht vor einer Manipulation geschützt ist. Der Service Provider empfängt eine Response vom Service Provider und kann feststellen, ob eine Manipulation der Response erfolgte.</td></tr>
</table>

<table>
<tr><td>Beispielszenarien</td><td>Ein Service Consumer fragt Flugdaten bei einem Reisebüro ab. Das Reisebüro als Service Provider übermittelt die Flugdaten.
Ein Service Consumer fragt eine Beschreibung eines Web Services bei einem Service Repository ab. Als Response erhält er das WSDL File oder Fehlermeldungen.</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Integrität der Response</td><td>digital signierte Message (XML Signature)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th colspan="2">Muster: Integrität des Requests und der Response gewährleistet</th><th>P3</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer sendet einen Request, der nicht vor Manipulation geschützt ist. Der Service Provider empfängt eine Response vom Service Provider und kann ebenfalls feststellen, ob eine Manipulation der Response erfolgte.</td></tr>
<tr><td>Beispiel-szenarien</td><td colspan="2">Ein Service Consumer fragt Bonitätsdaten für einen Kunden bei seiner Bank an. Die Bank fragt die Bonitätsdaten bei einem Service Provider an. Der Service Provider übermittelt die Bonitätsprüfung für den im Request benannten Kunden an die Bank. Diese sendet die Daten an den Service Consumer.</td></tr>
<tr><td>Beispiele für Technologien</td><td colspan="2">
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Integrität des Requests / der Response</td><td>digital signierte Message (XML Signature)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th colspan="2">Muster: Integrität des Kommunikationskanals gewährleistet</th><th>P4</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Zwischen Service Consumer und Service Provider wird ein vor Manipulationen geschützter Kommunikationskanal aufgebaut. Bei direkter Kommunikation sind damit sowohl Request als auch Response vor Manipulationen geschützt.</td></tr>
<tr><td>Beispiel-szenarien</td><td colspan="2">Ein Zulieferer stellt Logistikdaten auf dem Logistik-System des Kunden ein und erhält eine Bestätigung des Eingangs der Daten. Die Kommunikation verläuft über eine VPN-Verbindung zwischen Zulieferer und Kunden.</td></tr>
<tr><td>Beispiele für Technologien</td><td colspan="2"><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Integrität des Requests / der Response.</td><td>Geschützter SSL/TLS Kanal</td></tr>
</table></td></tr>
</table>

6.3 Auswahl der Technologien und Methoden

Schutzbedarf	Technologien und Methoden	Erläuterung
normal	SSL/TLS XML Signature	Wenn der Schutzbedarf der Message (Request oder Response) bzgl. Integrität als normal eingestuft ist, so können die Technologien SSL/TLS oder XML Signature eingesetzt werden.
hoch	XML Signature	Die Message wird gegen Manipulationen mittels XML Signature geschützt. Die digitale Signatur kann z.B. auf Basis von fortgeschrittener Signatur und Software-Tokens erfolgen.
Sehr hoch	XML Signature mit gesetzeskonformer digitaler Signatur	Die Message wird gegen Manipulationen mittels XML Signature geschützt. Qualifizierte Signatur ist umzusetzen.

6.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur

Nach Bestimmung der Architektur, der Anwendungsfälle, der Entwurfsmuster und des Schutzbedarfes stehen die möglichen Technologien fest, die eingesetzt werden können. Nehmen wir das Beispiel der Bonitätsprüfung. Hier gelten die Architekturen A2 oder A4 in Kombination mit der Architektur A5. Der Schutzbedarf für Request und Response bzgl. Integrität ist mindestens mit "hoch" anzusetzen. Daher muss die Technologie XML Signature eingesetzt werden. Besteht nun bereits eine Public Key Infrastruktur (PKI), so kann diese eingesetzt werden, um mit Hilfe von Zertifikaten die Technologie XML Signature umzusetzen.

Zu beachten ist, dass hier aus Gründen der Klarheit noch nicht diskutiert wurde, dass nur Teile der Message oder verschiedene Teile der Message durch unterschiedliche Teilnehmer signiert werden könnten.

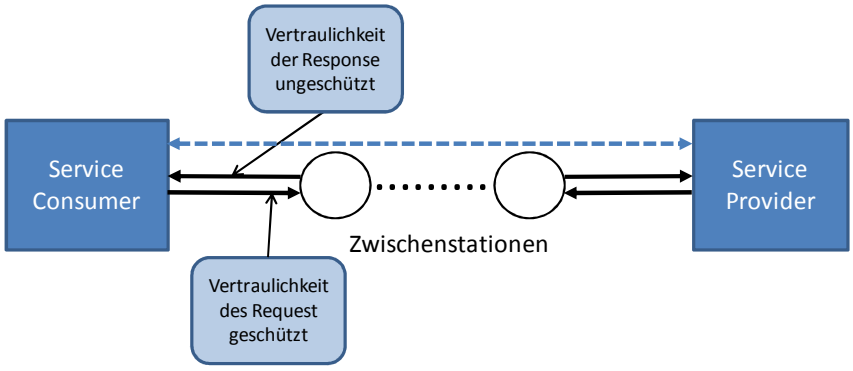
7. Entwurfsmuster für Vertraulichkeit

Grundlegende Entwurfsmuster zur Umsetzung der Vertraulichkeit werden in diesem Kapitel beschrieben. Die im Folgenden dargestellte Tabelle ermöglicht eine Zuordnung zwischen den Architekturen und Anwendungsfällen zu möglichen Entwurfsmustern, die zur Umsetzung einer konkreten Problemstellung herangezogen werden können.

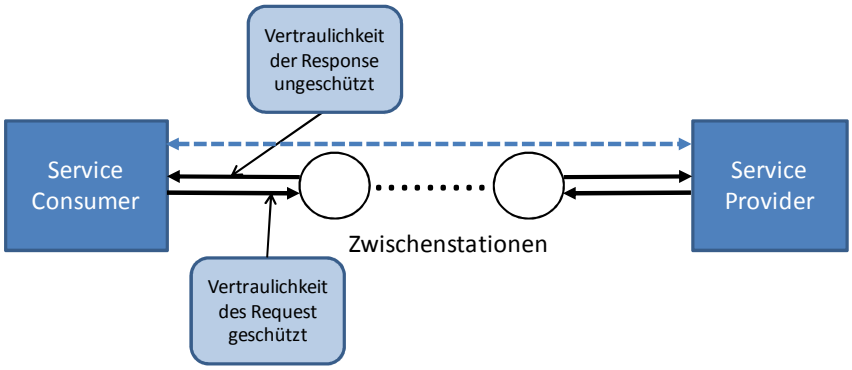
7.1 Auswahl der Entwurfsmuster

	(A2) mit (A5) Direkte Kommunikation beidseitige Vertrauensbeziehung	(A4) mit (A5) Indirekte Kommunikation beidseitige Vertrauensbeziehung
(DS_U1) Datenübermittlung	➔ (P4) Vertraulichkeit des Kommunikationskanals gewährleistet	➔ (P1) Vertraulichkeit des Request gewährleistet ➔ (P2) Vertraulichkeit der Response gewährleistet ➔ (P3) Vertraulichkeit des Requests und der Response gewährleistet
(DS_U2) Transaktion	➔ (P4) Vertraulichkeit des Kommunikationskanals gewährleistet	➔ (P1) Vertraulichkeit des Request gewährleistet ➔ (P2) Vertraulichkeit der Response gewährleistet ➔ (P3) Vertraulichkeit des Requests und der Response gewährleistet

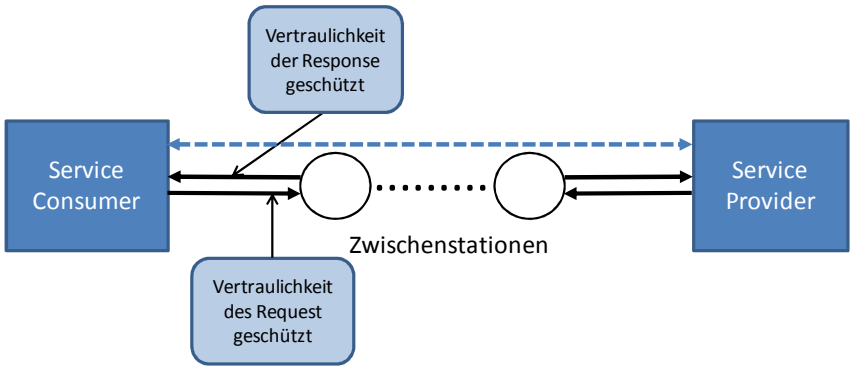
7.2 Entwurfsmuster

Muster: Vertraulichkeit des Request gewährleistet	P1
Beschreibung	 Der Service Consumer sendet einen Request, der vor unautorisierter Einsicht geschützt wird (Vertraulichkeit geschützt).

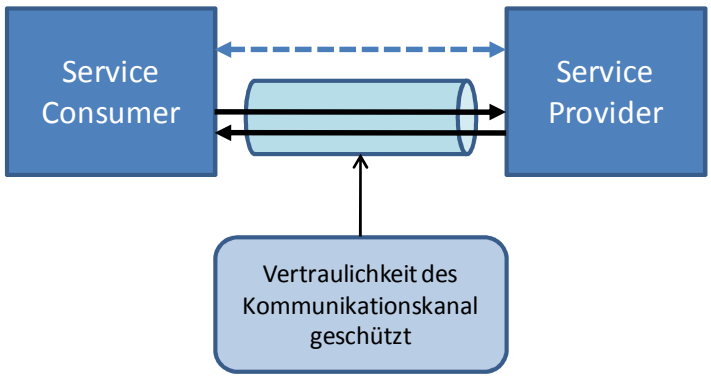
<table>
<tr><td>Beispiel-
szenarien</td><td>Ein Service Consumer bestellt eine Ware in einem Online-Shop und erhält eine Auftragsbestätigung. Was der Service Consumer bestellt, soll für einen Dritten verborgen bleiben. Der Request wird verschlüsselt.
Eine Bank will die Bonität eines Kunden überprüfen und stellt deswegen eine verschlüsselte Anfrage an die Schufa.</td></tr>
<tr><td>Beispiele für Technologien</td><td><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Vertraulichkeit des Requests</td><td>digital verschlüsselte Message (XML-Encryption)</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="2">Muster: Vertraulichkeit der Response gewährleistet</th><th>P2</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer sendet einen Request, der nicht vor unautorisierter Einsicht geschützt ist. Der Service Provider empfängt eine Response vom Service Provider. Die Response ist vor unautorisierter Einsicht geschützt (Vertraulichkeit geschützt).</td></tr>
<tr><td>Beispielszenarien</td><td colspan="2">Ein Service Consumer fragt Rentendaten beim Rentenversicherer ab. Der Rentenversicherer übermittelt die entsprechenden Daten, die verschlüsselt sind.</td></tr>
<tr><td>Beispiele für Technologien</td><td colspan="2"><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Vertraulichkeit der Response.</td><td>digital verschlüsselte Message (XML Encryption)</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="2">Muster: Vertraulichkeit des Requests und der Response gewährleistet</th><th>P3</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Der Service Consumer sendet einen Request, der vor unautorisierter Einsicht geschützt wird. Der Service Provider empfängt eine Response vom Service Provider, die vor unautorisierter Einsicht geschützt wird. Die Vertraulichkeit von Request und Response ist also geschützt.</td></tr>
<tr><td>Beispiel-szenarien</td><td colspan="2">Ein Service Consumer fragt Bonitätsdaten für einen Kunden bei seiner Bank an. Die Bank fragt die Bonitätsdaten bei einem Service Provider an. Der Service Provider übermittelt die Bonitätsprüfung für den im Request benannten Kunden an die Bank. Diese sendet die Daten an den Service Consumer. Request als auch Response sind verschlüsselt.</td></tr>
<tr><td>Beispiele für Technologien</td><td colspan="2"><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Integrität des Requests / der Response.</td><td>digital verschlüsselte Message (XML Encryption)</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="2">Muster: Vertraulichkeit des Kommunikationskanals gewährleistet</th><th>P4</th></tr>
<tr><td>Beschreibung</td><td colspan="2">

Zwischen Service Consumer und Service Provider wird ein Kommunikationskanal aufgebaut, der vor unautorisierter Einsicht geschützt wird (Vertraulichkeit geschützt). Bei direkter Kommunikation sind damit sowohl Request als auch Response vor unautorisierter Einsicht geschützt.</td></tr>
</table>

<table>
<tr><td>Beispiel-
szenarien</td><td>Die Bank fragt die Bonitätsdaten eines Kunden bei einem Service Provider an. Der Service Provider übermittelt die Bonitätsprüfung für den im Request benannten Kunden an die Bank. Request als auch Response sind über einen verschlüsselten VPN-Tunnel geschützt.</td></tr>
<tr><td>Beispiele für
Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/
Standards</th></tr>
<tr><td>Schutz der Integrität des
Requests / der Response.</td><td>Geschützter SSL/TLS Kanal</td></tr>
</table>
</td></tr>
</table>

7.3 Auswahl der Technologien und Methoden

Schutzbedarf	Technologien und Methoden	Erläuterung
normal	SSL/TLS XML Encryption	Wenn der Schutzbedarf der Message (Request oder Response) bzgl. Vertraulichkeit als normal eingestuft ist, so können die Technologien SSL/TLS oder XML Encryption eingesetzt werden.
hoch	XML Encryption	Die Message wird vor unautorisierter Einsicht mittels XML Encryption geschützt. Die Verschlüsselung kann z.B. mit symmetrischen oder asymmetrischen Verfahren realisiert werden. Die Anforderungen an die zu verwendenden Kryptoalgorithmen und Schlüssellängen sind nicht zu hoch (z.B. AES 128 Bit oder RSA 1024 Bit).
sehr hoch	XML Encryption mit aktuell als sehr sicher eingestuften Kryptoalgorithmen und Schlüssellängen	Die Message wird vor unautorisierter Einsicht mittels XML Encryption geschützt. Die Verschlüsselung kann z.B. mit symmetrischen oder asymmetrischen Verfahren realisiert werden. Die aktuell als sehr sicher eingestuften Kryptoalgorithmen und Schlüssellängen sind zu verwenden (z.B. AES 256 Bit oder RSA 4096 Bit).

7.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur

Nach Bestimmung der Architektur, der Anwendungsfälle, der Entwurfsmuster und des Schutzbedarfes stehen die möglichen Technologien fest, die eingesetzt werden können. Nehmen wir das Beispiel der Bonitätsprüfung. Hier gelten die Architekturen A2 oder A4 in Kombination mit der Architektur A5. Der Schutzbedarf für Request und Response bzgl. Vertraulichkeit ist mindestens mit "hoch" anzusetzen. Daher muss die Technologie XML Encryption eingesetzt werden. Besteht nun bereits eine Public Key Infrastruktur (PKI), so kann diese eingesetzt werden, um mit Hilfe von Zertifikaten die Technologie XML Encryption (Kryptoalgorithmus RSA, Schlüssellänge 1024 Bit) umzusetzen.

Zu beachten ist, dass hier aus Gründen der Klarheit noch nicht diskutiert wurde, dass nur Teile der Message oder verschiedene Teile der Message für unterschiedliche Teilnehmer verschlüsselt werden könnten.

8. *Entwurfsmuster für Nichtabstreitbarkeit*

Grundlegende Entwurfsmuster zur Umsetzung der Nichtabstreitbarkeit werden in diesem Kapitel beschrieben. Die im Folgenden dargestellte Tabelle ermöglicht eine Zuordnung zwischen den Architekturen und Anwendungsfällen zu möglichen Entwurfsmustern, die zur Umsetzung einer konkreten Problemstellung herangezogen werden können.

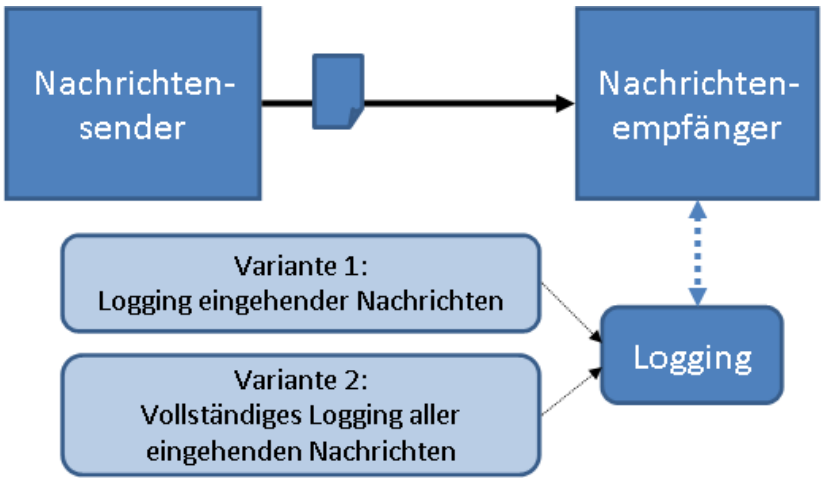
8.1 Auswahl der Entwurfsmuster

	A1 Direkter Request ohne Response	A2 Direkter Request mit Response	A3 Direkter Request mit Response und Bestätigung
(NA_U1) -Nichtabstreitbarkeit eines Aufrufes durch den Consumer	➔ (P1) Nichtabstreitbarkeit einer Nachricht durch Logging/Archivierung des Empfängers (Variante (1), d.h. einfaches Logging) ➔ (P2) Nichtabstreitbarkeit einer Nachricht durch Logging/Archivierung des Senders (Variante (2), d.h. vollständiges Logging) ➔ (P3) Nichtabstreitbarkeit einer Nachricht durch Signatur des Senders Dabei gilt: Sender = Service Consumer Empfänger = Service Provider Nachricht = Service Request	Entwurfsmuster wie bei NA_U2 für A1. Dabei gilt: Sender = Service Provider Empfänger = Service Consumer Nachricht = Service Response. Aus der Response muss der Request ableitbar sein.	Entwurfsmuster wie bei NA_U1 für A1. Dabei gilt: Sender = Service Consumer Empfänger = Service Provider Nachricht = Empfangsbestätigung vom Service Consumer an den Service Provider, dass der Service Consumer die Response erhalten hat; aus der Response ist der Request ableitbar.

(NA_U2) - Nichtabstreitbarkeit eines Aufrufes durch den Provider	➔ (P1) Nichtabstreitbarkeit einer Nachricht durch Logging/Archivierung des Empfängers (Variante (2), d.h. vollständiges Logging) ➔ (P2) Nichtabstreitbarkeit einer Nachricht durch Logging/Archivierung des Senders (Variante (1), d.h. einfaches Logging) Dabei gilt: Sender = Service Consumer Empfänger = Service Provider Nachricht = Service Request	Entwurfsmuster wie bei NA_U1 für A1. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U1 für A2.	Entwurfsmuster wie bei NA_U2 für A1. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U1 für A3.
(NA_U3) - Nichtabstreitbarkeit des Inhaltes eines Aufrufes eines Services durch den Consumer	Entwurfsmuster wie bei NA_U1 für A1, aber *Inhalt* des Requests wird geloggt bzw. signiert. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U1 für A1.	Entwurfsmuster wie bei NA_U2 für A2, aber aus der Response muss der *Inhalt* des Requests ableitbar sein. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U2 für A2.	Entwurfsmuster wie bei NA_U1 für A3, aber die Empfangsbestätigung muss den *Inhalt* des Service Request bestätigen. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U1 für A3.
(NA_U4) - Nichtabstreitbarkeit des Inhaltes eines Aufrufes eines Services durch den Provider	Entwurfsmuster wie bei NA_U2 für A1, aber *Inhalt* des Requests wird geloggt bzw. signiert. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U2 für A1.	Entwurfsmuster wie bei NA_U1 für A2, aber aus der Response muss der *Inhalt* des Requests ableitbar sein. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U1 für A2.	Entwurfsmuster wie bei NA_U2 für A3, aber die Empfangsbestätigung muss den *Inhalt* des Service Request bestätigen. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U2 für A3.

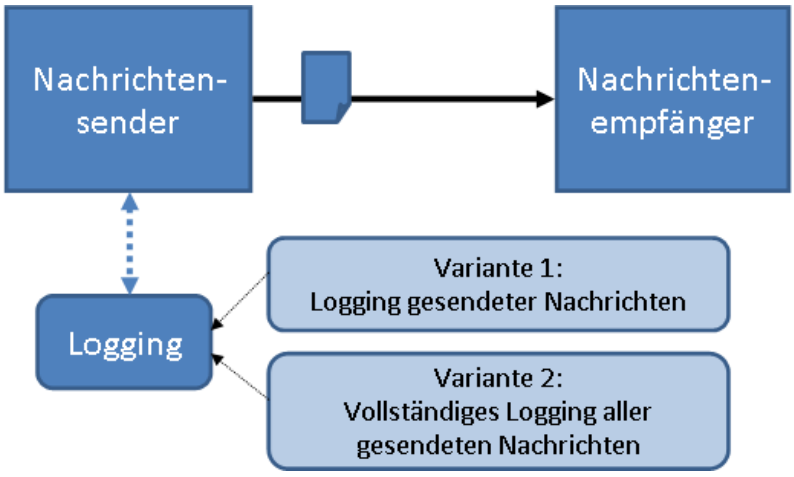
(NA_U5) - Nichtabstreitbarkeit einer Response durch den Consumer	--	Entwurfsmuster wie bei NA_U2 für A1. Dabei gilt: Sender = Service Provider Empfänger = Service Consumer Nachricht = Service Response.	Entwurfsmuster wie bei NA_U1 für A1. Dabei gilt: Sender = Service Consumer Empfänger = Service Provider Nachricht = Empfangsbestätigung der Response
(NA_U6) – Nichtabstreitbarkeit einer Response durch den Provider	--	Entwurfsmuster wie bei NA_U1 für A1. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U5 für A2.	Entwurfsmuster wie bei NA_U2 für A1. Dabei gilt: Sender, Empfänger, Nachricht: Wie bei NA_U5 für A3.
(NA_U7) – Nichtabstreitbarkeit des Inhaltes einer Response eines Services durch den Consumer	--	Entwurfsmuster wie bei NA_U5 für A2, aber *Inhalt* der Response wird geloggt bzw. signiert.	Entwurfsmuster wie bei NA_U5 für A3, aber die Empfangsbestätigung muss den *Inhalt* der Response bestätigen
(NA_U8) – Nichtabstreitbarkeit des Inhaltes einer Response eines Services durch den Provider	--	Entwurfsmuster wie bei NA_U6 für A2, aber *Inhalt* der Response wird geloggt bzw. signiert.	Entwurfsmuster wie bei NA_U6 für A3, aber die Empfangsbestätigung muss den *Inhalt* der Response bestätigen

8.2 Entwurfsmuster

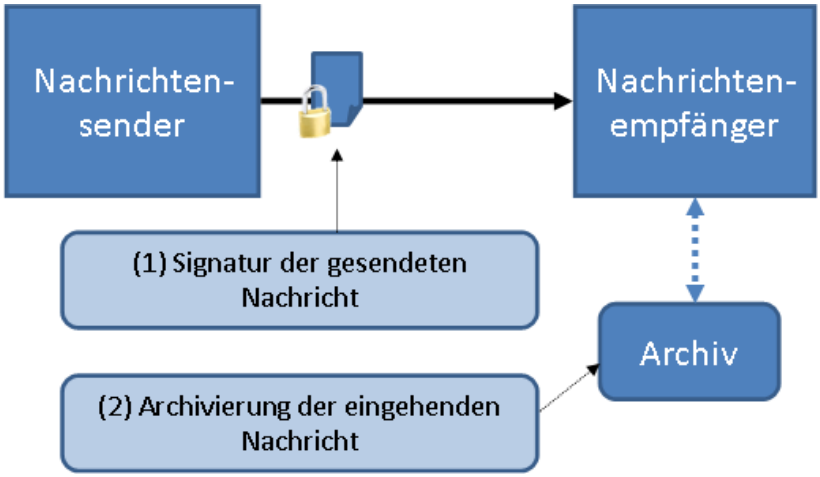
<table>
<tr><td colspan="2">Muster: Nichtabstreitbarkeit einer Nachricht durch Logging/Archivierung des Empfängers</td><td>P1</td></tr>
<tr><td>Beschreibung</td><td colspan="2">

Ein Sender sendet eine Nachricht an einen Empfänger.
Der Empfänger loggt mit seinem Logging-Modul:
• Identität des Senders
• Header bzw. Inhalt der Nachricht (abhängig davon, ob nur die Nachrichtenzustellung oder auch der Inhalt der Nachricht nicht abstreitbar sein soll)
• ggf. weitere Attribute wie Datum/Uhrzeit, IP-Adresse des Senders, Einmalwert (gegen Replay-Attacken) etc.
Die geloggten Daten speichert er in seinem Archivierungs-Modul. Die Aufbewahrungsdauer hängt vom Anwendungsfall ab.
Zwei Varianten der Auswertung:
(1) Annahme: Der Sender soll nicht abstreiten können, dass er eine Nachricht geschickt hat. Im Streitfall liest der Empfänger die geloggte Nachricht aus seinem Archivierungs-Modul und kann damit nachweisen, dass der Sender die Nachricht tatsächlich gesendet hat.
(2) Annahme: Der Empfänger soll nicht abstreiten können, dass er eine Nachricht bekommen hat. Im Streitfall liest der Empfänger sämtliche Logging-Einträge des fraglichen Zeitraums aus seinem Archivierungs-Modul und zeigt, dass die fragliche Nachricht nicht dabei war. Damit weist er nach, dass er die Nachricht nicht bekommen hat. Voraussetzung ist, dass das Logging beim Empfänger vollständig ist.</td></tr>
<tr><td>Beispiel-szenarien</td><td colspan="2">• NA_U1: Ein Service Consumer ruft einen kostenpflichtigen Service auf. Wenn der Service Provider diesen Aufruf in Rechnung stellt, muss er im Streitfall nachweisen können, dass der Service</td></tr>
</table>

	Consumer den Service tatsächlich aufgerufen hat. • NA_U2 / NA_U4: Ein Service Consumer ruft einen Service auf, mit dem er ein Abonnement kündigt. Im Streitfall behauptet der Service Provider, diese Kündigung habe er nicht erhalten. Jetzt muss der Service Consumer nachweisen, dass er diesen Kündigungs-Service tatsächlich aufgerufen hat. • NA_U3: Ein Service Consumer sendet einen fehlerhaften Service Request, der beim Service Provider einen Schaden verursacht. Der Service Provider fordert Schadensersatz vom Service Consumer. • NA_U3: Ein Service Consumer ruft einen Bestell-Service auf. Im Streitfall darf er den Inhalt der Bestellung nicht abstreiten können. • NA_U5: Der Service Consumer ruft einen kostenpflichtigen Service auf. Er verweigert die Zahlung, weil er behauptet, er habe keine Response erhalten. (Unterschied zu NA_U1: Bei NA_U1 streitet der Service Consumer den Aufruf ab. Hier gibt er den Aufruf zu, streitet aber ab, eine Response bekommen zu haben.) • NA_U8: Ein Service Provider sendet einem Service Consumer Börsenkurse, der daraufhin falsche Kaufentscheidungen trifft und deshalb Schadensersatz vom Service Provider fordert. Der Service Provider darf nicht abstreiten können, die falschen Daten geschickt zu haben.

Muster: Nichtabstreitbarkeit einer Nachricht durch Logging/Archivierung des Senders	P2
Beschreibung	 Ein Sender sendet eine Nachricht an einen Empfänger.

	Der Sender loggt mit seinem Logging-Modul: • Identität des Empfängers • Header bzw. Inhalt der Nachricht (abhängig davon, ob nur der Nachrichtenversand oder auch der Inhalt der Nachricht nicht abstreitbar sein soll) • ggf. weitere Attribute, siehe P1 Die geloggten Daten speichert er in seinem Archivierungs-Modul. Die Aufbewahrungsdauer hängt vom Anwendungsfall ab. Zwei Varianten der Auswertung: (1) Annahme: Der Empfänger soll nicht abstreiten können, dass er eine Nachricht bekommen hat. Im Streitfall liest der Sender die geloggte Nachricht aus seinem Archivierungs-Modul und kann damit nachweisen, dass er die Nachricht tatsächlich gesendet hat. (2) Annahme: Der Sender soll nicht abstreiten können, dass er eine Nachricht gesendet hat. Im Streitfall liest der Sender sämtliche Logging-Einträge des fraglichen Zeitraums aus seinem Archivierungs-Modul und zeigt, dass die fragliche Nachricht nicht dabei war. Damit weist er nach, dass er die Nachricht nicht gesendet hat. Voraussetzung ist, dass das Logging beim Sender vollständig ist.
Beispiel-szenarien	Siehe P1.
Beispiele für Technologien	Siehe P1.

<table>
<tr><th>Muster: Nichtabstreitbarkeit einer Nachricht durch Signatur des Senders</th><th>P3</th></tr>
<tr><td>Beschreibung</td><td>


Ein Sender sendet eine Nachricht an einen Empfänger.
Der Sender signiert die Nachricht (ggf. mit weiteren Attributen wie Datum/Uhrzeit etc.) mit seinem privaten Schlüssel.
Der Empfänger akzeptiert nur Nachrichten mit einer gültigen Signatur, ggf. muss er die zusätzlichen Attribute überprüfen. Er speichert in seinem Archivierungs-Modul:
• Identität des Empfängers
• Öffentlichen Schlüssel des Empfängers mit Zertifikat
• Nachricht (Header oder Inhalt)
• ggf. weitere Attribute
• Signatur der Nachricht (und ggf. der weiteren Attribute)
Annahme: Der Sender soll nicht abstreiten können, dass er eine Nachricht geschickt hat. (Mit anderen Worten: Der Empfänger soll nachweisen können, dass er die Nachricht erhalten hat.)
Im Streitfall liest der Empfänger die oben genannten Daten aus seinem Archivierungs-Modul. Mit Hilfe des Zertifikats verifiziert er den öffentlichen Schlüssel des Senders, mit Hilfe dieses öffentlichen Schlüssels verifiziert er die Signatur.
</td></tr>
<tr><td>Beispiel-szenarien</td><td>Siehe P1.</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Schutz der Integrität des Requests / der Response</td><td>digital signierte Nachricht (XML Signature)</td></tr>
</table>
</td></tr>
</table>

8.3 Auswahl der Technologien und Methoden

Für die Nichtabstreitbarkeit gelten die in folgender Tabelle möglichen Technologien und Methoden:

Schutzbedarf	Technologien und Methoden	Erläuterungen
normal	Logging und Archivierung der Nachricht, die nicht abstreitbar sein soll (Variante 1 bei Entwurfsmuster P1 und P2)	Die Instanz, die eine Nachricht nachweisen will (entweder dass sie die Nachricht gesendet hat oder dass sie die Nachricht erhalten hat), loggt das Senden bzw. das Empfangen der Nachricht und archiviert die geloggten Daten. Anhand des zugehörigen Log-Eintrags kann sie nachweisen, dass sie die Nachricht gesendet bzw. empfangen hat. Sicherheitsbewertung: Die loggende Instanz kann die Logging-Einträge selbst erzeugen. Ist die loggende Instanz der Empfänger (d.h. Entwurfsmuster P1), gilt außerdem: Selbst wenn die Logging-Einträge korrekt sind, kann ein Sender abstreiten, dass eine Nachricht von ihm stammt. Deshalb genügt diese Lösung nur bei normalem Schutzbedarf.
normal	Vollständiges Logging und Archivierung der Nachricht, die nicht abstreitbar sein soll (Variante 2 bei Entwurfsmuster P1 und P2)	Die Instanz, die eine Nachricht nicht abstreiten können soll (entweder dass sie die Nachricht gesendet hat oder dass sie die Nachricht erhalten hat), loggt sämtliche gesendeten bzw. empfangenen Nachrichten und archiviert die geloggten Daten. Ist die Nachricht nicht in den Logdaten vorhanden, ist das ein Nachweis, dass die Nachricht nicht gesendet bzw. empfangen wurde. Sicherheitsbewertung:

		Dieser Rückschluss ist nur erlaubt, wenn das Logging vollständig war. Die Instanz kann ihr Logging-Modul so manipulieren, dass es nicht zuverlässig loggt bzw. sie kann geloggte Datensätze löschen. Deshalb genügt diese Lösung nur bei normalem Schutzbedarf.
normal	Signatur / (vollständiges) Logging und Archivierung einer Response bzw. Bestätigung zu der Nachricht, die nicht abstreitbar sein soll. (NA_U1 und NA_U3 bei A2 sowie NA_U6 und NA_U8 bei A3)	Es gelten die gleichen Einschränkungen wie oben. Hinzu kommt, dass der Empfänger einer falschen Nachricht (als Response bzw. Bestätigung) nicht verpflichtet ist, mit einer Korrektur zu antworten. Deshalb genügt diese Lösung nur bei normalem Schutzbedarf.
hoch	Signatur und Archivierung (Entwurfsmuster P3)	siehe Integrität, Abschnitt 6.3
sehr hoch	Signatur mit sehr hohen Anforderungen und Archivierung (Entwurfsmuster P3)	siehe Integrität, Abschnitt 6.3

8.4 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur

Zum Logging:
Vorhandene Logging-Module und Archivierungs-Module sollten verwendet werden. Abhängig von der angestrebten Beweiskraft, muss die Zuverlässigkeit der Module nachgewiesen werden, z. B. durch einen Gutachter oder einer Zertifizierung.

Zur Signatur:
Siehe Integrität, Abschnitt 6.3.

Zur Archivierung:
Abhängig von der Anwendung und dem Schutzbedarf können vorhandene Datenbanken verwendet werden. Backup- und Recovery-Konzepte sind zu erstellen.

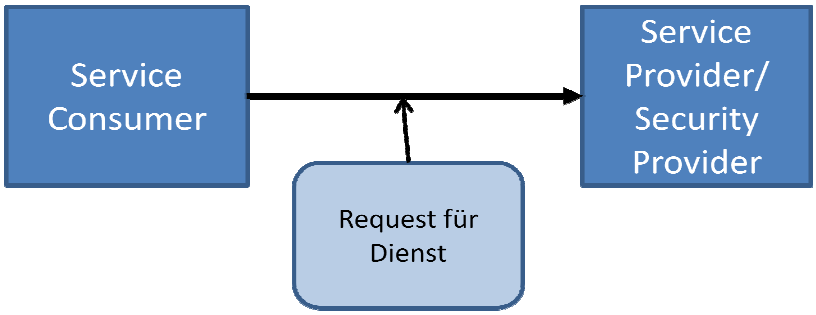
9. Entwurfsmuster für Autorisierung

Grundlegende Entwurfsmuster zur Umsetzung der Autorisierung werden in diesem Kapitel beschrieben. Die im Folgenden dargestellte Tabelle ermöglicht eine Zuordnung zwischen den Architekturen und Anwendungsfällen zu möglichen Entwurfsmustern, die zur Umsetzung einer konkreten Problemstellung herangezogen werden können.

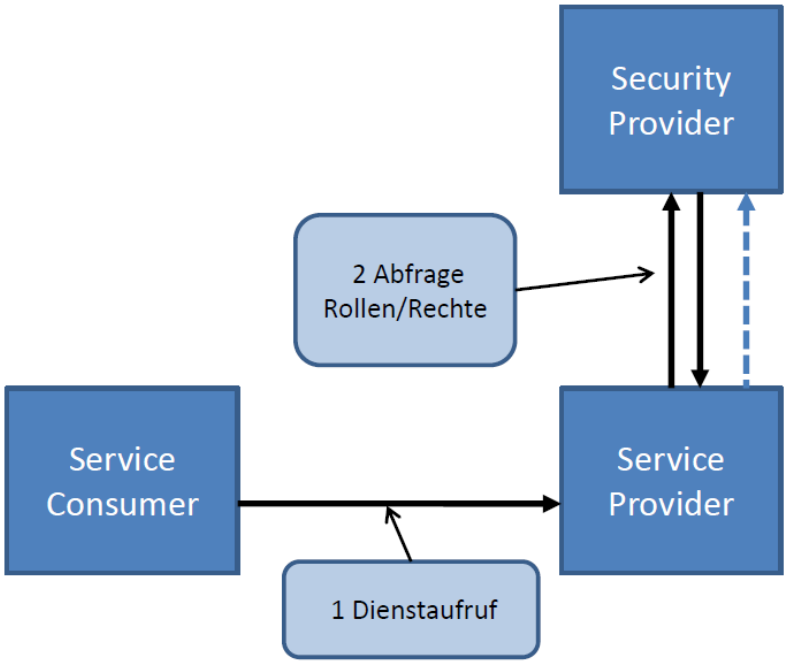
9.1 Auswahl der Entwurfsmuster

	(A5) Direktes Vertrauensverhältnis	(A6) Indirektes Vertrauensverhältnis
(I_U1)	➔ (P1) Direkte Autorisierung	---
(I_U2)	➔ (P1) Direkte Autorisierung	➔ (P2) Zentralisierte Autorisierung mit Security Provider ➔ (P3) Aufruf mit 4 Augen Bestätigung ➔ (P4) Autorisierung mit 4-Augen Prinzip Rückkopplung
(I_U3)	---	➔ (P3) Aufruf mit 4 Augen Bestätigung ➔ (P4) Autorisierung mit 4-Augen Prinzip Rückkopplung
(I_U4)	---	➔ (P5) Föderative Autorisierung mit Rückkopplung

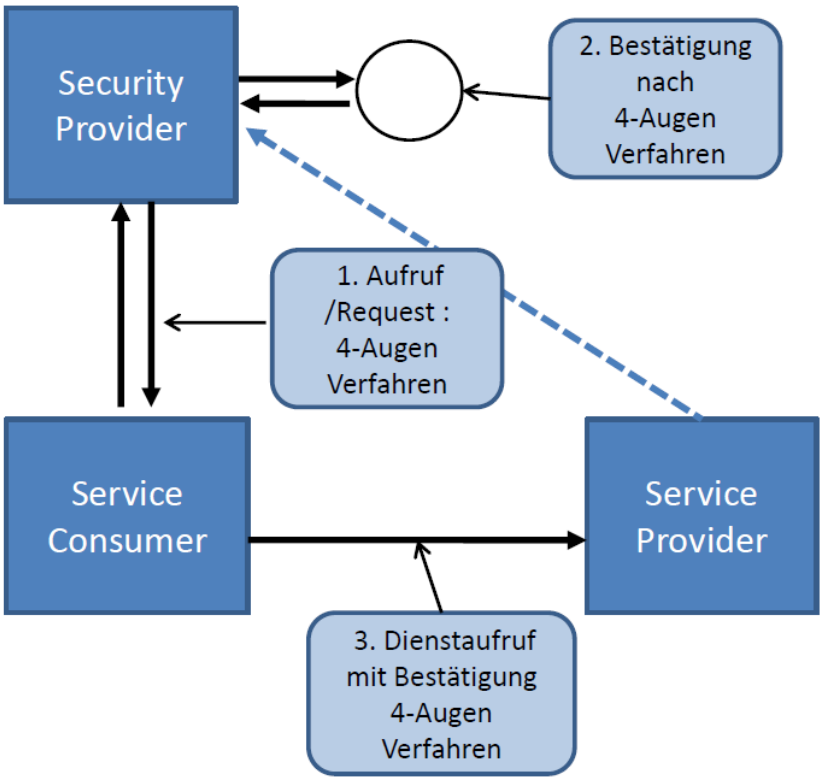
9.2 Entwurfsmuster

Muster: Direkte Autorisierung	P1
Beschreibung	 Der Service Consumer schickt einen Request für einen Service. Der Service Provider gibt ihm über eine eigene Rechteverwaltung die Erlaubnis für den Zugriff.

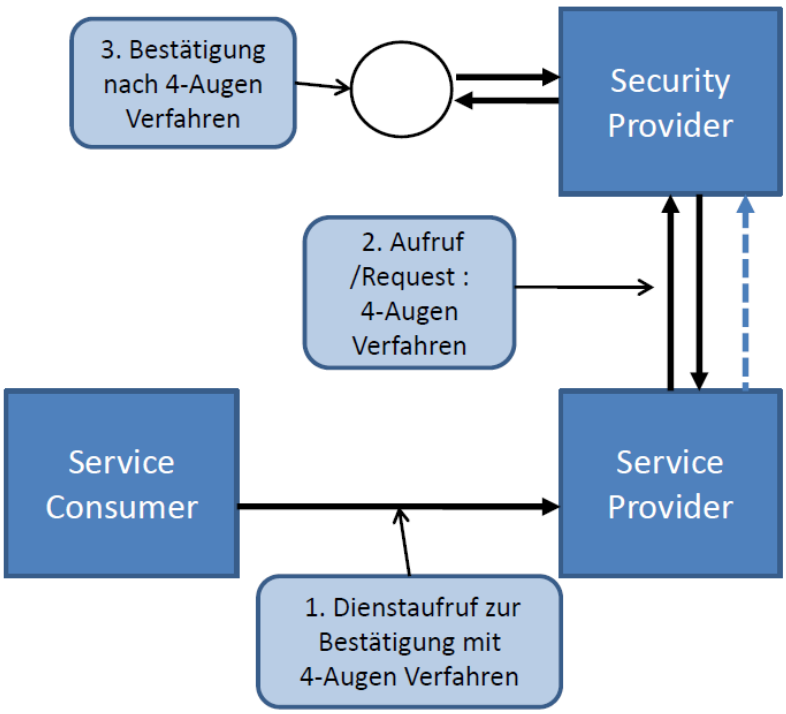
<table>
<tr><td>Beispiel-szenarien</td><td>Autorisierung eines Nutzers bei einer Webseite oder einem Service.</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Autorisierung des Benutzers:</td><td>Autorisierung basierend auf der Authentifizierung des Benutzers mit Benutzername/Passwort,…</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Muster: Zentralisierte Autorisation mit Security Provider</th><th>P2</th></tr>
<tr><td>Beschreibung</td><td>


Der Service Consumer schickt einen Request für einen Service. Der Service Provider kontaktiert einen Security Provider (Metadirectory, LDAP), um die Rechte dieses Nutzers zu erhalten.
</td></tr>
<tr><td>Beispiel-szenarien</td><td>Autorisierung eines Nutzers mittels eines unternehmensinternen LDAP.</td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Authentifizierung des Benutzers:</td><td>Benutzername/Passwort,…</td></tr>
<tr><td>Abfrage der Rechte aus einem eines Metadirectory</td><td>LDAP, WS-Trust,…</td></tr>
</table>
</td></tr>
</table>

Muster: Aufruf mit 4 Augen Bestätigung	P3
Beschreibung	 Zur Autorisierung beim Service wird ein Credential genutzt, in dem eine dritte (vom Service vertraute) Instanz Aussagen über die Rechte des Benutzer macht (Bestätigung nach 4 Augen Prinzip). Alle Services und Nutzer vertrauen dieser Instanz. Soll der Benutzer beim Service autorisiert werden, sind folgende Schritte notwendig: 1. Der Service Consumer muss sich durch den Security Provider autorisieren lassen. 2. Der Security Provider muss Aussagen über die Rechte (Claims) einfügen, mit dem sich der Service Consumer beim Service Provider autorisieren kann. 3. Der Security Provider holt eine zusätzliche Aussage über die Rechte (Claims) des Nutzers bei einer weiteren Person ein. Damit wird ein klassisches Vier-Augen Prinzip abgebildet. 4. Der Service Provider muss dem Security Provider vertrauen.

<table>
<tr><td>Beispiel-szenarien</td><td>Zentrale Rollen und Rechte Verwaltung im Unternehmen.</td></tr>
<tr><td>Beipiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Autoriserung des Benutzers:</td><td>LDAP, Active Directory</td></tr>
<tr><td>Bereitstellung der Rollen und Rechte:</td><td>RBAC</td></tr>
<tr><td>Abrufen der Claims:</td><td>SAML, Zertifikate</td></tr>
</table>
</td></tr>
</table>

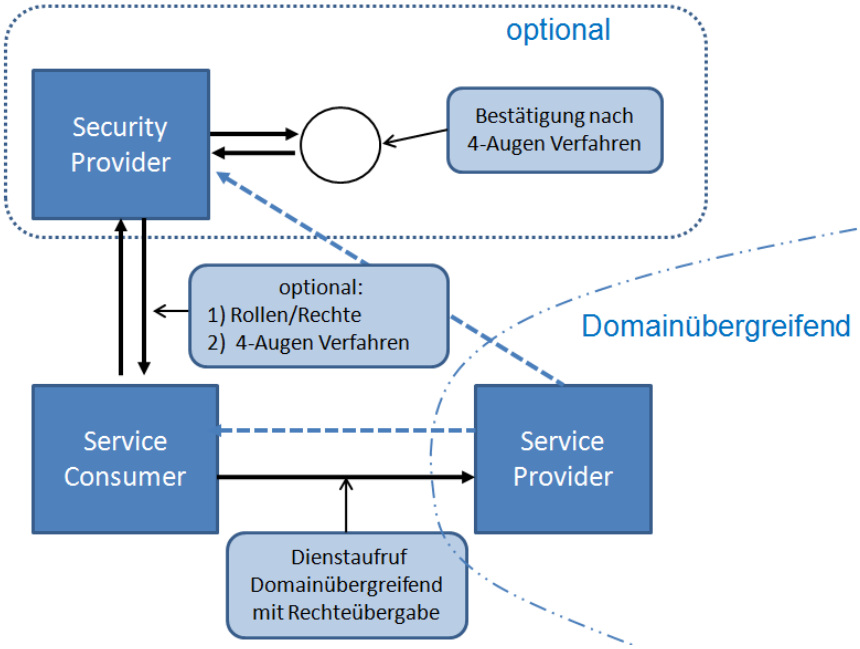
<table>
<tr><th>Muster: Autorisierung mit 4-Augen Prinzip Rückkopplung</th><th>P4</th></tr>
<tr><td>Beschreibung</td><td>


Beim Aufruf eines Service wählt der Provider von welchem Security Providern er die Rechte des Nutzers erhalten kann. Hierbei kann es mehrere Security Provider geben und der Service entscheidet welchen Security Providern vertraut wird.
Soll der Benutzer beim Service autorisiert werden, sind folgende Schritte notwendig:

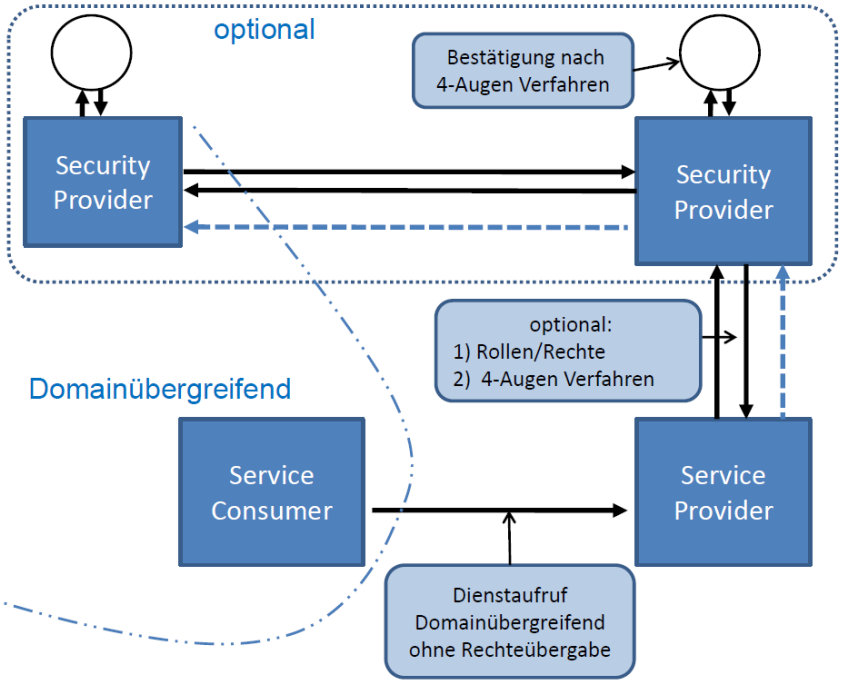
Der Service Provider schickt die Anfrage des Nutzers an einen Security Provider und erfragt dessen Rechte.
Der Security Provider liefert die Rechte des Nutzers zurück.
Der Security Provider kann die Rechte des Nutzers über ein 4-Augen Prinzip estätigen lassen. Die bestätigten Rechte werden an den Service Provider übermittelt.

</td></tr>
<tr><td>Beispiel-</td><td>Zentrale Rechteverwaltung im Unternehmen</td></tr>
</table>

<table>
<tr><td>szenarien</td><td></td></tr>
<tr><td>Beispiele für Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Autorisierung des Nutzers:</td><td>Rollen und Rechte</td></tr>
<tr><td>Bereitstellung der Rollen und Rechte:</td><td>RBAC</td></tr>
<tr><td>Abrufen der Claims:</td><td>SAML, Zertifikate</td></tr>
</table>
</td></tr>
</table>

Muster: Förderative Autorisierung	P5
Beschreibung	 Bei der förderativen Autorisierung existiert ein Vertrauenszirkel zwischen verschiedenen Security Providers aus unterschiedlichen Vertrauensdomainen. Förderationsverträge regeln die Vertrauensbeziehung zwischen den Partnern. Alle Security Provider akzeptieren Aussagen über den Besitz von Rechten von den Security Providern der Partner. Soll der Benutzer beim Service autorisiert werden, sind folgende Schritte notwendig: 1. Der Service Consumer muss sich durch den Security Provider autorisieren lassen. 2. Der Security Provider muss Aussagen über die Rechte (Claims) einfügen, mit dem sich der Service Consumer beim Service Provider autorisieren kann. 3. Der Security Provider holt eine zusätzliche Aussage über die Rechte (Claims) des Nutzers bei einer weiteren Person ein. Damit wird ein klassisches Vier-Augen Prinzip abgebildet. 4. Der Service Provider muss dem Security Provider vertrauen.

<table>
<tr><td>Einsatzszena-rien</td><td>Findet Anwendung in B2B Szenarien, in denen zwei Unternehmen bereits jeweils eine eigene Rechte Management besitzen. Jedes Unternehmen stellt einen eigenen Security Provider bereit.
Beispiel: Mitarbeiter des einen Unternehmens möchte auf eine Ressource im Partnerunternehmen zugreifen</td></tr>
<tr><td>Technologien</td><td>
<table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Autorisierung des Nutzers:</td><td></td></tr>
<tr><td>Bereitstellung der Autorisierungs-entscheidung:</td><td>SAML, …</td></tr>
<tr><td>Abrufen der Autorisierungs-Claims:</td><td>WS-Trust, SAML</td></tr>
<tr><td>Etablierung der Föderation:</td><td>WS-Federation, SAML</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Muster: Föderative Autorisierung mit Rückkopplung</th><th>P6</th></tr>
<tr><td>Beschreibung</td><td>


Bei der förderativen Autorisierung existiert ein Vertrauenszirkel zwischen verschiedenen Security Providers aus unterschiedlichen Vertrauensdomainen. Förderationsverträge regeln die Vertrauensbeziehung zwischen den Partnern. Alle Security Provider akzeptieren Rechte von den Security Providern der Partner.
Soll der Benutzer beim Service autorisiert werden, sind folgende Schritte notwendig:
1. Der Service Consumer muss sich durch den förderativer Security Provider autorisieren lassen
2. Der Security Provider muss Aussagen über die Rechte
</td></tr>
</table>

<table>
<tr><td></td><td>(Claims) einfügen, mit dem sich der Service Consumer beim Service Provider autorisieren kann.
3. Der Security Provider holt eine zusätzliche Aussagen über die Rechte (Claims) des Nutzers bei
a. einem weiteren Security Provider ein. Damit auch Domainübergreifend gearbeitet werden.
b. einer weiteren Person ein. Damit wird ein klassisches Vier-Augen Prinzip abgebildet.
4. Der Service Provider muss dem Förderativen Security Provider vertrauen.</td></tr>
<tr><td>Einsatzszenarien</td><td>Findet Anwendung in B2B Szenarien, in denen zwei Unternehmen bereits jeweils eine eigene Rechte Management besitzen. Jedes Unternehmen stellt einen eigenen Security Provider bereit.
Beispiel: Mitarbeiter des einen Unternehmens möchte auf Ressource im Partnerunternehmen zugreifen</td></tr>
<tr><td>Technologien</td><td><table>
<tr><th>Aspekt</th><th>Mögliche Technologien/ Standards</th></tr>
<tr><td>Autorisierung des Nutzers:</td><td></td></tr>
<tr><td>Bereitstellung der Autorisierungsentscheidung:</td><td>SAML, …</td></tr>
<tr><td>Abrufen der Autorisierungs-Claims:</td><td>WS-Trust, SAML</td></tr>
<tr><td>Etablierung der Föderation:</td><td>WS-Federation, SAML</td></tr>
</table></td></tr>
</table>

9.3 Auswahl der Technologien und Methoden

Die Zuordnung von Rechten zu Schutzbedarfsklassen muss in Hinblick auf den jeweiligen Anwendungsfall erfolgen. Für die Bereitstellung der Identitätsinformationen können verschiedene Credentials Verwendung finden, die im Kapitel „Auswahl der Technologien und Methoden“ der Entwurfsmuster für die Authentifizierung aufgelistet wurden.

10. Beispiel zur Vorgehensweise

Das hier beschriebene Modell des Best Practise Ansatzes für Service-Orientierte Architekturen soll es dem Benutzer erlauben für seinen konkreten Fall mit einem festgelegten Verfahren die relevanten Sicherheitsmaßnahmen zu identifizieren. Das Verfahren sieht folgende Arbeitsschritte vor:

1. Identifizierung der zutreffenden Architektur
2. Identifizierung der zugehörigen Anwendungsfälle
3. Auswahl der Entwurfsmusters gemäß den Ergebnissen aus den Arbeitsschritten 1 + 2
4. Bestimmung des Schutzbedarfs der verarbeiteten Daten und Informationen
5. Auswahl der möglichen Technologien und Methoden, um für die identifizierten Architekturen und Anwendungsfälle den geforderten Schutzbedarf für die verschiedenen Schutzziele zu erfüllen
6. Auswahl der umzusetzenden Technologien und Methoden (Sicherheitsmaßnahmen) auf der Grundlage von bereits existierender Infrastruktur, die genutzt werden kann

Nach Arbeitsschritt 6 sind alle Sicherheitsmaßnahmen identifiziert, die für den untersuchten Fall relevant sind und umgesetzt werden müssen.

Im Folgenden wird die Vorgehensweise am Beispiel des Anwendungsszenarios eines Bestellvorgangs aus dem BSI Kompendium für Sicherheit in Service-orientierten Architekturen[3] erläutert. Dieses Kompendium wird im Folgenden kurz als SOA-Security-Kompendium bezeichnet.

[3] Sicherheit in Service-orientierten Architekturen, Version 2.0 des Bundesamt für Sicherheit in der Informationstechnik (BSI), Seite 296, Kapitel 7.7 Anwendungsszenarien.

Abbildung 5 zeigt vereinfacht den Bestellvorgang:

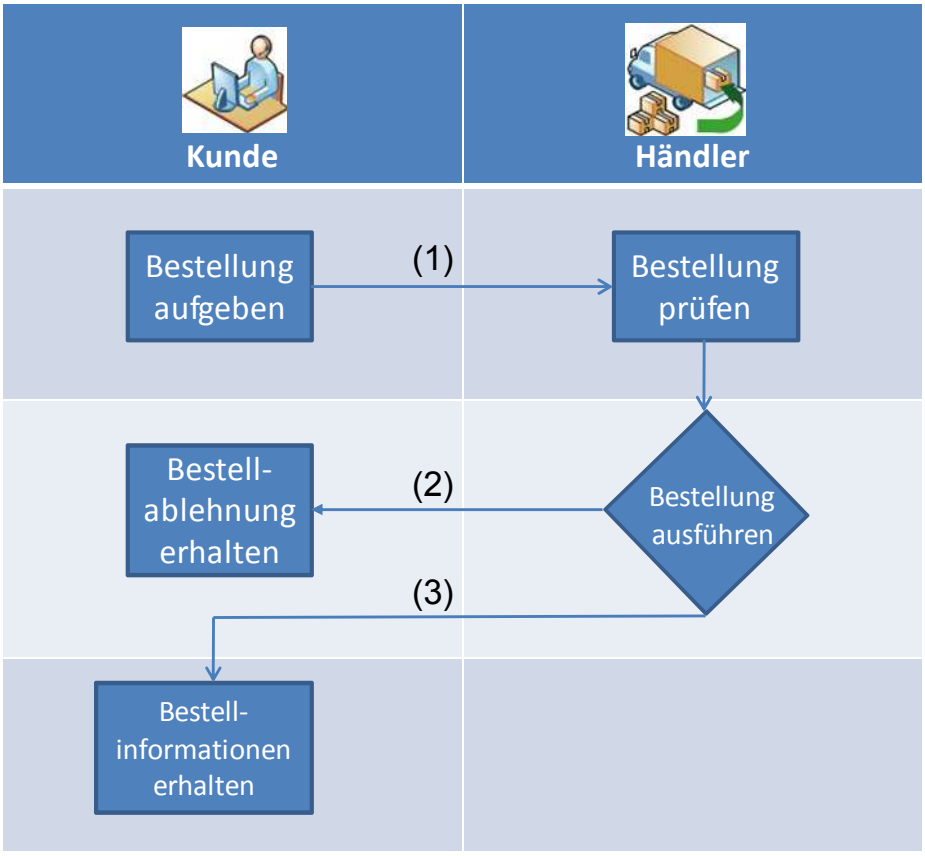


Abbildung 5: Bestellvorgang aus SOA-Security-Kompendium

In Abbildung 5 sind die einzelnen Kommunikationen mit Nummern versehen worden, um darauf im Folgenden referenzieren zu können.

10.1 Auswahl der Architektur

Für die in gezeigten Bestellvorgang gelten folgende Architekturen:

Kommunikation	Architektur
Nr. 1: Bestellung aufgeben; Nr. 2: Bestellablehnung erhalten	A4, A5
Nr. 1: Bestellung aufgeben; Nr. 3: Bestellinformationen erhalten	A4, A5

10.2 Auswahl der Anwendungsfälle

Für die in gezeigten Bestellvorgang gilt der Anwendungsfall DS_U1. Hierbei sendet ein Service Consumer einen Request und der Service Provider antwortet mit einer Response. Es werden jeweils Daten übermittelt.

Kommunikation	Anwendungsfall
Nr. 1: Bestellung aufgeben; Nr. 2: Bestellablehnung erhalten	DS_U1
Nr. 1: Bestellung aufgeben; Nr. 3: Bestellinformationen erhalten	DS_U1

10.3 Entwurfsmuster

Mit den ausgewählten Architekturen A4 und A5 und dem Anwendungsfall DS_U1 ergeben sich folgende mögliche Entwurfsmuster für Integrität und Vertraulichkeit.

Integrität:

	(A4) mit (A5) Indirekte Kommunikation beidseitige Vertrauensbeziehung
(DS_U1) Datenübermittlung	➔ (P1) Integrität des Request gewährleistet ➔ (P2) Integrität der Response gewährleistet ➔ (P3) Integrität des Requests und der Response gewährleistet

Vertraulichkeit:

	(A4) mit (A5) Indirekte Kommunikation beidseitige Vertrauensbeziehung
(DS_U1) Datenübermittlung	➔ (P1) Vertraulichkeit des Request gewährleistet ➔ (P2) Vertraulichkeit der Response gewährleistet ➔ (P3) Vertraulichkeit des Requests und der Response gewährleistet

10.4 Schutzbedarf der verarbeiteten Daten

Der Schutzbedarf wird hier beispielhaft gemäß SOA-Security-Kompendium angeben:

Kommunikation	Schutzziel	Schutzbedarf
Nr. 1: Bestellung aufgeben	Integrität	Hoch, da personenbezogene Daten übertragen werden
	Vertraulichkeit	Hoch, da personenbezogene Daten übertragen werden
Nr. 2: Bestellablehnung erhalten	Integrität	normal, da keine personen-bezogene Daten übertragen werden
	Vertraulichkeit	normal, da keine personen-bezogene Daten übertragen werden
Nr. 3: Bestellinformationen erhalten	Integrität	normal, da keine personen-bezogene Daten übertragen werden
	Vertraulichkeit	normal, da keine personen-bezogene Daten übertragen werden

10.5 Auswahl der Technologien und Methoden

Durch den Schutzbedarf sind folgende Technologien möglich:

Schutzbedarf	Technologien und Methoden	Erläuterung
normal	SSL/TLS XML Signature, XML Encryption	Wenn der Schutzbedarf der Message (Request oder Response) bzgl. Integrität und/oder Vertraulichkeit als normal eingestuft ist, so können die Technologien SSL/TLS oder XML Signature/XML Encryption eingesetzt werden.
hoch	XML Signature, XML Encryption	Integrität: Die Message wird gegen Manipulationen mittels XML Signature geschützt. Die digitale Signatur kann z.B. auf Basis von fortgeschrittener Signatur und Software-Tokens erfolgen. Vertraulichkeit: Die Message wird vor unautorisierter Einsicht mittels XML Encryption geschützt. Die Verschlüsselung kann z.B. mit symmetrischen oder asymmetrischen Verfahren realisiert werden. Die Anforderungen an die zu verwendenden Kryptoalgorithmen und Schlüssellängen sind nicht zu hoch (z.B. AES 128 Bit oder RSA 1024 Bit)

Folglich ergeben sich die in folgender Tabelle dargestellten Verhältnisse.

Kommunikation	Architektur	Anwendungs-fall	Schutzbedarf	Entwurfsmuster
Nr.1: Bestellung aufgeben; Nr.2: Bestellab-lehnung erhal-ten	A4, A5	DS_U1	Nr.1: hoch, Nr.2: normal	Integrität: P1 Vertraulichkeit: P1
Nr.1: Bestellung aufgeben; Nr.3: Bestellin-formationen erhalten	A4, A5	DS_U1	Nr.1: hoch, Nr.3: normal	Integrität: P1 Vertraulichkeit: P1

Somit muss der Service Request (Bestellung aufgeben) in diesem Beispiel mittels XML Signature und XML Encryption abgesichert werden. Für die Response (Bestellablehnung erhalten oder Bestellinformationen erhalten) muss nicht unbedingt eine Absicherung auf Datenebene erfolgen.

10.6 Umsetzung mit vorhandenen Technologien und bestehender Infrastruktur

Die Absicherung des Service Request (Bestellung aufgeben) mittels der Technologien XML Signature und XML Encryption kann z.B. auf Basis von digitalen Zertifikaten erfolgen. Besteht bereits eine Public Key Infrastruktur (PKI), so kann diese eingesetzt werden, um die hier bereitgestellten Zertifikate zu nutzen. Stellt die PKI aber z.B. keine Zertifikate für Verschlüsselung aus, muss die ausstellende CA (Certification Authority) um Zertifikatsprofile für die Verschlüsselung erweitert werden, ein Key Management (Key History) für die Verschlüsselungs-Keys muss realisiert werden und die Zertifikate müssen geeignet veröffentlicht werden.

11. Anhang

A. Klassifizierung der Webservice-Spezifikationen

Folgende Spezifikationen können genutzt werden, um die in den Entwurfsmustern beschriebenen Ansätze und Empfehlungen umzusetzen.

SOAP

Spezifikation:	http://www.w3.org/TR/soap/
Kategorie:	Messaging
Ziel:	Messaging Nachrichtenformat
Vorbedingungen:	-
Abhängigkeiten:	-

Beschreibung:

SOAP spezifiziert ein Nachrichtenformat zum standardisierten Austausch von strukturierten Informationen in einer dezentralen, verteilten Umgebung.

WS-Addressing

Spezifikation:	http://www.w3.org/Submission/ws-addressing/
Kategorie:	Messaging
Ziel/Konzept:	Messagingheader für Ende-zu-Ende-Kommunikation, Nachrichtenidentifizierung für Message-Exchange-Pattern
Vorbedingungen:	-
Abhängigkeiten:	SOAP

Beschreibung:

WS-Addressing ermöglicht die Identifizierung und Referenzierung eines Webservice-Endpunkts und spezifiziert SOAP-Header, um Informationen über den Nachrichtenaustausch in der Nachricht einzufügen. Diese Header können den Empfänger und den Absender der Nachrichte beschreiben, um ein Zustellen von Webservice-Nachrichten über mehrere Zwischenstationen (Ende-zu-Ende-Verbindung) unabhängig vom Transportprotokoll zu ermöglichen. Zudem kann eine Information hinzugefügt werden, die die Nachricht selbst identifiziert, um zusätzliche Message Exchange Pattern zu unterstützen – beispielsweise asynchrone Aufrufe.

WSDL

Spezifikation:	http://www.w3.org/TR/wsdl20
Kategorie:	Service-Metadaten
Ziel/Konzept:	Beschreibung des Service
Vorbedingungen:	-
Abhängigkeiten:	-

Beschreibung:
Die Web Services Description Language (WSDL) spezifiziert eine auf XML-basierende Sprache, um die Funktionalität eines Netzwerkdienstes abstrakt zu beschreiben. Zudem ermöglicht WSDL, dass diese abstrakte Beschreibung an eine konkrete Servicebeschreibung gebunden wird, welche spezifiziert „wo“ und „wie“ ein Service erreichbar ist.

WS-Policy

Spezifikation:	http://www.w3.org/TR/ws-policy/
Kategorie:	Service-Metadaten
Ziel/Konzept:	Policy
Vorbediungungen:	-
Abhängigkeiten:	-

Beschreibung:
Der W3C Standard WS-Policy definiert ein Modell und ein Framework, um Policies auszudrücken, die sich auf domänen-spezifische Anforderungen und Eigenschaften von Akteuren in einem Web-Service-basierten System beziehen. Dazu definiert WS-Policy eine Struktur, um Anforderungsalternativen als eine Reihe von einfachen oder verschachtelten und/oder-Verknüpfungen zu definieren. Das konkrete Vokabular zur Definition der Alternativen ist Teil von domainspezifischen Spezifikationen, wie WS-SecurityPolicy oder WS-PolicyAssertions.

WS-Policy Attachment

Spezifikation:	http://www.w3.org/TR/ws-policy-attach/
Kategorie:	Service-Metadaten
Ziel/Konzept:	Assoziation einer Policy mit Dienstbeschreibungen
Vorbedingungen:	-
Abhängigkeiten:	WS-Policy, (WSDL, UDDI)

Beschreibung:
WS-PolicyAttachment spezifiziert zwei generelle Mechanismen, um Webservice Policies mit dem Subject (bsp. Service, Endpunkt, Operation,…) einer Policy zu verbinden. Eine Policy kann durch ein externes Binding mit einem Subjekt verbunden werden, wozu WS-PolicyAttachment eine XML-Struktur

definiert. Zum anderen kann eine Policy aber auch direkt mit den Metadatan verknüpft werden, die ein Subjekt beschreiben. Dazu spezifiziert WS-PolicyAttachment, wie Policies mit WSDL-Definitionen und UDDI-Einträgern verknüpft werden können.

WS-MetadataExchange

Spezifikation:	http://www.w3.org/Submission/WS-MetadataExchange/
Kategorie:	Service-Metadaten
Ziel/Konzept:	Metadatenaustausch
Vorbedingungen:	-
Abhängigkeiten:	WS-Transfer

Beschreibung:
WS-MetadataExchange spezifiziert ein einnfaches Protokoll zum Abruf von Metadaten wie beispielsweise XSD, WSDL oder WS-Policy. Dazu spezifiziert der Standart, wie Metadaten als WS-Transfer-Ressource behandelt werden können.

WS-Security

Spezifikation:	http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.pdf
Kategorie:	Security
Ziel/Konzept:	Secure Messaging (Vertraulichkeit, Integrität und Nachrichtenauthentizität auf Nachrichtenebene)
Vorbedingungen:	Ende-zu-Ende-Kommunikation über Zwischenstationen
Abhängigkeiten:	SOAP

Beschreibung:
WS-Security spezifiziert eine Erweiterung von SOAP, um verschiedene Sicherheitstechnologien (beispielsweise PKI oder Kerberos) zur Absicherung von SOAP-Nachrichten nutzen zu können. Dazu beschreibt WS-Security die Verwendung von XML-Signiture und XML-Encryption in SOAP, um die Nachrichtenvertraulichkeit und -integrität zu realisieren. Zudem ermöglicht die Spezifikation zur Sicherstellung der Nachrichtenauthentizität, dass Identitätsinformationen (Zertifikate, Kerberos-Tokens, etc.) in Form von Security-Tokens an die Nachricht gebunden werden. Somit kann eine Ende-zu-Ende-Sicherheit von ausgetauschten Nachrichten gewährleistet werden.

WS-SecurityPolicy

Spezifikation:	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf
Kategorie:	Security
Ziel/Konzept:	Security Policy
Vorbedingungen:	WS-Policy
Abhängigkeiten:	WS-Security, (WS-Trust, WS-SecureConverstion)

Beschreibung:
WS-SecurityPolicy spezifiziert eine Sammlung von Policy-Assertions für die Nutzung in WS-Policy-Dokumenten, um sicherheitsbezogene Anforderungen und Fähigkeiten – hinsichtlich der Verwendung von WS-Security, WS-Trust und WS-SecureConversation – auszudrücken.

WS-Trust

Spezifikation:	http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.pdf
Kategorie:	Security
Ziel/Konzept:	Identity Service
Vorbedingungen:	Bereitstellung von Identitätsinformationen erforderlich
Abhängigkeiten:	WS-Security

Beschreibung:
WS-Trust beschreibt ein Webservice-Interface für das Ausstellen, Erneuern, Bestätigen und Verwerfen von Security Tokens, welches von einem sogenannten Security Token Service (STS) angeboten wird.

SAML

Spezifikation:	http://saml.xml.org/saml-specifications
Kategorie:	Security
Ziel/Konzept:	Security Token, Identity Service, Identity Federation
Vorbedingungen:	Austausch von Identitätsinformationen erforderlich
Abhängigkeiten:	-

Beschreibung:
SAML (Security Assertion Markup Language) ist ein von der OASIS Security Services Technical Committee (SSTC) standardisiertes XML-basiertes Framework für die Beschreibung und Abfrage von Authentifizierungs-, Autorisierungs- und Attributinformationen eines Nutzers über eine Reihe von Kommunikationsprotokollen wie HTTP oder SOAP.

WS-SecureConversation

Spezifikation:	http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.3/ws-secureconversation.pdf
Kategorie:	Security
Ziel/Konzept:	Secure Session
Vorbedingungen:	Austausch einer Nachrichtenfolge
Abhängigkeiten:	WS-Security

Beschreibung:
WS-SecureConversation spezifiziert die Erzeugung und Verwendung eines Sicherheitskontextes, um eine Folge von Webservice-Nachrichten mittels WS-Security unter Verwendung eines Sitzungsschlüssels absichern zu können. Grundlage stellt die Verwendung eines Security Context Tokens dar, welches den Sitzungsschlüssel kapselt und durch WS-SecureConversion beschrieben wird. Zudem beschreibt WS-SecureConversation die Erzeugung des Kontextes, wobei zu verwendende Security Token von einem STS bereitgestellt, von einem Kommunikationspartner erzeugt oder von beiden Parteien ausgehandelt werden kann.

WS-Federation

Spezifikation:	http://specs.xmlsoap.org/ws/2006/12/federation/ws-federation.pdf
Kategorie:	Security
Ziel/Konzept:	Identity Federation
Vorbedingungen:	Mehrzahl von Vertrauensdomänen
Abhängigkeiten:	WS-Security, WS-Trust, WS-SecurityPolicy

Beschreibung:
WS-Federation ist eine von IBM und Microsoft entworfene Spezifikation um Identity Federation in webbasierten und webservicebasierten Umgebungen umzusetzen. Identity Federation beschreibt dabei die gemeinsame Nutzung von Identitätsinformationen, welche von administrativ eigenständigen Instanzen verwaltet werden. WS-Federation definiert die Mechanismen, um unterschiedliche Sicherheitsdomänen zu einem Verbund (Federation) zusammenzuschließen, Identitäten im Verbund zu verbinden (Account Linking) und Identitätsinformationen zwischen diesen Domänen abzurufen und auszutauschen unter Beachtung der Privatsphäre des Benutzers.

TeleTrusT Deutschland e.V.

Der IT-Sicherheitsverband TeleTrusT Deutschland e.V. wurde 1989 gegründet, um verlässliche Rahmenbedingungen für den vertrauenswürdigen Einsatz von Informations- und Kommunikationstechnik zu schaffen. TeleTrusT entwickelte sich zu einem bekannten Kompetenznetzwerk für IT-Sicherheit. Heute vertritt TeleTrusT rund 100 Mitglieder aus Industrie, Wissenschaft und Forschung sowie öffentlichen Institutionen. In Projektgruppen zu aktuellen Fragestellungen der IT-Sicherheit und des Sicherheitsmanagements tauschen Forscher, Anbieter und Anwender von IT-Sicherheitstechnologie ihr Know-how aus und erarbeiten gemeinsam Lösungen. TeleTrusT äußert sich zu politischen und rechtlichen Fragen und organisiert Konferenzen und Messebeteiligungen. Die 'European Bridge CA', eine von TeleTrusT betreute Public-Key-Infrastructure-Anwendung für sichere E-Mail-Kommunikation, deckt bislang über 700.000 Public-Key-Zertifikate ab. Das Zertifizierungsprogramm "TeleTrusT Information Security Professional" (T.I.S.P.) qualifiziert Absolventen als Experten für IT-Sicherheit. Hauptsitz des Verbandes ist Berlin.

Kontakt:
Dr. Holger Mühlbauer
TeleTrusT Deutschland e.V.
Chausseestraße 17
10115 Berlin
Tel.: + 49 30 400 54 310

holger.muehlbauer@teletrust.de
www.teletrust.de


TeleTrusT
Pioneers in IT security.